E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Predicción de resultados de béisbol mediante técnicas de inteligencia artificial

Grado en Ingeniería Informática

Trabajo Fin de Grado

Alumno: Jarei Basabe López

Director: Mikel Sesma Sara

Pamplona, 7 de septiembre de 2021

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

AGRADECIMIENTOS

A mis amigos, quienes me han apoyado a océanos de distancia.

A Mikel Sesma, por su ayuda a lo largo de todo el proceso de este trabajo.

A todas aquellas personas que han contribuido al acceso público y gratuito del conocimiento científico, sin las que este y muchos otros estudios no habrían podido realizarse.

RESUMEN

Utilizando los datos y estadísticas oficiales de los partidos jugados en los últimos años de la Liga Mayor de Béisbol (MLB), se crean distintos modelos basados en el aprendizaje automático para la predicción de resultados en partidos futuros.

Para ello, se realiza una selección de las características más relevantes debido al gran volumen de datos disponible (incluyendo métricas básicas y compuestas de los jugadores y equipos, además de datos de carácter temporal sobre los últimos partidos jugados), con las cuales se entrena una serie de distintos modelos y se evalúa su eficacia a la hora de predecir nuevos resultados en base a datos previamente desconocidos.

Se analizan las técnicas empleadas en estudios previos, y se identifican las combinaciones de modelos, parámetros, y conjuntos de variables que maximizan la precisión de las predicciones.

PALABRAS CLAVE

Béisbol, baseball, sabermetrics, MLB, aprendizaje automático, machine learning

ÍNDICE

1. INTRODUCCIÓN

El béisbol, deporte conocido como el pasatiempo de los Estados Unidos (*the nation's pastime*), ha pasado de ser un juego recreacional a un deporte de élite con representación olímpica, así como una industria multimillonaria de carácter internacional.

Una de las cuestiones más importantes a considerar por los gerentes generales de cada equipo es la decisión de qué jugadores contratar teniendo en cuenta el número limitado de la plantilla. El gasto en ojeadores, la limitación a la hora de escoger jóvenes promesas en el *draft* anual[1], y la penalización económica[2] en caso de sobrepasar el límite salarial requieren estudiar qué factores o cualidades de los jugadores tienen una mayor repercusión en el resultado final de los partidos.

A lo largo de la historia del béisbol, la cantidad de datos comúnmente anotados ha ido en aumento: desde el más básico número de carreras por entrada, pasando por la invención del *box-score* que detalla diversas métricas tanto ofensivas como defensivas de cada jugador, hasta llegar al registro de cada jugada individual en la actualidad. A día de hoy, es común encontrarse con estadísticas como el OPS (*on-base plus slugging*), cálculo basado en 8 métricas básicas, en las retransmisiones televisivas. Nuevas tecnologías como *Statcas*t [2] o *PITCHf/x* [3] ponen al alcance tanto de equipos profesionales como de aficionados medidas anteriormente desconocidas como la velocidad de cada lanzamiento o el ángulo de salida de cada pelota bateada.

A diferencia de otros deportes, los posibles efectos de cada jugada toman valores discretos (p. ej.: avanzar de primera a segunda base, a diferencia de un pase en fútbol, cuyo efecto es más complejo de cuantificar), y los resultados de cada bateo conforman un conjunto finito de posibilidades.

Dichos efectos son fácilmente atribuibles a jugadores concretos y las acciones quedan registradas en los llamados *play-by-play* de fácil acceso por cualquier persona.

Por estas razones, el análisis estadístico y los modelos predictivos son ampliamente utilizados a la hora de realizar apuestas monetarias [4], [5] y para evaluar el efecto que jugadores individuales tienen (o podrían tener en caso de ser contratados) en la cantidad de victorias obtenidas por su equipo, lo cual conlleva un gran incentivo económico [6].

1 Regla 4(b) del reglamento oficial [1]
2 Regla 3(c)(4)(B) del reglamento oficial

1.1. OBJETIVOS

El principal objetivo de este estudio es la creación de un modelo capaz de predecir el resultado de partidos individuales de béisbol en base a características conocidas con antelación, como los resultados previos, las alineaciones iniciales o las estadísticas de los jugadores.

Por otra parte, se desea estudiar qué métricas son indicadoras del posible futuro rendimiento de los jugadores.

Para cumplir dichos objetivos, se consideran necesarios los siguientes pasos a seguir:

- Recolección de datos: Información sobre los partidos jugada a jugada, datos personales de los jugadores, métricas sobre los estadios.
- Limpieza y extracción de los datos recogidos.
- Creación de una base de datos relacional.
- Análisis estadístico de diferentes métricas y su influencia a la hora de ganar partidos.
- Creación de distintos modelos predictivos de aprendizaje supervisado basados en la clasificación binaria de ganadores y perdedores, utilizando distintas configuraciones de variables.
- Creación de distintos modelos predictivos de aprendizaje supervisado basados en la regresión del diferencial de puntos anotados, utilizando distintas configuraciones de variables.
- Análisis de los resultados obtenidos y comparación con otros estudios.

2. CONCEPTOS BÁSICOS

A continuación, explicamos algunos de los conceptos básicos del béisbol y de los métodos de aprendizaje automático utilizados posteriormente.

2.1. BÉISBOL

2.1.1. EL DEPORTE

El béisbol es un deporte jugado entre dos equipos de hasta 40 jugadores, 9 de los cuales forman la alineación en cualquier momento dado, con el objetivo de anotar más puntos que el equipo rival para ganar el partido.

El lanzador o *pitcher* lanza bolas hacia la zona en la que se encuentran el receptor o *catcher* y el bateador (mostrada en la Figura 1), el cual trata de golpear los lanzamientos que pueden superar los 160 km/h con su bate. Dichos lanzamientos se suceden hasta completar un turno al bate (*at bat*, AB), cuando el bateador queda eliminado o consigue un bateo válido, pasando el turno de bateo al siguiente jugador. Estos conceptos se especifican con mayor detalle más adelante y los términos y abreviaturas quedan recogidos en el Anexo i.

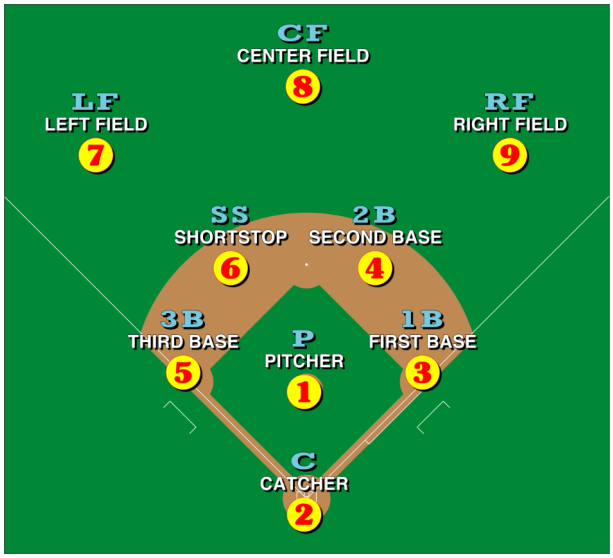

Figura 1: Posiciones típicas de los jugadores del equipo defensivo.
Imagen de Michael J. bajo licencia CC BY-SA 4.0

A diferencia de otros deportes, ambos equipos se turnan a la hora de tomar los roles de defensa y ataque, siendo los atacantes quienes batean mientras que los defensores tratan de evitar que sus rivales anoten carreras.

Para anotar una carrera, la cual equivale a un punto, el bateador debe recorrer las bases (mostradas en la Figura 2) en dirección anti-horaria sin ser eliminado, hasta llegar de nuevo a la posición inicial desde donde bateó.

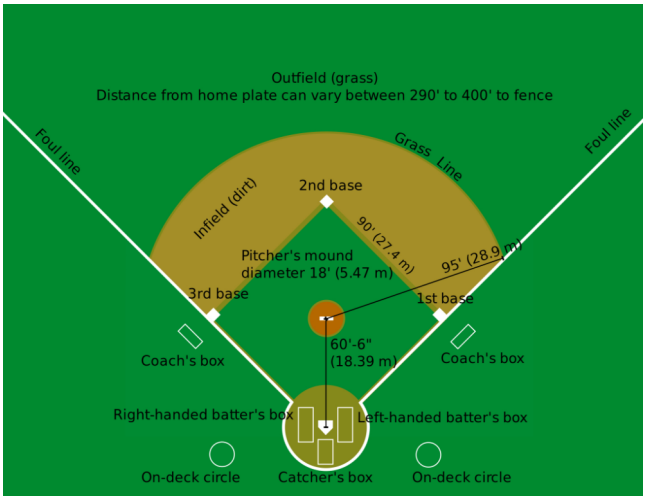

Figura 2: Especificaciones del campo de juego

Imagen de Cburnett bajo licencia CC BY-SA 3.0

Para ello, el bateador debe primero convertirse en corredor, generalmente tras un bateo exitoso entre las líneas de falta que delimitan el terreno de juego válido. Después de llegar de forma segura a la primera base, puede intentar seguir recorriendo el resto o esperar al turno del siguiente bateador, en el cual podrá avanzar de nuevo tras su bateo.

Cada vez que ambos equipos hayan finalizado su turno de bateo, se dice que han completado una de las nueve entradas o *innings* del partido, las cuales no tienen límite de tiempo sino que terminan tras conseguir eliminar a tres jugadores.

Si tras concluir las nueve entradas ambos equipos han anotado la misma cantidad de carreras, el partido continúa hasta que uno de ellos se encuentre por delante en el marcador al final de cualquier entrada.

El objetivo del equipo defensivo, sin embargo, es eliminar a los mencionados tres jugadores rivales para concluir la mitad del *inning* y así evitar que anoten carreras.

Las formas más comunes de eliminar al bateador son conseguir atrapar una pelota bateada antes de que esta toque el suelo o que el bateador cometa tres *strikes*.

Se considera un *strike* cada ocasión en la que el bateador no consiga contactar la pelota tras un intento de bateo o no trate de golpear una pelota que atraviese la zona de *strike* mostrada en la Figura 3.

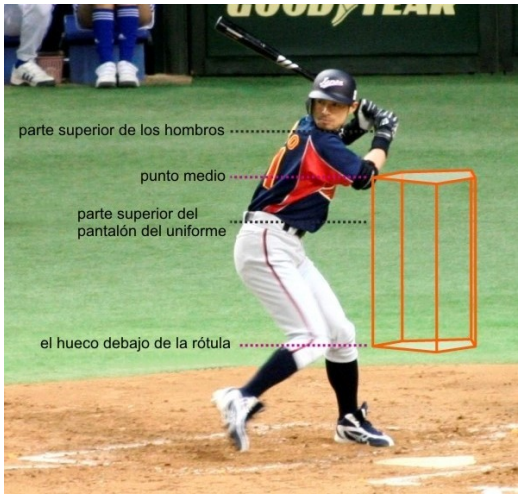

Figura 3: Zona de strike
Imagen de Mori Chan (original), Num (recorte y edición), y Uawiki (traducción) bajo licencia CC BY 2.0, recortada

Por otra parte, si el bateador no intenta golpear un lanzamiento que no atraviese dicha zona (llamado “bola”) en cuatro ocasiones, avanzará automáticamente a la primera base.

Si la pelota es bateada hacia el territorio de *foul* delimitado por las líneas blancas visibles en la Figura 2, se considera *strike* a no ser que este fuera a ser el tercer *strike* del bateador en cuestión, en cuyo caso el turno de bateo continua sin aumentar la cuenta de *strikes*.

Cualquier corredor puede también ser eliminado si un defensor en posesión de la pelota le toca mientras el corredor está fuera de la base, o si el defensor con la pelota consigue tocar la base a la que debe dirigirse en caso de avance obligado antes que el corredor (dos jugadores no pueden ocupar la misma base, por lo que tras un bateo exitoso el jugador en la primera base debe avanzar forzosamente).

La casuística de formas en las que un bateador puede avanzar a primera base (p. ej.: ser golpeado por la pelota (HBP) o que el lanzador cometa una infracción (*balk*)) o ser eliminado (p. ej.: causar una interferencia) es mucho mayor a la aquí mencionada, y las reglas que tratamos contienen múltiples excepciones, por lo que es recomendable consultar el Anexo i que contiene breves descripciones de los conceptos y el reglamento oficial [1] para un mayor detalle.

2.1.2. LA COMPETICIÓN

Existen diversas organizaciones que administran la forma en la que los partidos de béisbol se llevan a cabo y establecen sus conjuntos de normas tanto sobre las reglas del juego como sobre la gestión de los clubes.

Además de competiciones internacionales como las olimpiadas o la Serie del Caribe, decenas de países constan con ligas nacionales, siendo las más conocidas aquellas en los EE. UU. (MLB), Latinoamérica (LBPRC, LMP, LVBP, …), y Asia Oriental (NPB, KBO, CPBL).

En el caso de la MLB, cada equipo juega 162 partidos en la temporada regular durante un periodo aproximado de seis meses. Debido a la gran cantidad de encuentros, estos se dividen en series que enfrentan al mismo par de equipos de forma consecutiva en el mismo estadio, reduciendo así la cantidad de traslados necesarios.

La MLB cuenta con 30 equipos divididos en dos ligas: la Americana (AL) y la Nacional (NL). Estas están a su vez compuestas por las divisiones geográficas Este, Central, y Oeste. A pesar de ello, se disputan partidos entre las distintas divisiones y ligas de forma regular.

Los equipos con el mayor número de victorias de cada división avanzan automáticamente al campeonato anual de postemporada, junto los ganadores del “comodín” de cada liga (disputados entre los dos equipos con mayor cantidad de victorias de la liga que no se hayan clasificado de forma directa).

Cada enfrentamiento en la postemporada se decide en una única serie, por lo que estos partidos tienen una importancia mucho mayor.

2.2. INTELIGENCIA ARTIFICIAL

La inteligencia artificial es una rama de la informática que estudia el razonamiento computacional tras las decisiones comúnmente achacadas a “conductas inteligentes” de seres racionales, así como la creación de sistemas que realicen acciones o tomen decisiones semejantes a las llevadas a cabo por dichos seres inteligentes.

El aprendizaje automático o aprendizaje máquina (*machine learning*), por otra parte, es un campo dentro de la inteligencia artificial centrado en las formas por las que una máquina puede llegar a exhibir dichas conductas inteligentes sin que sea explícitamente programada para ello. Es decir, logra “aprender” basándose simplemente en la serie de datos que se le proporciona.

Para poder obtener resultados satisfactorios a la hora de realizar estos procesos, son necesarios distintos pasos que se ocupan de la obtención de la información, su limpieza y tratamiento para un correcto uso, su análisis estadístico para observar relaciones o tendencias, etc.

Todos estos factores son necesarios para que podamos crear un sistema capaz de aprender adecuadamente.

Una de las formas para llevar a cabo el aprendizaje es mediante la técnica denominada “aprendizaje supervisado”, basada en el uso de ejemplos en los que, además de unos datos de entrada, se incluye también el resultado que se sabe es correcto con antelación y se espera obtener. Después de que el sistema se entrene con estos ejemplos, será capaz de generalizar lo aprendido para predecir el resultado relativo a datos que no hayan sido previamente utilizados para el entrenamiento.

Para ello, es necesario obtener una serie de ejemplos previamente anotados, lo cual no siempre es posible o puede resultar muy costoso. En caso de no disponer de ellos, existen técnicas de aprendizaje no supervisado con las que poder lograr objetivos similares.

Dentro del aprendizaje supervisado, existen problemas de regresión y clasificación.

En los problemas de clasificación, se pretende predecir la clase a la que pertenece el conjunto de datos de entrada. Pueden emplearse dos o más clases diferentes, como por ejemplo la clasificación de correos electrónicos en las clases “*Spam*” y “No *spam*” o la clasificación de fotografías de animales en “Perro”, “Gato”, “Caballo”, “Rana”… Por otra parte, en los problemas de regresión se desea obtener un valor numérico como el precio de una casa o la cantidad de litros de lluvia acumulada.

En el problema que nos atañe, disponemos de los resultados finales de los partidos que ya han sido disputados en años previos, por lo que emplearemos métodos de aprendizaje supervisado detallados a continuación.

A pesar de que se trate de un problema de clasificación binaria para determinar el equipo ganador, utilizamos también técnicas de regresión para predecir la diferencia de carreras

entre ambos equipos (carreras del equipo local – carreras del equipo visitante), la cual puede utilizarse para decidir el ganador.

Los datos de entrada se componen por una serie de estadísticas conocidas con antelación al inicio de cada partido que incluyen información sobre ambos equipos y sus jugadores, el estadio, u otra información sobre el estado de la temporada. La selección de estas variables es elaborada más adelante.

2.2.1. NAÏVE BAYES

Podemos utilizar el teorema de Bayes, mostrado en la Fórmula (2), para calcular la probabilidad de que un partido dado por su vector de datos $\vec{x}=(x_1,x_2,...,x_n)$ resulte en una victoria del equipo local $P(local|\vec{x})$ o del equipo visitante $P(visitante|\vec{x})$.

$$P(y|\vec{x})=\frac{P(\vec{x}|y)\cdot P(y)}{P(\vec{x})} \quad (1)$$

Para facilitar el cómputo, ya que calcular la probabilidad de que se den todas las condiciones que conforman el vector de datos relativo al partido que se esté tratando en cada momento es muy costoso y complejo, los modelos *naïve Bayes* asumen la independencia entre todo par de variables.

A pesar de que esta condición no se cumpla en la realidad, los resultados obtenidos con este método pueden ser de interés.

Al asumir que todo x_i es independiente de x_j, podemos simplificar el término mostrado en la Fórmula (2).

$$P(x_i|x_1,x_2,...,x_{i-1},x_{i+1},...,x_n,y)=P(x_i|y) \quad (2)$$

Es decir, bajo la asunción de independencia, el hecho de que el resto de x_j se hayan dado no altera de ninguna manera la probabilidad de que ocurra x_i, únicamente el resultado de la predicción es relevante.

La Fórmula (2) puede ser, por tanto, reescrita como se indica en la Fórmula (3).

$$P(y|\vec{x})=\frac{P(\vec{x}|y)\cdot P(y)}{P(\vec{x})}=\frac{P(x_1|y)\cdot ...\cdot P(x_n|y)\cdot P(y)}{P(\vec{x})} \quad (3)$$

Dado que $P(\vec{x})$ está presente en el cálculo de ambas probabilidades en caso de victoria local o visitante, podemos obviarlo ya que corresponde al mismo valor numérico, dando lugar a la Fórmula (4).

$$P(y|\vec{x})\propto P(y)\cdot\prod_{i=1}^{n}P(x_i|y) \quad (4)$$

La probabilidad de que un partido acabe en victoria local o visitante (*P(y)*) es una constante obtenida del conjunto total de datos, y la probabilidad de que la variable x_i tenga

el valor del ejemplo dándose la victoria del equipo "*y*" se aproxima utilizando la estimación del máximo a posteriori.

Existen diversas posibilidades para obtener el valor de esta estimación, dependiendo de la distribución que se asuma que siguen las variables.

A la hora de realizar la predicción, elegiremos aquella clase que obtenga una probabilidad mayor en la Fórmula (4).

Al tratarse de un método de cálculo de probabilidades, no dispone de la capacidad para predecir valores numéricos continuos, por lo que utilizamos *naïve Bayes* únicamente para la clasificación binaria.

2.2.2. k-NN

El método de los *k* vecinos más cercanos consiste en predecir para cada *input* la clase mayoritaria entre los *k* ejemplos conocidos que se encuentren más próximos al conjunto de datos de entrada.

Cada una de las variables de entrada representa una dimensión del espacio donde se encuentran los ejemplos con coordenadas dadas por los valores que toman dichas variables. Entre ellos, se escogen los *k* más cercanos respecto al dato de entrada utilizando una métrica de distancia.

El funcionamiento de este método consiste en la asunción de que aquellos vectores de datos que son parecidos tienden a ser pertenecientes a la misma clase.

Cabe notar, por tanto, que la escala de cada variable es de gran relevancia, ya que las variables que toman valores muy distantes tienen un impacto mayor en la distancia.

En lugar de predecir la clase mayoritaria entre los vecinos más cercanos de un punto dado, conociendo el valor del diferencial de carreras de los partidos cercanos puede calcularse el resultado como la media aritmética de todos ellos.

2.2.3. REGRESIÓN LOGÍSTICA Y LINEAL

A la hora de predecir valores continuos, uno de los modelos frecuentemente utilizados es la regresión lineal, en el que el valor predicho viene dado por la Fórmula (5).

$$y = \beta_0 + x_1\beta_1 + ... + x_n\beta_n \tag{5}$$

Los valores de β_i actúan como pesos que modifican el efecto de cada variable sobre el resultado a predecir.

Teniendo en cuenta que los resultados posibles no están acotados, a la hora de predecir una variable binaria se aplica una función sigmoide o logística (Fórmula (6)) que solo produce valores en el rango [0, 1], los cuales se pueden interpretar como las probabilidades de que el conjunto de datos introducido corresponda con la clase positiva.

$$S(x) = \frac{1}{1+e^{-x}} \tag{6}$$

Para discretizar el valor de dicha probabilidad, se puede tomar una frontera de decisión (usualmente en 0.5) que asigne todo valor superior a la clase positiva y los valores inferiores a la clase negativa.

Como hemos mencionado anteriormente, la regresión lineal dada por la Fórmula (5) se utiliza para calcular el valor resultante, por lo que el modelo de regresión es análogo, sin la necesidad de acotar los valores mediante la función sigmoide.

2.2.4. SVM

El objetivo principal de las *Support Vector Machines* es separar el espacio N-dimensional (siendo N el número de variables de entrada) de modo que la mayoría de los ejemplos de cada clase queda a un u otro lado de la frontera. Además, esta frontera trata de maximizar el margen, de modo que la distancia entre los ejemplos más cercanos de cada clase sea la máxima posible. A estos ejemplos más cercanos se les denomina vectores soporte.

Tras crear el hiperplano que separa el espacio, cualquier nuevo dato puede ser clasificado como perteneciente a una u otra clase dependiendo de en qué lado del vector recae.

Ya que puede que una separación perfecta sea imposible dependiendo de la distribución de los datos, la frontera consta de un cierto margen de error que permite una clasificación inicial errónea para alguno de los datos de entrenamiento, con el objetivo de crear un modelo más robusto y representativo de la realidad.

Por otra parte, para convertir el problema en uno linealmente separable (es decir, hacer posible que una frontera separe los datos de entrenamiento sin ninguna clasificación errónea), pueden proyectarse los datos a un espacio de mayor dimensión mediante el uso de, por ejemplo, *kernels* polinomiales de órdenes superiores.

Por el contrario que en el problema de clasificación en el que se quieren separar los datos distintos, las SVM pueden utilizarse para la regresión mediante los llamados *Support Vector Regressor* (SVR) que buscan generar una línea lo más cercana posible a los datos de entrenamiento. De forma complementaria a SVM, los datos que quedan a una distancia menor o igual al margen ε no contribuyen al error, a diferencia de aquellos más alejados.

2.2.5. ÁRBOL DE DECISIÓN

Los modelos basados en árboles ofrecen una fácil comprensión del razonamiento tras las decisiones tomadas, haciéndolos unos de los más interpretables.

Un árbol de decisión se forma a través de nodos que dividen el conjunto total del nodo en distintas ramas en base a una de las variables de entrada, p.ej.: $x_i > k$. Los elementos que cumplen la regla de dicho nodo pasan a una de las ramas, mientras que los elementos que no la cumplen son asignados a otra rama. De forma recursiva, se crean más nodos que dividen el conjunto por otras variables y/u otros valores hasta alcanzar las “hojas”, en

las cuales no se producen más divisiones. Los elementos de cada hoja son clasificados como la clase mayoritaria de dicha hoja.

Para evitar sobre-ajustar el modelo al conjunto de entrenamiento, se consta con una serie de medidas como limitar el número máximo de nodos, forzar el mínimo de elementos para poder seguir dividiendo los nodos, limitar la profundidad máxima de las ramas, …

Los árboles de decisión funcionan de la misma manera para los problemas de regresión, salvo que el valor asignado a las hojas es uno numérico, el cual puede ser calculado como la media de los elementos de entrenamiento asignados a dicha hoja.

2.2.6. RANDOM FOREST

El modelo conocido como *random forest* se construye como un conjunto de árboles de decisión de menor tamaño. A diferencia de los árboles de decisión que basan la división de las ramas en aquellas que satisfacen un cierto criterio como la ganancia de información, los *random forests* tienen un componente de aleatoriedad que les otorga una mayor diversidad.

En primer lugar, cada árbol se entrena con un subconjunto aleatorio de los datos de entrenamiento escogidos con remuestreo, una técnica conocida como *bootstrapping*. Por otra parte, cada nodo de los árboles solamente tiene la posibilidad de dividirse en ramas utilizando un subconjunto aleatorio de las variables de entrada.

Gracias a estas restricciones, el *ensemble* de árboles se ve forzado a utilizar atributos diversos y a adaptarse a distintos datos de entrada, otorgándole una mayor flexibilidad y adaptabilidad a todo tipo de *inputs* a pesar de que cada árbol no sea tan eficaz en la totalidad de forma individual.

A la hora de clasificar elementos, cada uno de los árboles aporta su predicción y la clase más común es escogida.

Al utilizar las predicciones de múltiples árboles más simples, ninguno de ellos es tan propenso a sobre-aprender. Sin embargo, el beneficio de la interpretabilidad que aportan los árboles individuales no es tan claro en este método ya que pueden llegar a usarse cientos de árboles de decisión sin ser ninguno una pieza clave al calcular las clasificaciones por votación mayoritaria.

A la hora de utilizar *random forests* para tareas de regresión, las hojas representan valores numéricos y el resultado final es la media de todas las predicciones individuales.

2.2.7. ADABOOST

De forma similar a los *random forests*, AdaBoost se basa en un conjunto de predictores débiles que toman la decisión de manera colectiva. Pese a que habitualmente se utilizan árboles binarios como los modelos básicos, puede llevarse a cabo con otros modelos.

En nuestro caso, utilizamos árboles, los cuales constan de un único nodo (también conocidos como *stumps*).

En lugar de crear los árboles de forma independiente, los errores cometidos por árboles previos influyen la creación de los nuevos árboles. Cada ejemplo de entrenamiento lleva asociado un peso, el cual es incrementado (o reducido) cada vez que un árbol lo clasifica incorrectamente (o correctamente) de acuerdo a la importancia del árbol en cuestión. De este modo, se hace un mayor énfasis en aquellos ejemplos más difíciles de clasificar, y cada nuevo árbol creado tiende a cubrir esos ejemplos en mayor medida.

A la hora de realizar las votaciones, cada árbol tiene un peso diferente dependiendo de su error. A diferencia de la votación mayoritaria o media aritmética de *random forest*, las predicciones de los árboles con mayor precisión en iteraciones previas tienen un peso mayor.

Este método se utiliza tanto para clasificación como para regresión, de forma análoga a la comentada anteriormente.

2.2.8. XGBOOST

XGBoost, o *Extreme Gradient Boost*, es otro modelo de múltiples árboles.

En primer lugar, se parte de un valor dado por la media o la clase más común del conjunto de entrenamiento. Tras ello, se calculan los residuos o diferencias entre los valores actuales y la constante, los cuales se utilizan para generar un árbol de poca profundidad que los prediga. De esta forma, la predicción actual viene dada por la constante más el valor residual de la hoja relativa al ejemplo de entrenamiento.

De forma iterativa, los residuos de iteraciones siguientes se calculan respecto a la última predicción, creando nuevos árboles que predicen dichos residuos. Se suman los valores de todos los árboles generados de esta forma para calcular el nuevo resultado junto con la constante inicial.

Los valores residuales están ponderados por el ratio de aprendizaje, lo cual hace que cada iteración compense el error previo ligeramente, la base del descenso por gradiente.

Utilizamos XGBoost para los casos de clasificación y regresión.

2.2.9. RED NEURONAL

Las redes neuronales usan un conjunto de perceptrones (cuya idea está basada en el funcionamiento de las neuronas humanas) en distintas capas para obtener una probabilidad indicativa de la clase a la que pertenece el conjunto de entrada. Se muestra su estructura general en la Figura 4.

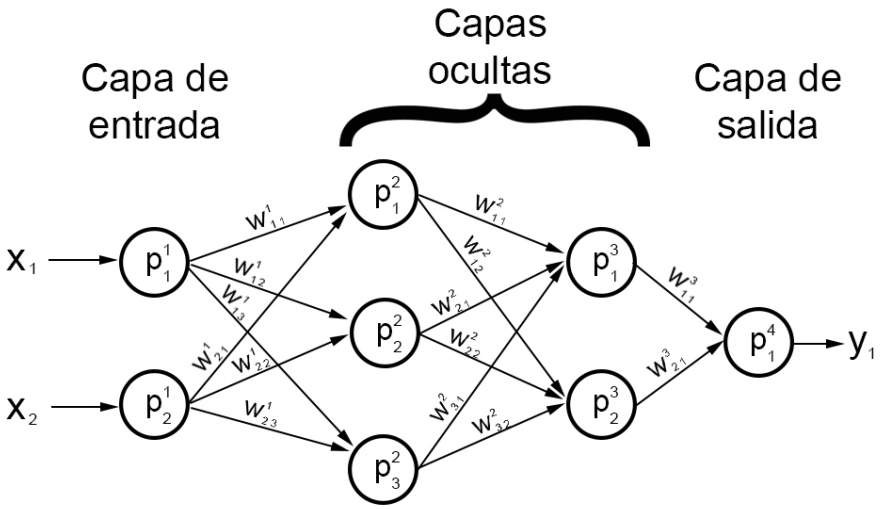

Figura 4: Ejemplo de la estructura de una red neuronal

La primera de las capas, la capa de entrada, contiene los valores que toman los atributos de las instancias.

Cada una de las neuronas de una capa recibe información de todas las neuronas de la capa anterior y, similarmente, envía su información a todas las de la siguiente capa. Los valores transmitidos entre cada par de neuronas se ven afectados por el peso de la conexión, indicado en la Figura 4 como $w^{c}_{i\,j}$, así como por la función de activación no lineal y un valor constante de *bias*, incluido en la Figura 5.

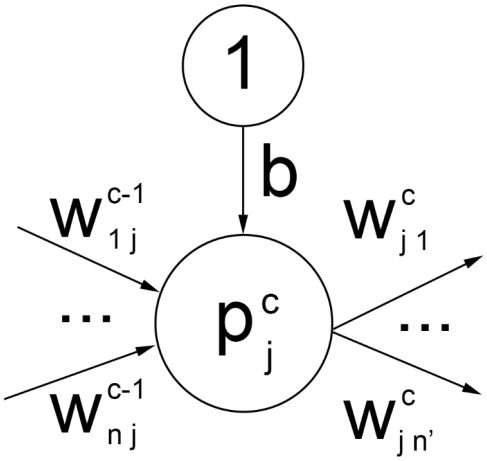

Figura 5: Ejemplo de una neurona de la red

La ecuación de la j-ésima neurona de la c-ésima capa es por tanto como se indica en la Fórmula (7), donde *f* es una función de activación y *b* es el peso correspondiente al factor de *bias*.

$$p_j^c = f\left(1 \cdot b + \sum_{i=1}^{N} p_i^{c-1} \cdot w_{ij}^{c-1}\right) \tag{7}$$

Las funciones de activación como *tanh* (Fórmula (8)) o *relu* (Fórmula (9)) definen el valor de salida de las neuronas, y son las que añaden un componente no lineal al método, ya que de otra forma la salida de una neurona sería simplemente una suma ponderada de sus *inputs*.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{8}$$

$$relu(x) = \begin{cases} x \; si \; x > 0 \\ 0 \; si \; x \leq 0 \end{cases} = max(0, x) \tag{9}$$

En la medida en la que se realizan las predicciones, se evalúa el error de la red neuronal respecto al valor real que debería predecirse, el cual se usa para actualizar los pesos del resto de neuronas de forma iterativa.

Debido a la alta complejidad e interconectividad entre las distintas variables, la lógica tras los resultados obtenidos es difícil de comprender ya que cada peso no afecta únicamente al par de neuronas que une sino que se propaga sobre el resto de la red.

3. OBTENCIÓN DE DATOS

Para este estudio, hemos decidido utilizar información sobre cada jugada ocurrida en partidos de la MLB (incluyendo la liga nacional (NL) y la liga americana (AL)) durante las temporadas de 2010 hasta 2019, ambas inclusive.

Decidimos no incluir datos sobre otras ligas como la KBO o la NPB (ligas coreana y japonesa, respectivamente) ni de partidos entre selecciones nacionales para mantener la uniformidad de los datos frente a distintas fuentes y para garantizar la relativa homogeneidad entre los partidos incluidos, ya que cada país tiene un estilo de juego y conjunto de normas diferenciado.

Durante el periodo comprendido entre las temporadas 2010 y 2019, las reglas en la MLB no sufrieron cambios mayores, a diferencia de la temporada 2020, que se ha visto alterada de forma considerable (p. ej.: a partir de la novena entrada, un corredor es automáticamente posicionado en la segunda base, o la reducción a 7 entradas por partido en los partidos dobles).

3.1. FUENTES DE DATOS

Una de las principales fuentes de información en el ámbito del béisbol es la *Lahman's Baseball Database* [7], que incluye información sobre jugadores, gerentes, estadios, premios, etc. desde 1871 hasta la actualidad abarcando las 6 mayores ligas. Sin embargo, los datos se encuentran agregados de forma anual, perdiendo así la mayor granularidad necesaria para nuestros objetivos.

Baseball-reference [8] ofrece una gran variedad tanto de datos agregados como jugadas individuales, añadiendo también otra información como la probabilidad esperada de ganar el partido al comienzo de cada bateo o las jugadas más influyentes del partido. Esta página es actualizada de forma diaria, por lo que contiene información sobre los partidos más recientes no disponible en otras fuentes.

A pesar de ofrecer un motor de búsqueda altamente detallado, está más orientado al uso individual o esporádico a diferencia de nuestra necesidad exhaustiva, que requeriría un mayor esfuerzo de *web scraping*.

De forma similar, la página oficial de la MLB [9] ofrece información detallada de cada partido visualizable en el navegador, sin opción a descargar los datos.

Retrosheet [10] es una de las fuentes más completas y accesibles, siendo mantenida por voluntarios. Más de cien años de partidos con datos de cada jugada concreta están disponibles para su descarga, así como otra información sobre jugadores o estadios.

En nuestro caso, por tanto, hemos optado por esta última para las jugadas de cada partido en los años 2010-2019.

Por otra parte, las fechas de nacimiento y debut han sido obtenidas de *Baseball Databank* [11].

3.2. PARSING

Los archivos obtenidos de *Retrosheet* son los siguientes:

- ***YYYYTTT***.EV***X***: contienen los eventos ocurridos en todos los partidos de la temporada regular del año ***YYYY*** en los que el equipo ***TTT*** jugó como equipo local, siendo ***X*** la liga (A para la liga americana y N para la liga nacional).
- ***YYYYL***.EVE: de forma análoga a los archivos .EVX, contiene los eventos ocurridos en los partidos de la postemporada del año ***YYYY***, indicando la fase (*Wildcard, Division Series, League Championship Series, World Series*) mediante ***L***.
- ***TTTYYYY***.ROS: lista los jugadores disponibles por el equipo ***TTT*** en la temporada regular y postemporada del año ***YYYY***, incluyendo información como la mano dominante con la que lanzan y batean o la posición defensiva que ocupan. Este último parámetro presenta valores inexactos y erróneamente formateados: “OF” (*outfielder*, jardinero exterior*)* engloba en múltiples ocasiones los posibles valores “LF”, “CF” y “RF” (*left fielder*, *center fielder*, *right fielder*; jardinero izquierdo, central y derecho, respectivamente). Algunos valores incluyen un espacio en blanco tras la abreviatura correspondiente.
- TEAM***YYYY***: listado de equipos participantes en ambas ligas en la temporada regular y postemporada del año ***YYYY***.

El archivo *People.csv* de *Baseball Databank* [11] es usado para extraer las columnas que contienen el identificador de cada jugador, su fecha de nacimiento y la fecha en la que realizaron su debut en la MLB.

La página “*The Event File*” [12] detalla el formato seguido por los archivos .EVX, de estructura similar a un archivo de valores separados por comas, salvo que la cantidad de columnas por cada línea es variable en base a los valores tomados por distintos campos, y las columnas adquieren distintos significados dependiendo de los valores que las preceden.

Es por ello que la tarea de creación de la base de datos no puede ser una conversión directa basada en archivos CSV, sino que es necesario tratar cada línea en base a la multitud de criterios establecidos en la página mencionada.

3.3. CREACIÓN DE LA BASE DE DATOS

Una vez *parseados* los ficheros de eventos, procedemos a la creación de una base de datos relacional utilizando SQLite, herramienta que ofrece una fácil integración con *Python* mediante la librería *sqlite3*.

Cada línea de los ficheros .EVX anteriormente mencionados puede contener distintas clases de información, como metadatos relativos a la forma en la que se anotó el partido en cuestión, condiciones meteorológicas, alineaciones iniciales y sustituciones, comentarios explicativos, etc.

Basándonos en las tuplas de tipo "*info"*, creamos una tabla con los datos relativos a cada partido, incluyendo los equipos enfrentados, la fecha y hora en la que dio comienzo y otros valores como el estadio y su asistencia. No todos los partidos contienen la información completa sobre las condiciones meteorológicas o del terreno de juego, por lo que no los incluimos en la base de datos ni en los subsiguientes procesos de predicción.

Entre las múltiples clases de información, las filas marcadas como "*play"* nos son las más relevantes, ya que especifican lo sucedido en cada jugada en gran detalle.

Cada registro de este tipo contiene una serie de campos que indican: el *inning* actual, qué equipo está bateando, el código identificador del bateador, el número de *strikes* y bolas lanzadas, la secuencia y tipos concretos de dichos lanzamientos y el resultado final del bateo.

El resultado final se encuentra a su vez subdividido en secciones que detallan, a grandes rasgos, el tipo de bateo (en caso de haberse sucedido), qué jugadores defensivos toman parte en las posibles eliminaciones y los avances de corredores que ya ocupan una base.

Cabe destacar que la mayoría de estas acciones contienen información implícita sobre los jugadores que las realizan, siendo únicamente especificadas las posiciones de dichos jugadores en lugar de sus códigos de identificación personal. Por ejemplo, una asistencia atribuida al jugador "7" del equipo "0" se refiere al *left fielder* del equipo visitante.

Es por ello que es necesario tratar los datos de forma cronológica para poder conocer el estado del partido en cada momento. Al trasladar esta información a la base de datos, sin embargo, incluimos la configuración actual de los jugadores (p. ej.: los códigos de cada corredor ocupando las bases y la alineación defensiva en el momento de suceder la jugada) para una ejecución más rápida y simple a la hora de realizar cálculos futuros.

Tras iterar sobre todas las jugadas de un partido dado, obtenemos el resultado final de cuántas carreras ha anotado cada equipo y el consiguiente ganador.

3.3.1. MODELO RELACIONAL

La Figura 6 muestra el diagrama del modelo relacional diseñado, omitiendo algunos campos y tablas por razones de brevedad.

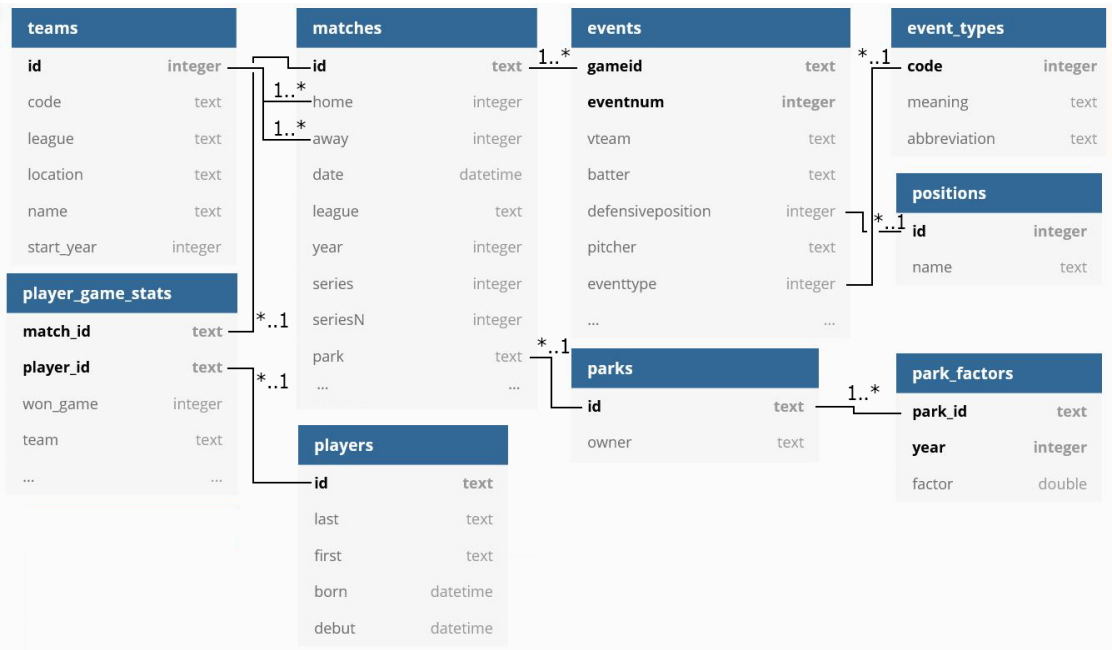

Figura 6: Diagrama del modelo relacional de la base de datos

Los archivos TEAMYYY se ven reflejados en la tabla **teams**, que contiene la información básica de cada equipo para una mayor legibilidad a la hora de presentar y visualizar los datos. Incluimos también el año de su inicio, ya que los Houston Astros pasaron de ser un equipo de la liga americana a la nacional en 2013, a pesar de mantener el mismo código "HOU".

Cada jugada en los archivos .EVX se corresponde con una fila en la tabla **events**, después de haber expandido la información implícita de los jugadores partícipes en el instante de la jugada. De esta forma, se puede analizar el resultado de la jugada y las condiciones previas a ella de forma independiente al resto de los eventos anteriores.

La tabla contiene 97 columnas, que incluyen toda la información ofrecida en los ficheros empleados para su construcción salvo los árbitros del partido, que no son considerados en nuestro análisis.

La tabla **matches** recoge la información y meta-información anteriormente mencionada, junto con el resultado final de cada partido. Incluimos también los campos *series* y *seriesN* que reflejan el orden en el que se jugaron los partidos, ya que éstos se llevan a cabo en "series" de múltiples partidos entre los mismos rivales de forma consecutiva.

Los datos básicos de los jugadores quedan plasmados en la tabla **players**, donde incluimos la información proveniente de *Baseball Databank* [11] para poder contar con sus edades y los años de experiencia que tienen a la hora de cada enfrentamiento.

La tabla **parks** relaciona los estadios con los equipos a los que corresponden.

Hemos creado las tablas auxiliares **event_types** y **positions** para una mejor legibilidad a la hora de tratar con variables de tipo enumerado, y así poder utilizar sus representaciones alfabéticas. P. ej.: “K” o *Strikeout* en lugar del número “3”, o la abreviatura común “SS” (*shortstop*) a diferencia del número “6”.

A raíz de los datos obtenidos en **events**, creamos la tabla **player_game_stats** que agrega los datos por cada par único de partido y jugador, para reducir el futuro tiempo de ejecución, ya que una gran parte de las consultas se realiza en base al rendimiento de cada jugador en una serie de partidos. Los valores anotados en estas tablas son esencialmente análogos, ignorando aquellos factores que pueden no mantenerse constantes a lo largo del partido como el número de compañeros en base o la mano dominante del lanzador entre otros.

De forma similar, creamos tablas que almacenan la suma acumulada de los valores en **player_game_stats** durante los últimos 3, 5, y 10 partidos así como los datos cumulativos totales hasta la fecha para cada uno de los partidos. Estas tablas se utilizan para una obtención de datos más rápida a la hora de entrenar y ejecutar los modelos predictivos.

3.3.2. DETECCIÓN DE INCONSISTENCIAS Y OUTLIERS

Una vez generada la base de datos, detectamos una serie de valores inesperados o que pueden dificultar la futura predicción.

Los siguientes jugadores han realizado apariciones en partidos con fechas anteriores a su supuesto debut, mostrados en la Tabla 1.

Tabla 1: Discrepancias en los debuts de jugadores

ID de jugador	Debut erróneo	Debut correcto
mckad001	2019-05-20	2019-05-19
mondr003	2016-07-26	2015-10-30
murps001	2019-09-04	2019-05-19
neuss001	2019-08-30	2019-05-19
sotoj001	2018-05-20	2018-05-15

Es posible que otras fechas de debuts sean también incorrectas, dado que los partidos que hemos estudiado son solo aquellos ocurridos entre los años 2010 y 2019.

En estos casos, hemos modificado las fechas incorrectas de forma manual.

Al menos un partido ha sido registrado con la hora incorrecta: el primer partido entre los Dodgers y Mets, jugado el 27 de abril de 2010, comenzó a las 4:10 de la tarde, a pesar de que aparece como las 4:10 de la mañana. Hemos corregido este dato manualmente.

El partido entre los Cubs y los Pirates jugado el 29 de septiembre de 2016 fue suspendido por las condiciones meteorológicas, resultando en un empate, el único en el conjunto de datos utilizado.

A la hora de calcular la cantidad de partidos ganados o hacer predicciones, ignoramos este caso, ya que no se ajusta a formato binario de victoria/derrota esperado.

25 partidos tuvieron lugar en estadios que no pertenecen a ningún equipo de la MLB, dado que algunos partidos se juegan en otros países como México o Japón en ocasiones especiales, así como partidos jugados en campos de ligas menores o bases militares.

Por otra parte, algunos partidos que quedaron suspendidos debido a condiciones meteorológicas u otros acontecimientos ocurridos en la ciudad (p. ej.: la cumbre del G20, conciertos, protestas, etc.), fueron jugados en el estadio del equipo rival.

Estos casos han sido excluidos para los cálculos del factor campo (*park factor*), causando ligeras discrepancias respecto a las estadísticas oficiales.

4. ESTUDIO ESTADÍSTICO

Todas las variables recogidas en la tabla **player_game_stats**, sobre las cuales basamos una gran parte del estudio estadístico, únicamente toman valores enteros no negativos, ya que representan el conteo de las ocasiones en las que ha ocurrido un cierto evento (p. ej.: cuántos *home runs* ha anotado el jugador, cuántos lanzamientos han resultado en *strike*, etc.).

A la hora de comparar estos datos entre distintos jugadores, años o ligas, se deben normalizar para dar un peso proporcional a la cantidad de ocasiones que han tenido para ocurrir dichos eventos. Por ejemplo, un jugador que anota 10 *hits* en 10 bateos no es análogo a quien anota 10 *hits* en 500 bateos.

Las métricas relativas al bateo toman valores en el rango [0, 1] tras dividirlas entre la cantidad de veces que el jugador completa un turno al bate (*at bats*, AB).

De forma similar, las métricas de los lanzadores toman también valores en [0, 1] al dividirlas entre la cantidad de bateadores a los que se enfrentan (*batters faced*, BF) o la cantidad de bolas lanzadas (*pitches*, PIT).

El nivel de significación escogido para los siguientes análisis es de $\alpha = 0.05$, que será ajustado siguiendo la Fórmula (10) en base a la cantidad de test realizados, para contrarrestar el aumento en la probabilidad de encontrar errores de tipo I como explican Jafari y Ansari-Pour [13].

$$\alpha' = \frac{\alpha}{m}, \quad \alpha = 0.05, \quad m = número\ de\ tests \tag{10}$$

En primer lugar, deseamos comprobar si se produce algún cambio significativo a lo largo de los años al considerar partidos de distintas temporadas, lo cual impediría tratar los partidos con independencia de cuándo se disputaron. Para ello, estudiamos las tendencias en la media de bateo (*batting average*, BA) y las carreras anotadas (*runs scored*, R) respecto al tiempo, mostradas en las Figuras 7 y 8.

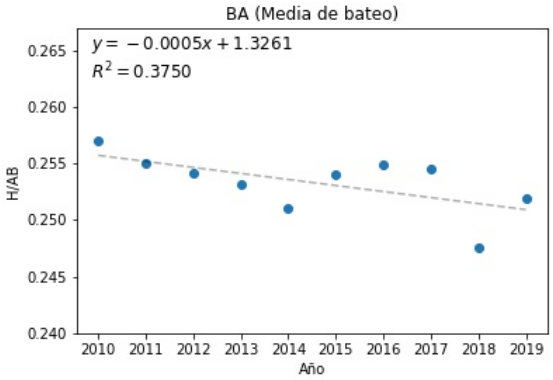

Figura 7: Evolución de la media de bateo

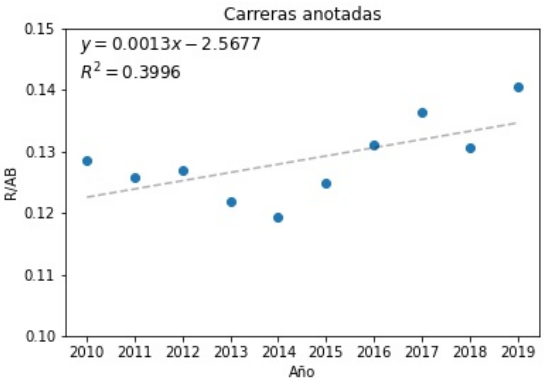

Figura 8: Evolución de la media de carreras anotadas

Las líneas de mejor ajuste parecen indicar una tendencia negativa en el caso de la media de bateo y positiva en el de las carreras anotadas.

Para evaluar estas hipótesis, primero consideramos las gráficas que muestran los residuos de las regresiones, mostradas en las Figuras 9 y 10. Si los modelos de regresión fueran aplicables a los datos, deberían mostrar una distribución uniformemente aleatoria centrada en 0.

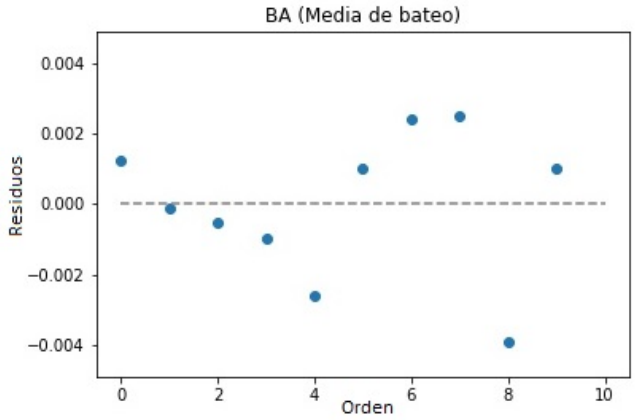

Figura 9: Residuos de la regresión lineal sobre la media de bateo

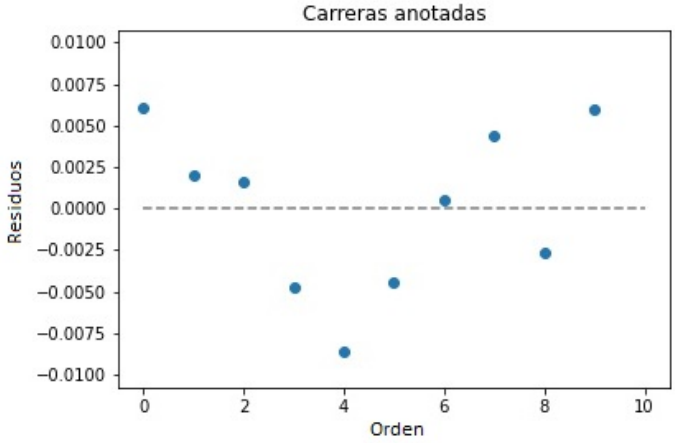

Figura 10: Residuos de la regresión lineal sobre la media de carreras anotadas

Observamos una tendencia similar en ambos casos, en los cuales el error fluctúa de manera no uniforme. Esto indica que los datos originales no siguen una tendencia lineal, lo cual podría ser explicado con funciones de orden polinomial mayor. Sin embargo, al tratarse de únicamente 10 observaciones, podría llevar al sobre-ajuste.

Nuestra hipótesis inicial, H_0, es que los datos no corresponden a una tendencia lineal. Para comprobar si lo opuesto es correcto, realizamos dos test *t* de Student mostrados en las Fórmulas (12) y (14).

$$H_0: y = \beta_0 + \beta_1 x + \varepsilon\,;\, \beta_1 = 0 \tag{11}$$

$$t_{BA} = \frac{\hat{\beta}_1 - \beta_1}{SE_{\hat{\beta}_1}} = \frac{\hat{\beta}_1 - 0}{SE_{\hat{\beta}_1}} \approx \frac{-0.0005}{0.00024} \approx -2.191 \tag{12}$$

$$P\left(T < -\left|t_{BA}\right| \cup T > \left|t_{BA}\right|\right) = 0.05983 \tag{13}$$

$$t_R = \frac{\hat{\beta}_1 - \beta_1}{SE_{\hat{\beta}_1}} = \frac{\hat{\beta}_1 - 0}{SE_{\hat{\beta}_1}} \approx \frac{0.0013}{0.00058} \approx 2.307 \tag{14}$$

$$P\left(T < -\left|t_R\right| \cup T > \left|t_R\right|\right) = 0.04989 \tag{15}$$

Ambas probabilidades obtenidas en las Fórmulas (13) y (15) son superiores a $\alpha' = \frac{\alpha}{2} = 0.025$, por lo que no podemos rechazar H_0.

Al comparar los mismos datos segregados por meses, sin embargo, comprobamos una tendencia cíclica en relación a la situación de la temporada, observable en la Figura 11.

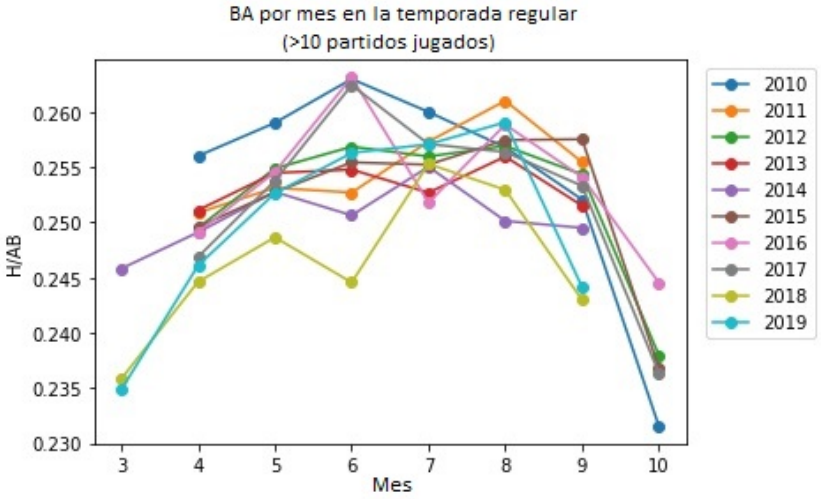

Figura 11: Tendencia cíclica en la media de bateo

La media de bateo es sistemáticamente menor en los primeros y últimos meses de cada temporada. Hipotetizamos que esto se debe a que en los últimos meses las clasificaciones de cada liga están matemáticamente resueltas en algunos casos. Los equipos que no tienen opción a modificar su puesto en la clasificación tienen un menor incentivo a la hora de jugar los partidos restantes. En cuanto a los primeros meses, diversos factores como la falta de entrenamiento o la adaptación a cambios en la plantilla pueden ser los causantes.

Por otra parte, consideramos la influencia que factores como la edad o la experiencia en la MLB pueden tener en las métricas principales de los jugadores.

Las Figuras 12 y 13 muestran su efecto en la media de bateo.

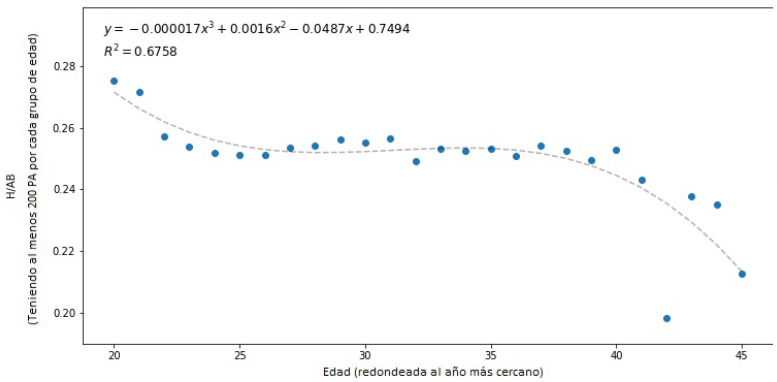

Figura 12: Influencia de la edad sobre la media de bateo

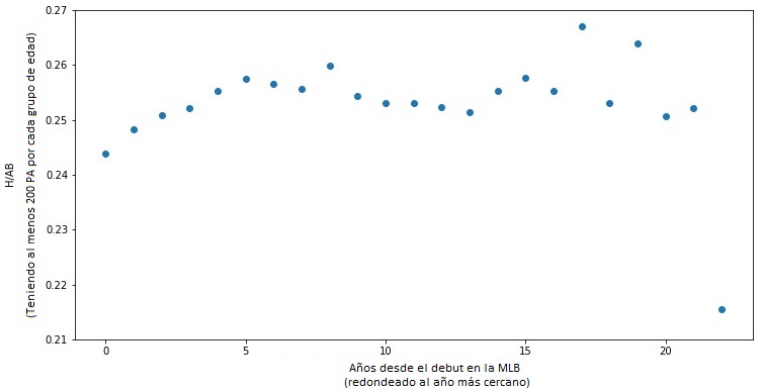

Figura 13: Influencia de la experiencia sobre la media de bateo

Estudios anteriores [14], [15] han demostrado que el rendimiento máximo entre atletas de élite decae aproximadamente a partir de los 25 años en ejercicios de *sprint*, pero no encontraron una relación significativa entre la edad y el tiempo de reacción [16], ambos factores determinantes a la hora de llegar de forma segura a las bases.

Los primeros 5 años de experiencia en la MLB incrementan de forma gradual el rendimiento.

Es posible que las muestras obtenidas no sean plenamente extrapolables a la carrera de jugadores concretos ya que es más probable que aquellos que tengan un peor rendimiento sean sustituidos de sus respectivos equipos.

El orden de bateo sigue una estrategia ampliamente extendida [17], [18], por lo que los jugadores cumplen funciones diferentes dependiendo de su posición en la alineación. Además, la cantidad de ocasiones de bateo sigue una función monótona decreciente, ya que es posible que los últimos jugadores no lleguen a batear en la última rotación de la alineación. Por tanto, la importancia de los porcentajes de bateo, *hits* para bases extra, y *home runs* es de mayor importancia entre los primeros bateadores (excepto para los jugadores en 1ª y 2ª posición, cuyo factor más relevante es la velocidad). Estas asunciones se ven reflejadas en los datos empíricos como muestran las Figuras 14 y 15.

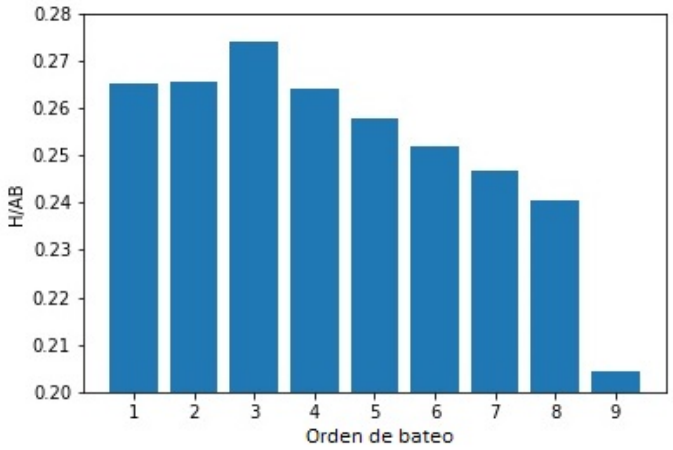

Figura 14: Influencia de la media de bateo en el orden de la alineación

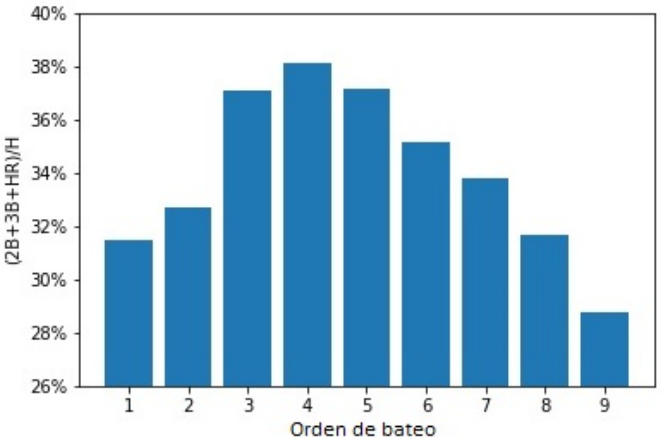

Figura 15: Influencia de la potencia en el orden de la alineación

Una de las estadísticas más comunes a la hora de evaluar la labor de los lanzadores es el ratio de *strikeouts* frente a bases por bolas, K:BB. La media de carreras limpias permitidas, ERA, es otra métrica común, pero involucra una mayor cantidad de jugadores a lo largo de varias jugadas, por lo que no es tan representativa del lanzador de forma exclusiva.

En las figuras 16 y 17 analizamos el poder predictivo que tiene el ratio K:BB de los lanzadores sobre el porcentaje de victorias obtenidas en el año siguiente en aquellos partidos en los que ha sido el lanzador durante al menos 5 entradas completas.

A diferencia de ERA, el ratio K:BB solo toma en cuenta al lanzador y al bateador, por lo que los posibles cambios en las plantillas ocurridos en el segundo año tienen una influencia menor.

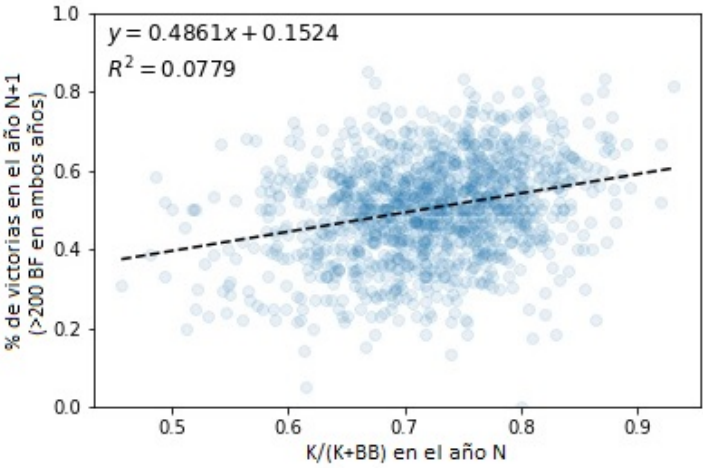

Figura 16: K:BB por lanzador como factor predictivo del número de victorias

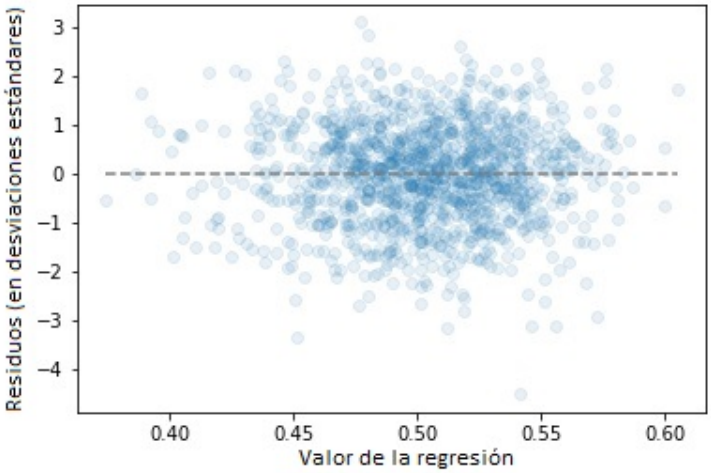

Figura 17: Residuos de la regresión lineal sobre K:BB

A pesar de existir una gran variabilidad no explicada por la línea de mejor ajuste (92.21%), la tendencia positiva es estadísticamente significativa (p=2.00^{-24}), por lo que concluimos que la cantidad de *strikeouts* y bases por bolas son relevantes a la hora de predecir futuros resultados.

La desviación en los residuos es uniforme independientemente del valor predicho, y el 68.22%, 95.82% y 99.53% de los residuos se encuentran dentro del rango de 1, 2 y 3 desviaciones estándar, respectivamente, cerca del 68.27%, 95.45% y 99.73% esperado de una distribución normal.

5. CONFIGURACIÓN EXPERIMENTAL

El objetivo principal de la experimentación es predecir el equipo ganador de los partidos, por lo que utilizamos en primer lugar los 9 modelos de clasificación tratados en la Sección 2.2. (*naïve Bayes*, k-NN, regresión logística, SVM, árboles de decisión, *random forests*, AdaBoost, XGBoost y redes neuronales).

Por otra parte, estudiamos también los modelos análogos a los mencionados en sus variantes de regresión para calcular la diferencia de carreras entre ambos equipos, la cual usamos después para obtener el equipo ganador, aquel que haya conseguido más carreras. El único modelo que no consta de alternativa para regresión es *naïve Bayes*, por lo que utilizamos los otros 8 modelos restantes.

En ambos casos, comprobamos el efecto que tienen las técnicas de selección de variables así como el uso de todas las variables disponibles en los resultados. Realizamos la selección únicamente para los modelos que disponen de alguna forma de ordenación de importancia de las variables, ya que en esos casos podemos eliminar iterativamente el atributo menos importante del conjunto.

Finalmente, utilizamos cuatro distintas configuraciones de variables, definidas en la Sección 5.1. para cada uno de los casos anteriormente mencionados. Las situaciones estudiadas quedan resumidas en la Tabla 2.

Tabla 2: Configuraciones de la experimentación

<table>
<tr><th>Modelo base</th><th>Tipo de predicción</th><th>Selección de variables</th><th>Configuración de variables</th></tr>
<tr><td>Naïve Bayes</td><td>Clasificación</td><td rowspan="2">Sin selección de variables</td><td rowspan="10">Últimos N partidos, últimos N enfrentamientos, temporada pasada, histórico</td></tr>
<tr><td>k-NN</td><td rowspan="9">Clasificación, regresión</td></tr>
<tr><td>Red neuronal</td></tr>
<tr><td>Regresión logística</td><td rowspan="7">Con selección de variables, sin selección de variables</td></tr>
<tr><td>SVM</td></tr>
<tr><td>Árbol de decisión</td></tr>
<tr><td>Random Forest</td></tr>
<tr><td>AdaBoost</td></tr>
<tr><td>XGBoost</td></tr>
</table>

En primer lugar, llevamos a cabo una selección de variables independiente para cada modelo, tipo de predicción, y conjunto de datos, con el fin de escoger las variables más relevantes para el problema y reducir el tiempo de ejecución. Esta selección se realiza mediante la eliminación recursiva de las variables menos relevantes, utilizando una validación cruzada de 10 iteraciones o *folds* para obtener el número óptimo de variables.

Por otra parte, los modelos empleados para la predicción constan de una serie de hiper-parámetros que pueden mejorar tanto el tiempo de ejecución como la eficacia de los métodos, consiguiendo unos resultados más generalizables al evitar el sobre-ajuste a los datos de entrenamiento.

Para ello, seleccionamos 100 configuraciones del conjunto de valores que pueden tomar todos los parámetros estudiados, escogiendo cada uno de ellos de una lista de posibles valores para los parámetros discretos y partiendo de distribuciones uniformes para los parámetros continuos. Se utiliza de nuevo una validación cruzada de 10 *folds* para obtener los valores óptimos de los hiper-parámetros.

Tras ello, se entrenan los modelos obtenidos y se registra su rendimiento en el conjunto de test. La división entre los conjuntos de entrenamiento y test es del 80% y 20% respectivamente, manteniendo el mismo equilibrio entre las clases para ambos conjuntos.

Finalmente, comparamos los resultados obtenidos de esta manera con los modelos básicos de predicción aleatoria y predicción constante del equipo local, así como con los resultados de otros estudios.

5.1. CONFIGURACIONES DE VARIABLES

Creamos cuatro distintas configuraciones de variables de entrada para los modelos, basándonos en diferentes periodos de tiempo y enfrentamientos entre rivales.

En todos los supuestos, las variables relativas al bateo y el *pitching* están normalizadas respecto al número de ABs o de BFs para evitar darle una importancia mayor a aquellos jugadores que simplemente hayan tenido más apariciones, tanto como para la mejora en los tiempos de convergencia de los algoritmos.

Las variables relativas a los bateadores son las siguientes: BA, SLG, RA, SOA, SBP y FEP.

Estas variables cubren los principales aspectos ofensivos y defensivos. Por una parte, BA toma en cuenta la capacidad de conseguir *hits* que avancen tanto al bateador como al resto de corredores. Mediante SLG se tiene en consideración la potencia de dichos *hits*, mientras que RA y SBP se centran en la habilidad de recorrer las bases. FEP, por otra parte, resume la información sobre los eventos negativos y positivos a la hora de defender.

Las variables específicas de los lanzadores son: PITA, KA, HAA, PE y ERA.

KA y HAA engloban dos de los desenlaces principales de los turnos al bate: las eliminaciones causadas únicamente por el *pitcher* y los *hits* permitidos. PE incluye otras formas en las que los bateadores logran avanzar por culpa de errores individuales del lanzador, mientras que ERA representa solamente aquellos corredores que logran anotar, ya que los corredores eliminados tras llegar a alguna base no alteran el resultado final del partido de forma directa. Finalmente, PITA puede servir como indicador del esfuerzo físico realizado por el lanzador.

Las cuatro configuraciones son las siguientes:

- **Últimos N partidos:** En primer lugar, utilizamos las variables que atañen a los jugadores, teniendo en cuenta únicamente las estadísticas acumuladas en los últimos N partidos en los que jugaron. De esta forma, analizamos la relevancia que los partidos más recientes tienen en el rendimiento de los jugadores.

  Por otra parte, incluimos el porcentaje de los partidos de la liga que ya han sido jugados, para tener en consideración la importancia del momento del año en el que se juega el partido.

  Añadimos también el factor campo o *park factor* del estadio en el que se disputa el partido.

- **Últimos N enfrentamientos:** Se mantienen las variables de jugadores, el porcentaje de liga completado y el *park factor*. Las variables de los jugadores, sin embargo, toman los datos únicamente de los N últimos partidos de la temporada en los que la misma pareja de equipos se han enfrentado.

  Además, se añade el porcentaje de esos últimos N enfrentamientos en los que el equipo que en el momento juega como local ha ganado.

  Ya que los mismos equipos se enfrentan en múltiples ocasiones por temporada, observamos si estos partidos son de mayor relevancia al tratarse de los mismos jugadores (exceptuando posibles cambios en la alineación durante la temporada).

- **Temporada pasada**: Se utilizan de nuevo las variables relativas a los jugadores, el porcentaje de liga completado, y el *park factor*. En este caso, las variables de los jugadores toman sus estadísticas durante la totalidad de la temporada pasada.

  Junto a ello, se incluye el porcentaje de partidos ganados por cada equipo en la temporada pasada.

- **Histórico**: Todas las variables son análogas a las de la configuración de la temporada pasada, salvo que se utilizan todos los datos previos disponibles (desde 2009 hasta el momento del partido) de los jugadores, en lugar de únicamente los de la última temporada al momento del partido. De esta forma, se tiene una visión más global de la carrera de cada jugador.

En los dos primeros casos, utilizamos 3, 5 y 10 como valores de N.

5.2. EVALUACIÓN

Los datos de 2010-2019 están ligeramente desequilibrados (53.54% victorias del equipo local y 46.46% del visitante), por lo que creamos una matriz de confusión para observar los errores divididos por las clases predichas y las reales, como se muestra en la 3.

Tabla 3: Matriz de confusión

		Clase predicha	
		1, victoria local	0, victoria visitante
Clase actual	1, victoria local	TP	FN
	0, victoria visitante	FP	TN

Accuracy indica el número total de casos en los que la clase predicha es la correcta, calculado mediante la Fórmula (16).

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{16}$$

Esta medida puede tomar valores dentro del rango [0, 1], siendo 1 los casos en los que todas la predicciones son correctas.

Sin embargo, debido al desequilibrio entre ambas clases, utilizar únicamente *accuracy* puede llevar a conclusiones incorrectas, ya que los modelos que predicen la clase mayoritaria con mayor frecuencia tienden a obtener valoraciones superiores sin ofrecer tanta información respecto al relevancia que tienen las variables de entrada en cuanto a la predicción.

Dos métricas ampliamente utilizadas son *precision* (Fórmula (17)) y *recall* (Fórmula (18)).

$$precision = \frac{TP}{TP + FP} \tag{17}$$

$$recall = \frac{TP}{TP + FN} \tag{18}$$

Precision mide cuántos de los casos predichos como “clase 1” son correctos, mientras que *recall* muestra cuántas instancias de la clase positiva se han predicho como tal.

Es común emplear ambas medidas en conjunción para obtener una visión más general sobre el modelo predictivo, siendo *F-score* una de las opciones más empleadas.

La medida F_1 se calcula como expresa la Fórmula (19), mediante la media armónica entre *precision* y *recall*.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \quad (19)$$

Sin embargo, como muestra la Fórmula (19), no tiene en consideración el número de clasificaciones correctas de la clase negativa, TN. Esto puede ser un factor positivo en aquellos problemas en los que la clase negativa está muy sobrerrepresentada. Sin embargo, la distribución solo está ligeramente desequilibrada en nuestro caso, por lo que consideramos métricas que tomen en cuenta este factor.

Por otra parte, la curva ROC (*receiver operating characteristic*) representa la forma en la que el modelo varía sus predicciones al modificar el umbral de decisión. De manera habitual, se clasifican como una clase u otra aquellos ejemplos en los que la probabilidad de pertenencia a dicha clase sea igual o superior al 50%, pero alterar el punto de corte puede ser útil si se desea clasificar correctamente una mayor cantidad de instancias de una de las clases, a pesar de ir en detrimento de la otra.

Por ejemplo, si decidimos clasificar como la clase positiva aquellos datos con los que se obtenga una probabilidad superior al 10%, seleccionaremos una mayor parte de los ejemplos positivos, aunque al mismo tiempo se aumentará la cantidad de casos negativos erróneamente clasificados.

Para ello, se representan el *true positive rate* (*TPR*, Fórmula (20)) y el *false positive rate* (*FPR*, Fórmula (21)) en una gráfica como la mostrada en la Figura 18.

$$TPR = \frac{TP}{TP + FN} \quad (20)$$

$$FPR = \frac{FP}{FP + TN} \tag{21}$$

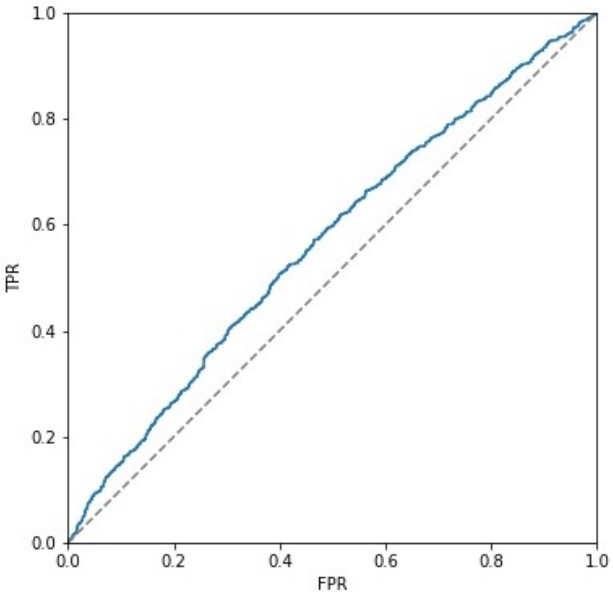

Figura 18: Ejemplo de la curva ROC

TPR indica la cantidad de elementos de la clase positiva han sido clasificados como tal, mientras que FPR muestra cuántos ejemplos negativos han sido clasificados erróneamente. De esta forma, puede observarse el beneficio de ser más laxo a la hora de predecir ejemplos como positivos frente al error inducido al clasificar también ejemplos negativos de forma incorrecta.

Como puede comprobarse en las Fórmulas (20) y (21), la curva ROC incluye todos los tipos de errores y aciertos, ya que abarca los cuatro casos posibles, TP, FP, TN y FN.

Para poder comparar los resultados de distintos modelos, ya que es posible que sus curvas ROC no sean superiores o inferiores en todos los intervalos, se utiliza el área bajo la curva o AUC. Ya que se desea un mayor TPR, aquellos modelos que obtengan valores superiores del AUC son considerados mejores.

Por tanto, evaluamos los modelos entrenados comprobando su *accuracy* y el área bajo la curva ROC.

Comparamos los resultados obtenidos de esta manera con los modelos triviales de referencia que predicen el equipo ganador aleatoriamente o siempre predicen como ganador al equipo local, así como con los resultados de estudios similares.

6. RESULTADOS Y ANÁLISIS

6.1. SELECCIÓN DE VARIABLES

El primer aspecto de la experimentación, como se define en la Sección 5., es la selección de variables. Los conjuntos de variables seleccionadas para cada modelo están incluidos en el Anexo ii. El prefijo "*L_*" o "*V_*" indica si hacen referencia al equipo local o visitante, respectivamente. Se usan también prefijos para identificar la posición del jugador.

Cabe mencionar que las variables escogidas con mayor frecuencia son las relativas a los *pitchers*, en especial KA y PE. Los jugadores con el mayor impacto en el partido son los *pitchers* de cada equipo, ya que toman parte en todas las jugadas defensivas, que comienzan con sus lanzamientos.

Dentro de las posibles estadísticas exclusivas de los *pitchers*, destacan las métricas que lo involucran de forma relativamente aislada: los errores de HB y BK incluidos en PE son causados únicamente por él, y ningún otro jugador defensivo participa a la hora de conseguir un *strikeout*. Por el contrario, para que al *pitcher* se le asigne un HA, el bateador debe conseguir un contacto válido y alcanzar la primera base antes de que el resto de la defensa logre eliminarlo.

A pesar de que la estadística ERA sea una de las más utilizadas a la hora de evaluar el rendimiento de los lanzadores, incluye más factores que otras métricas básicas, ya que para anotar las carreras es posible que los corredores avancen a lo largo de múltiples ABs necesitando que sucesivos bateadores los asistan, consigan robar bases… Algunos de estos factores están fuera del control del *pitcher*, por lo que esta variable no resume su participación de forma independiente.

En cuanto al resto de las estadísticas de los jugadores, FEP y SBP son las menos destacadas. La cantidad de errores defensivos por partido es muy baja (inferior a 1 de media) en comparación con el número de asistencias y eliminaciones, y solo los fallos flagrantes son considerados como errores de forma oficial, por lo que esta métrica no plasma correctamente la eficacia defensiva de los jugadores. De forma similar, las bases robadas son infrecuentes y requieren que el jugador haya llegado a la primera base previamente.

Por otra parte, el PF es la variable de menor importancia entre las no relativas a los jugadores individuales, aunque resulta seleccionada en algunos casos. Muchos estadios tienen un factor cercano a 1.0, aunque puede ser una métrica relevante para otros partidos.

El porcentaje de partidos de la temporada ya disputados es seleccionado por múltiples modelos basados en árboles, pero obtiene peores posiciones en la regresión logística y lineal y en SVM. Esto puede deberse a que por sí mismo el dato no otorga mucha información, ya que, por ejemplo, los equipos que todavía pueden alcanzar puestos de campeonato siguen teniendo un claro incentivo a pesar de encontrarse cerca del final de la temporada.

Finalmente, el porcentaje de victorias de los equipos durante la temporada pasada se utiliza en la gran mayoría de casos que disponen de esta variable. Aunque los clubs sufran algunos cambios de un año al siguiente, los equipos que obtienen mejores resultados suelen hacerlo de forma continuada.

6.2. SELECCIÓN DE HIPERPARÁMETROS

Las implementaciones de *sklearn* de los modelos utilizados disponen de una serie de hiperparámetros que cumplen diferentes funciones como la regularización, técnicas para evitar el *overfitting* como la poda de árboles, el establecimiento de criterios de parada, la paralelización de la ejecución, etc.

En el caso de ***naïve Bayes***, se tienen dos parámetros: *priors* y *var_smoothing*. El primero sirve para establecer la probabilidad previa de cada clase, pero lo omitimos para que se calcule ajustándose a los datos proporcionados.

El segundo parámetro se utiliza para estabilizar los cálculos, ya que datos con una varianza demasiado baja pueden causar errores numéricos. El valor se multiplica por la mayor de las varianzas y se añade al resto de varianzas. Tras probar valores en el rango de [$1x10^{-11}$, $1x10^{-7}$], la mayoría toman valores del orden de magnitud de 10^{-8}.

Para las versiones de clasificación y de regresión de **k-NN**, uno de los parámetros principales es el número de vecinos que se toman en cuenta a la hora de realizar las predicciones. Utilizamos como valores posibles los enteros del 1 al 10.

Por otra parte, se consideran dos formas de dar pesos a cada vecino: uniforme (todos los puntos tienen el mismo peso) y basado en la distancia (los puntos más cercanos tienen una mayor influencia en la predicción).

En la mayoría de los casos, se obtienen mejores resultados usando el peso basado en las distancias, con el que los ejemplos más parecidos al *input* a predecir obtienen una mayor relevancia.

Para este y otros modelos siguientes, el parámetro *n_jobs* se fija a “-1” para hacer uso de todas las unidades de procesamiento disponibles y acelerar así el tiempo de ejecución.

Las implementaciones de la **regresión logística y lineal** permiten aplicar regularización para reducir la complejidad de los modelos, logrando así que sean más generalizables al no adaptarse completamente a la variabilidad o ruido que pueden contener los datos de entrenamiento. En el caso de la regresión lineal, la intensidad de la regularización se calibra con el parámetro *alpha*, para el cual probamos los valores en (0, 5]. En la regresión logística, de forma similar, el parámetro *C* cumple la misma función, salvo que indica el inverso de la intensidad por lo que valores menores implican una mayor regularización. Usamos valores de *C* pertenecientes al intervalo (0, 1.5].

A diferencia del valor por defecto de 500, establecemos un límite máximo de 10,000 iteraciones hasta llegar a la convergencia.

Los algoritmos utilizados para el problema de minimización de la función de coste son *lbfgs*, *liblinear* y *saga* en el caso de clasificación y *lsqr*, *sparse_cg* y *saga* para la regresión.

El algoritmo de L-BFGS es una aproximación a los métodos newtonianos que necesita una menor cantidad de memoria. *Liblinear* basa la minimización de la función de coste multivariable en una serie de funciones de una única variable, lo cual permite su ejecución más rápida en paralelo. *Saga* es una versión de descenso por gradiente que utiliza un subconjunto de los gradientes en cada iteración, haciéndolo más adecuado a problemas con una gran cantidad de variables. Por otra parte, *lsqr* busca los mínimos de los errores al cuadrado sobre un sistema de ecuaciones lineales de forma iterativa, mientras que *sparse_cg* utiliza el método del gradiente conjugado.

Los hiperparámetros óptimos varían para cada modelo, conjunto de variables y tipo de problema a resolver.

Debido a limitaciones en el tiempo de ejecución y teniendo en cuenta la alta dimensionalidad de los conjuntos de datos, utilizamos únicamente el *kernel* lineal para **SVC** y **SVR**, un número máximo de 1,000 iteraciones y el factor de regularización *C* en el rango [0.5, 1.5].

Los **árboles de decisión** pueden ser propensos a ajustarse demasiado a los datos de entrenamiento. Para evitarlo, establecemos la profundidad máxima que puede alcanzarse entre 3 y 10 niveles, no permitiendo así árboles que dividan los nodos hasta llegar a soluciones demasiado complejas y no generalizables. La mayoría de modelos óptimos toman valores intermedios.

Además de ello, se hace una poda del árbol resultante, eligiendo el sub-árbol de mayor complejidad que no supere un valor *ccp_alpha* seleccionado entre el posible rango [0, 0.05].

Para calcular la calidad de las divisiones, se utilizan los criterios basados en la impureza de Gini (mide cuántos casos son clasificados erróneamente) o en la ganancia de información (inversa a la entropía, basada en la cantidad de ejemplos de cada clase en un nodo dado) para la clasificación y el error cuadrático medio con y sin la mejora de Friedman [19] para la regresión.

En los modelos de ***random forest***, de forma similar, probamos a limitar la profundidad máxima de cada árbol entre 1 y 10 niveles, el factor de complejidad máximo *ccp_alpha* y los criterios de valoración de las divisiones (a excepción del error cuadrático medio de Friedman, ya que no se ofrece como posibilidad en la implementación de *sklearn*). El número de árboles de los que se componen son entre 20 y 150 en nuestra experimentación. Además, a la hora de buscar la mejor división de un nodo, limitamos la búsqueda a un conjunto aleatorio con cardinalidad igual a la raíz cuadrada de las variables disponibles tanto como a la totalidad. De esta forma, se da la oportunidad a variables que no necesariamente obtienen el mejor resultado a ser seleccionadas.

Gran parte de los modelos escogidos llevan a cabo esta limitación, la cual ayuda a obtener resultados más generalizables al otorgarle una mayor flexibilidad respecto a datos diversos de entrada. La cantidad de árboles base que conforman los modelos oscila entorno a los 100.

A la hora de generar las **redes neuronales**, se deben escoger el número de capas ocultas (1, 2 o 3 en nuestro caso) y la cantidad de neuronas por cada capa (de 20 a 100 para las redes de una capa, de 20 a 50 para las de dos y de 10 a 33 para las de tres). El parámetro *alpha* controla el factor de regularización, tomando distintos valores en [0, 1]. Pueden utilizarse las funciones de activación logística, tangente hiperbólica y ReLU (Fórmulas (6), (8) y (9) de las Secciones 2.2.3. y 2.2.9., respectivamente). La función para resolver la optimización de los pesos puede ser la anteriormente mencionada L-BFGS o el algoritmo *Adam* [20] basado en el descenso por gradiente.

Los modelos de regresión obtienen en la mayoría de los casos mejores resultados con redes neuronales de más capas en comparación a los análogos de clasificación.

En lo relativo a **AdaBoost**, usamos árboles de decisión de profundidad 1 como modelo base, de los cuales puede haber entre 20 y 500. Comprobamos también distintos factores de aprendizaje del rango [0.0001, 1].

Para **XGBoost**, además del mismo rango del factor de aprendizaje, utilizamos distintos valores para el término de regularización de los pesos, pudiendo estar entre 0.5 y 1.5. A diferencia de la profundidad máxima por defecto de 6, también se establece como un hiperparámetro a seleccionar, con valores entre 1 y 10.

El valor seleccionado más común para la profundidad es de 1, caso en el que cada uno de los árboles son de los modelos más simples, pero que contribuyen de tal manera que el resultado final es un modelo robusto.

6.3. CONFIGURACIONES DE VARIABLES

A continuación se muestran los resultados obtenidos sobre el conjunto de test con los mejores hiperparámetros seleccionados mediante la validación cruzada mencionada anteriormente. Se incluyen también los datos experimentales tras llevar a cabo la selección de variables en aquellos modelos que lo permiten.

6.3.1. ÚLTIMOS N PARTIDOS

La Tabla 4 muestra el rendimiento de los modelos con los mejores hiperparámetros hallados mediante validación cruzada, los cuales son elegidos al obtener el mejor AUC entre los 100 posibles modelos evaluados. Sin embargo, es posible que estos valores sobrestimen el resultado real, ya que se han escogido únicamente los mejores candidatos. Por tanto, entrenamos los modelos con el conjunto total de datos de entrenamiento y evaluamos su rendimiento respecto al conjunto de test no conocido previamente.

Tabla 4: Resultados de clasificación con datos de los últimos 3 partidos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	120	0.5146	0.5121	0.5051	k=2, pesos=distancia
Regresión Logística	120	0.5376	0.5453	0.5276	C=1.246, f=lbfgs
	8	0.5383	0.5381	**0.5459**	C=0.5655, f=liblinear
SVC	120	0.5321	**0.5488**	0.5269	C=0.8229
	8	0.5208	0.5283	0.5371	C=0.9965
Decision Tree	120	0.5394	0.5225	0.5209	ccp_alpha=0.001013, profundidad=10, criterio=entropía
	6	0.5179	0.5037	0.5097	ccp_alpha=0.0004850, profundidad=3, criterio=gini
Random Forest	120	0.5362	0.5270	0.5323	ccp_alpha=0.003137, profundidad=7, max_variables=no, árboles=137, criterio=entropía
	68	0.5362	0.5309	0.5339	ccp_alpha=0.001873, profundidad=7, max_variables=sqrt, árboles=98, criterio=entropía
Red Neuronal	120	0.5369	0.5467	0.5246	activación=tanh, alpha=0.08340, nodos=[39], f=adam
Naive Bayes	120	0.5099	0.5222	0.5211	var_smoothing=6.970e-09
AdaBoost	120	0.5435	0.5427	0.5322	aprendizaje=0.2161, árboles=71
	5	0.5336	0.5292	0.5385	aprendizaje=0.5306, árboles=59
XGBoost	120	0.5427	0.5459	0.5319	aprendizaje=0.1289, profundidad=1, reg_lambda=0.9328
	92	**0.5467**	0.5459	0.5358	aprendizaje=0.3302, profundidad=1, reg_lambda=0.5655
Aleatorio	0	0.5000			
Local	0	0.5362			

El 53.62% de los partidos del conjunto de test corresponden a victorias del equipo local, por lo que el modelo trivial que predice la clase mayoritaria en todos los casos obtiene un *accuracy* de 0.5362. Podemos observar que la mayoría de modelos no superan al trivial, en parte debido a los peores resultados obtenidos al utilizar un subconjunto considerablemente menor de las variables posibles.

Los aumentos en el AUC respecto a los obtenidos en la afinación de hiperparámetros pueden deberse a la utilización de el conjunto total de entrenamiento al no tener la necesidad de dividirlo en *folds*, de los cuales 1 de 10 se usa para su validación.

Al realizar las clasificaciones basándonos en modelos que predicen la diferencia de carreras anotadas, observamos en la Tabla 5 un empeoramiento de los resultados en la media del AUC, a pesar de que algunos modelos sufran ligeras mejorías.

Tabla 5: Resultados de regresión con datos de los últimos 3 partidos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	120	0.5146	0.5151	0.5043	k=1, pesos=uniforme
Regresión Lineal	120	0.5416	**0.5523**	0.5327	alpha=0.4201, f=lsqr
	3	0.5376	0.5325	**0.5417**	alpha=3.484, f=lsqr
SVR	120	0.5325	0.5272	0.5269	C=1.498
	40	0.5376	0.5385	0.5363	C=1.003
Decision Tree	120	0.5336	0.5246	0.5172	ccp_alpha=0.02495, profundidad=7, criterio=mse
	1	0.5212	0.5081	0.5116	ccp_alpha=0.008650, profundidad=3, criterio=mse
Random Forest	120	**0.5500**	0.5519	0.5396	ccp_alpha=0.03692, profundidad=6, max_variables=sqrt, árboles=63
	57	0.5486	0.5453	0.5410	ccp_alpha=0.004407, profundidad=6, max_variables=sqrt, árboles=98
Red Neuronal	120	0.5190	0.5146	0.5249	activación=relu, alpha=0.4041, nodos=[39, 39], f=lbfgs
AdaBoost	120	0.5351	0.5423	0.5358	aprendizaje=0.01143, árboles=225
	5	0.5281	0.5281	0.5402	aprendizaje=0.8575, árboles=40
XGBoost	120	0.5431	0.5437	0.5335	aprendizaje=0.1113, profundidad=1, reg_lambda=0.5232
	1	0.5343	0.5033	0.5110	aprendizaje=0.2916, profundidad=1, reg_lambda=0.8879
Aleatorio	0	0.5000			
Local	0	0.5362			

Cabe destacar que los modelos de regresión basados tanto en *decision trees* como en XGBoost seleccionan una única variable en todos los casos estudiados.

Los datos de la Tabla 6 muestran como al incrementar el número de partidos sobre los cuales se calculan las estadísticas de los jugadores, conseguimos un aumento en el AUC de la mayoría de modelos, pasando de una media de 0.5326 a 0.5404, traduciéndose también en un ligero incremento en el *accuracy* de 0.5324 a 0.5359, más cercano al modelo trivial.

Tabla 6: Resultados de clasificación con datos de los últimos 5 partidos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	120	0.5102	0.5017	0.5071	k=8, pesos=distancia
Regresión Logística	120	0.5416	0.5477	0.5288	C=1.113, f=liblinear
	4	0.5303	0.5411	0.5499	C=0.5225, f=saga
SVC	120	0.5362	0.5481	0.5275	C=1.070
	65	**0.5471**	0.5508	0.5485	C=0.9516
Decision Tree	120	0.5351	0.5315	0.5231	ccp_alpha=0.0008502, profundidad=6, criterio=gini
	39	0.5351	0.5313	0.5259	ccp_alpha=0.0007237, profundidad=5, criterio=gini
Random Forest	120	0.5365	0.5483	0.5390	ccp_alpha=0.0002457, profundidad=3, max_variables=sqrt, árboles=88, criterio=gini
	61	0.5369	0.5463	0.5395	ccp_alpha=0.0005390, profundidad=4, max_variables=sqrt, árboles=96, criterio=entropía
Red Neuronal	120	0.5391	0.5477	0.5293	activación=logística, alpha=0.6419, nodos=[54]
Naive Bayes	120	0.5292	0.5376	0.5166	var_smoothing=7.144e-08
AdaBoost	120	0.5431	**0.5537**	0.5491	aprendizaje=0.2905, árboles=45
	30	0.5394	0.5429	**0.5629**	aprendizaje=0.9987, árboles=58
XGBoost	120	0.5431	0.5513	0.5471	aprendizaje=0.1551, profundidad=1, reg_lambda=0.5084
	11	0.5358	0.5257	0.5395	aprendizaje=0.08681, profundidad=3, reg_lambda=1.321
Aleatorio	0	0.5000			
Local	0	0.5362			

Los resultados obtenidos mediante k-NN son los más bajos, habiendo incluso empeorado al incrementar el número de partidos considerados.

A pesar de ello, 6 de los 9 modelos obtienen resultados iguales o superiores al modelo de referencia cuando se usan todas las variables.

Como observamos en la Tabla 7, de nuevo, la tendencia general al utilizar modelos de regresión es negativa respecto al mismo conjunto de datos de entrada en clasificación, a pesar de que algunos modelos como k-NN o *random forest* mejoren sus resultados.

Tabla 7: Resultados de regresión con datos de los últimos 5 partidos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	120	0.5131	0.5109	0.5104	k=1, pesos=distancia
Regresión Lineal	120	0.5446	0.5508	0.5384	alpha=0.6395, f=lsqr
	5	0.5380	0.5452	**0.5512**	alpha=2.683, f=sparse_cg
SVR	120	0.5248	0.5230	0.5317	C=1.269
	37	0.5358	0.5299	0.5486	C=1.135
Decision Tree	120	0.5248	0.5175	0.5288	ccp_alpha=0.0003133, profundidad=5, criterio=mse
	1	0.5347	0.5217	0.5017	ccp_alpha=0.01216, profundidad=7, criterio=mse
Random Forest	120	0.5449	**0.5588**	0.5496	ccp_alpha=0.03817, profundidad=7, max_variables=sqrt, árboles=105
	47	**0.5500**	0.5496	0.5494	ccp_alpha=0.01412, profundidad=3, max_variables=sqrt, árboles=74
Red Neuronal	120	0.5292	0.5374	0.5310	activación=relu, alpha=0.7932, nodos=[13, 13, 13], f=lbfgs
AdaBoost	120	0.5325	0.5385	0.5478	aprendizaje=0.04303, árboles=90
	100	0.5369	0.5380	0.5472	aprendizaje=0.01125, árboles=384
XGBoost	120	0.5369	0.5431	0.5503	aprendizaje=0.1851, profundidad=1, reg_lambda=0.9485
	1	0.5340	0.4965	0.5052	aprendizaje=0.04797, profundidad=2, reg_lambda=0.5474
Aleatorio	0	0.5000			
Local	0	0.5362			

De la misma forma que comprobamos al pasar de N=3 a N=5 últimos partidos, al utilizar una cantidad todavía mayor se obtienen mejores resultados, reflejados en la Tabla 8.

Tabla 8: Resultados de clasificación con datos de los últimos 10 partidos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	120	0.5161	0.5095	0.5156	k=2, pesos=distancia
Regresión Logística	120	0.5380	0.5493	0.5424	C=1.189, f=liblinear
	2	0.5427	0.5460	0.5530	C=0.9781, f=saga
SVC	120	0.5420	0.5468	0.5402	C=1.005
	80	0.5365	0.5491	**0.5543**	C=0.8734
Decision Tree	120	0.5270	0.5122	0.5292	ccp_alpha=5.099, profundidad=6, criterio=entropía
	33	0.5332	0.5270	0.5328	ccp_alpha=0.0002963, profundidad=4, criterio=gini
Random Forest	120	0.5362	0.5440	0.5494	ccp_alpha=0.001779, profundidad=2, max_variables=sqrt, árboles=137, criterio=gini
	42	**0.5552**	0.5589	0.5514	ccp_alpha=0.0002931, profundidad=5, max_variables=sqrt, árboles=90, criterio=gini
Red Neuronal	120	0.5416	0.5488	0.5428	activación=logística, alpha=0.4192, nodos=[84], f=lbfgs
Naive Bayes	120	0.5446	0.5573	0.5285	var_smoothing=9.473e-08
AdaBoost	120	0.5504	**0.5604**	0.5434	aprendizaje=0.2548, árboles=98
	1	0.5310	0.5201	0.5249	aprendizaje=0.01173, árboles=477
XGBoost	120	0.5471	0.5484	0.5480	aprendizaje=0.01737, profundidad=1, reg_lambda=0.7438
	44	0.5453	0.5556	0.5529	aprendizaje=0.03875, profundidad=4, reg_lambda=0.5505
Aleatorio	0	0.5000			
Local	0	0.5362			

Esto puede deberse a que al obtener las estadísticas de los jugadores utilizando solamente las últimas 3 observaciones no se logra una muestra representativa de sus cualidades. Al incrementar los partidos considerados, se reduce la posible variabilidad de partidos extraordinarios y las métricas regresan hacia la media esperada.

Como podemos observar en la Tabla 9, la regresión lineal (de la misma forma que la regresión logística en clasificación) obtiene mejores resultados al seleccionar únicamente 7 de las 120 variables más relevantes.

Tabla 9: Resultados de regresión con datos de los últimos 10 partidos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	120	0.5095	0.5018	0.5143	k=4, pesos=distancia
Regresión Lineal	120	0.5467	0.5503	0.5466	alpha=1.498, f=sparse_cg
	7	**0.5544**	**0.5564**	0.5579	alpha=3.708, f=lsqr
SVR	120	0.5208	0.5151	0.5284	C=0.9606
	31	0.5369	0.5445	**0.5582**	C=1.342
Decision Tree	120	0.5230	0.5296	0.5389	ccp_alpha=0.03471, profundidad=4, criterio=mse
	1	0.5318	0.5249	0.5330	ccp_alpha=0.009602, profundidad=5, criterio=mse
Random Forest	120	0.5541	0.5556	0.5557	ccp_alpha=0.03441, profundidad=6, max_variables=sqrt, árboles=117
	94	0.5340	0.5475	0.5545	ccp_alpha=0.02818, profundidad=4, max_variables=no, árboles=98
Red Neuronal	120	0.5146	0.5157	0.5384	activación=relu, alpha=0.8793, nodos=[17, 17, 17], f=lbfgs
AdaBoost	120	0.5336	0.5508	0.5541	aprendizaje=0.01337, árboles=392
	51	0.5340	0.5432	0.5506	aprendizaje=0.01629, árboles=229
XGBoost	120	0.5475	0.5531	0.5512	aprendizaje=0.05008, profundidad=1, reg_lambda=0.5433
	1	0.5325	0.4957	0.5073	aprendizaje=0.1976, profundidad=4, reg_lambda=0.6316
Aleatorio	0	0.5000			
Local	0	0.5362			

6.3.2. ÚLTIMOS N ENFRENTAMIENTOS

En las Tablas 10-15 observamos tendencias similares a las explicadas anteriormente: los resultados obtenidos mediante los modelos de clasificación son, en media, ligeramente superiores a los de regresión.

El conjunto de partidos disponibles para el entrenamiento y test es menor al utilizado en el resto de configuraciones ya que se requiere que ambos equipos se hayan enfrentado en 2, 4 o 9 ocasiones previas durante la temporada. Esto reduce el número de partidos disponibles al 64%, 45% y 21% del total para N=3, 5 y 10, respectivamente.

Es por ello que, a diferencia de los casos anteriores, el rendimiento empeora con N=10 debido a la disminución en la cantidad de datos disponibles.

Tabla 10: Resultados de clasificación con datos de los últimos 3 enfrentamientos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	121	0.5171	0.5003	0.5296	k=5, pesos=distancia
Regresión Logística	121	0.5412	0.5186	0.5044	C=0.004200, f=liblinear
	4	0.5400	0.4867	0.5282	C=0.2242, f=saga
SVC	121	0.5165	0.4978	0.5127	C=0.7804
	1	0.5102	0.4952	0.5106	C=0.7520
Decision Tree	121	0.5406	0.5108	0.5066	ccp_alpha=0.001386, profundidad=4, criterio=entropía
	24	0.5140	0.5001	0.4958	ccp_alpha=0.0002939, profundidad=9, criterio=entropía
Random Forest	121	0.5412	0.5167	0.5211	ccp_alpha=0.001017, profundidad=2, max_variables=None, árboles=84, criterio=gini
	89	**0.5457**	0.5105	0.5231	ccp_alpha=0.0006218, profundidad=7, max_variables=sqrt, árboles=105, criterio=gini
Red Neuronal	121	0.5260	**0.5203**	0.5256	activación=tanh, alpha=0.04621, nodos=[69], f=lbfgs
Naive Bayes	121	0.4956	0.4989	0.5059	var_smoothing=6.414e-08
AdaBoost	121	0.5286	0.5159	0.5293	aprendizaje=0.6004, árboles=83
	1	0.5355	0.5082	0.5138	aprendizaje=0.8427, árboles=478
XGBoost	121	0.5273	0.5167	**0.5331**	aprendizaje=0.1291, profundidad=3, reg_lambda=0.6934
	1	0.5203	0.5045	0.5261	aprendizaje=0.7935, profundidad=7, reg_lambda=1.018
Aleatorio	0	0.5000			
Local	0	0.5425			

A pesar de que el *accuracy* se ve incrementado en 7 de los 15 modelos respecto a los análogos que utilizan los 3 últimos partidos (Tabla 4), el modelo de referencia contra el que se comparan ha aumentado su precisión en 0.64 puntos porcentuales (1.00 y 0.84 para los casos de N=5 y 10, respectivamente).

La mayoría de los modelos basados en esta configuración de variables no supera el umbral base de la predicción de la clase mayoritaria independientemente de la cantidad de partidos previos utilizados.

En la Tabla 11 se muestra la experimentación de los modelos de regresión, en los que la configuración de variables basada en los últimos 3 enfrentamientos entre ambos equipos otorga peores resultados en comparación con la configuración de 3 últimos partidos.

Tabla 11: Resultados de regresión con datos de los últimos 3 enfrentamientos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	121	0.5070	0.4966	0.5202	k=5, pesos=distancia
Regresión Lineal	121	0.5121	0.4961	0.5089	alpha=4.824, f=lsqr
	2	0.5247	0.4870	0.5246	alpha=0.3322, f=sparse_cg
SVR	121	0.4714	0.4971	0.5191	C=1.091
	36	0.4543	0.4889	**0.5346**	C=0.8039
Decision Tree	121	0.5349	0.4898	0.5210	ccp_alpha=0.03289, profundidad=3, criterio=mse
	1	0.5381	0.4927	0.5029	ccp_alpha=0.002335, profundidad=5, criterio=mse
Random Forest	121	0.5076	0.5126	0.5298	ccp_alpha=0.02806, profundidad=10, max_variables=sqrt, árboles=132
	119	0.5324	**0.5165**	0.5292	ccp_alpha=0.02300, profundidad=7, max_variables=sqrt, árboles=78
Red Neuronal	121	0.4892	0.5057	0.5216	activación=logística, alpha=0.7126, nodos=[29, 29, 29], f=lbfgs
AdaBoost	121	0.5343	0.5018	0.5227	aprendizaje=0.03138, árboles=119
	101	**0.5393**	0.5064	0.5217	aprendizaje=0.2805, árboles=25
XGBoost	121	0.5032	0.4972	0.5278	aprendizaje=0.6236, profundidad=10, reg_lambda=0.6856
	1	0.5209	0.4854	0.4964	aprendizaje=0.9557, profundidad=1, reg_lambda=0.7316
Aleatorio	0	0.5000			
Local	0	0.5425			

En las Tablas 12 y 13 puede observarse un ligero incremento al utilizar los últimos 5 enfrentamientos en lugar de 3, aunque siguen siendo inferiores a los casos análogos en los que se usan los 5 últimos partidos de cada jugador.

Tabla 12: Resultados de clasificación con datos de los últimos 5 enfrentamientos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	121	0.5254	0.5311	0.5166	k=10, pesos=uniforme
Regresión Logística	121	0.5453	0.5274	0.5183	C=0.0067, f=lbfgs
	16	0.5380	0.5309	0.5482	C=0.3600, f=liblinear
SVC	121	0.5399	0.4575	0.5203	C=0.9697
	16	0.5335	**0.5359**	**0.5491**	C=0.9072
Decision Tree	121	0.5127	0.4978	0.5152	ccp_alpha=0.001383, profundidad=7, criterio=entropía
	23	0.5226	0.4988	0.5287	ccp_alpha=0.001812, profundidad=6, criterio=entropía
Random Forest	121	0.5217	0.4991	0.5312	ccp_alpha=0.003052, profundidad=5, max_variables=no, árboles=79, criterio=entropía
	82	**0.5498**	0.5065	0.5304	ccp_alpha=0.004162, profundidad=8, max_variables=no, árboles=84, criterio=entropía
Red Neuronal	121	0.5462	0.5041	0.5265	activación=logística, alpha=0.9125, nodos=[22, 22], f=adam
Naive Bayes	121	0.5389	0.5334	0.4911	var_smoothing=3.126e-08
AdaBoost	121	0.5353	0.4980	0.5207	aprendizaje=0.2243, árboles=22
	3	0.5444	0.5271	0.5289	aprendizaje=0.4255, árboles=43
XGBoost	121	0.5199	0.5188	0.5238	aprendizaje=0.4153, profundidad=10, reg_lambda=1.225
	61	0.5217	0.5029	0.5353	aprendizaje=0.3207, profundidad=8, reg_lambda=0.5629
Aleatorio	0	0.5000			
Local	0	0.5462			

Tabla 13: Resultados de regresión con datos de los últimos 5 enfrentamientos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	121	0.5118	0.4798	0.5184	k=1, pesos=uniforme
Regresión Lineal	121	0.5408	0.5282	0.5221	alpha=4.951, f=saga
	8	0.5163	0.5245	0.5376	alpha=4.846, f=saga
SVR	121	0.5462	0.5204	0.5373	C=1.303
	31	**0.5471**	**0.5484**	**0.5515**	C=0.8063
Decision Tree	121	0.5072	0.5149	0.5272	ccp_alpha=0.04444, profundidad=4, criterio=mse
	1	0.5453	0.5175	0.5112	ccp_alpha=0.01817, profundidad=8, criterio=mse
Random Forest	121	0.5199	0.4992	0.5422	ccp_alpha=0.02896, profundidad=6, max_variables=no, árboles=26
	100	0.5236	0.5072	0.5426	ccp_alpha=0.00068070, profundidad=5, max_variables=sqrt, árboles=45
Red Neuronal	121	0.5399	0.5304	0.5202	activación=tanh, alpha=0.3212, nodos=[39], f=lbfgs
AdaBoost	121	0.5263	0.5195	0.5425	aprendizaje=0.05938, árboles=118
	65	0.5236	0.4971	0.5399	aprendizaje=0.1995, árboles=21
XGBoost	121	0.5362	0.5090	0.5359	aprendizaje=0.01456, profundidad=4, reg_lambda=0.8542
	1	0.5462	0.5153	0.5014	aprendizaje=0.004016, profundidad=1, reg_lambda=0.8562
Aleatorio	0	0.5000			
Local	0	0.5462			

Finalmente, al incluir los últimos 10 enfrentamientos, los resultados obtenidos mostrados en las Tablas 14 y 15 no superan aquellos en los que N=5, como sucede al utilizar los últimos partidos en lugar de enfrentamientos. Esto puede deberse a la pérdida del 79% de los datos al no poder incluir los primeros 9 enfrentamientos entre cada par de equipos.

Tabla 14: Resultados de clasificación con datos de los últimos 10 enfrentamientos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	121	0.5294	0.5131	0.5423	k=7, pesos=distancia
Regresión Logística	121	0.5389	0.5123	0.5284	C=0.0223, f=liblinear
	6	0.5275	0.4959	0.5612	C=0.7765, f=liblinear
SVC	121	0.4991	0.4753	0.5246	C=0.5974
	29	0.5009	0.5167	**0.5783**	C=0.9903
Decision Tree	121	0.5446	0.5000	0.5124	ccp_alpha=0.005751, profundidad=7, criterio=entropía
	50	0.5465	**0.5405**	0.5325	ccp_alpha=0.003115, profundidad=8, criterio=entropía
Random Forest	121	0.5446	0.5206	0.5366	ccp_alpha=0.006418, profundidad=1, max_variables=no, árboles=59, criterio=entropía
	114	**0.5560**	0.5403	0.5414	ccp_alpha=0.0009752, profundidad=6, max_variables=sqrt, árboles=128, criterio=entropía
Red Neuronal	121	0.5446	0.5009	0.5345	activación=logística, alpha=0.5727, nodos=[33, 33], f=adam
Naive Bayes	121	0.4972	0.5076	0.5403	var_smoothing=6.401e-08
AdaBoost	121	0.5294	0.5201	0.5231	aprendizaje=0.3662, árboles=380
	1	0.5085	0.4809	0.5497	aprendizaje=0.9829, árboles=498
XGBoost	121	0.5370	0.5250	0.5371	aprendizaje=0.3712, profundidad=2, reg_lambda=1.101
	102	0.5560	0.5401	0.5406	aprendizaje=0.1796, profundidad=5, reg_lambda=0.5749
Aleatorio	0	0.5000			
Local	0	0.5446			

Tabla 15: Resultados de regresión con datos de los últimos 10 enfrentamientos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	121	0.4953	0.4939	0.5371	k=10, pesos=distancia
Regresión Lineal	121	0.4953	0.4968	0.5310	alpha=4.059, f=lsqr
	2	0.5275	0.5079	0.5471	alpha=0.3629, f=saga
SVR	121	0.5142	0.5078	0.5444	C=0.8527
	14	0.5161	0.5133	**0.5710**	C=1.485
Decision Tree	121	0.5161	0.4871	0.5499	ccp_alpha=0.03379, profundidad=7, criterio=friedman_mse
	1	0.5218	0.5149	0.5291	ccp_alpha=0.01527, profundidad=8, criterio=mse
Random Forest	121	0.5332	0.5244	0.5569	ccp_alpha=0.01983, profundidad=5, max_variables=sqrt, árboles=142
	79	**0.5617**	0.5184	0.5559	ccp_alpha=0.03123, profundidad=2, max_variables=sqrt, árboles=21
Red Neuronal	121	0.4915	0.4829	0.5346	activación=logística, alpha=0.1676, nodos=[30, 30, 30], f=lbfgs
AdaBoost	121	0.5370	**0.5465**	0.5437	aprendizaje=0.08333, árboles=99
	8	0.5446	0.5376	0.5282	aprendizaje=0.6199, árboles=343
XGBoost	121	0.4991	0.4855	0.5437	aprendizaje=0.8321, profundidad=10, reg_lambda=1.197
	1	0.5370	0.5319	0.5343	aprendizaje=0.1988, profundidad=1, reg_lambda=0.7573
Aleatorio	0	0.5000			
Local	0	0.5446			

6.3.3. TEMPORADA PASADA

Las Tablas 16 y 17 muestran los resultados obtenidos al utilizar el conjunto de variables relativo a la temporada anterior a la fecha del partido.

Tabla 16: Resultados de clasificación con datos de la última temporada

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	122	0.5178	0.5212	0.5244	k=7, pesos=uniforme
Regresión Logística	122	0.5566	0.5657	0.5775	C=0.8380, f=liblinear
	10	0.5543	0.5575	0.5843	C=0.9193, f=liblinear
SVC	122	0.5525	0.5666	0.5782	C=0.9837
	10	0.5367	0.5587	0.5822	C=0.9828
Decision Tree	122	0.5552	0.5409	0.5522	ccp_alpha=0.001591, profundidad=3, criterio=entropía
	64	0.5579	0.5615	0.5573	ccp_alpha=0.001436, profundidad=4, criterio=entropía
Random Forest	122	0.5606	0.5854	0.5853	ccp_alpha=0.0004574, profundidad=6, max_variables=sqrt, árboles=143, criterio=gini
	96	0.5336	0.5744	0.5768	ccp_alpha=0.002418, profundidad=1, max_variables=sqrt, árboles=133, criterio=gini
Red Neuronal	122	0.5507	0.5660	0.5769	activación=tanh, alpha=0.1068, nodos=[57], f=adam
Naive Bayes	122	0.5358	0.5382	0.5315	var_smoothing=6.910e-08
AdaBoost	122	0.5750	0.5831	0.5803	aprendizaje=0.09561, árboles=247
	42	0.5683	0.5804	**0.5903**	aprendizaje=0.3943, árboles=89
XGBoost	122	**0.5759**	**0.5854**	0.5793	aprendizaje=0.2545, profundidad=1, reg_lambda=1.433
	45	0.5647	0.5737	0.5855	aprendizaje=0.02735, profundidad=4, reg_lambda=1.409
Aleatorio	0	0.5000			
Local	0	0.5362			

Comparando los resultados de las Tablas 16 y 8, observamos una mejora en el *accuracy* de 13 modelos, todos salvo *random forest* con selección de variables y *naïve Bayes*. Además, el mejor de los modelos aumenta su *accuracy* en más de 2 puntos porcentuales respecto al mejor de los casos basados en los N últimos partidos.

Tanto en clasificación como en regresión (mostrados en la Tabla 17), la mayoría de los modelos superan el *accuracy* respecto a la predicción mayoritaria, llegando en el mejor de los casos a un incremento del 7.42% (3.98 p.p.).

Tabla 17: Resultados de regresión con datos de la última temporada

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	122	0.5043	0.5114	0.5311	k=8, pesos=distancia
Regresión Lineal	122	0.5566	0.5649	0.5801	alpha=4.293, f=lsqr
	12	0.5475	0.5614	0.5845	alpha=0.4894, f=sparse_cg
SVR	122	0.5511	0.5642	0.5628	C=0.5512
	22	0.5336	0.5653	0.5875	C=0.5859
Decision Tree	122	0.5421	0.5511	0.5728	ccp_alpha=0.03631, profundidad=3, criterio=friedman_mse
	1	0.5295	0.4974	0.5040	ccp_alpha=0.01151, profundidad=10, criterio=friedman_mse
Random Forest	122	0.5570	0.5710	0.5894	ccp_alpha=0.01067, profundidad=4, max_variables=sqrt, árboles=100
	86	0.5665	**0.5734**	**0.5901**	ccp_alpha=0.01200, profundidad=5, max_variables=sqrt, árboles=142
Red Neuronal	122	0.5489	0.5647	0.5690	activación=relu, alpha=0.8632, nodos=[21, 21], f=lbfgs
AdaBoost	122	0.5602	0.5671	0.5821	aprendizaje=0.02634, árboles=292
	61	0.5584	0.5686	0.5836	aprendizaje=0.07074, árboles=232
XGBoost	122	**0.5683**	0.5720	0.5846	aprendizaje=0.05429, profundidad=1, reg_lambda=1.096
	1	0.5115	0.5099	0.5258	aprendizaje=0.04367, profundidad=5, reg_lambda=1.107
Aleatorio	0	0.5000			
Local	0	0.5362			

Comprobamos por tanto que usar los datos de una temporada completa otorga mejores resultados que los obtenidos utilizando los N últimos partidos o enfrentamientos. A pesar de que el rendimiento de un jugador puede verse modificado de una temporada a otra, es esperable que sus estadísticas y el valor que aporta al equipo se mantengan relativamente estables. En estos casos, los datos de hasta 162 partidos (si un jugador es titular durante toda la temporada) evitan en mayor medida la aparición de valores anómalos y sirven para generalizar las predicciones de forma más acertada.

6.3.4. HISTÓRICO

Los resultados de los modelos predictivos basados en el conjunto total de los datos de una década de partidos se muestran en las Tablas 18 y 19.

Tabla 18: Resultados de clasificación con datos históricos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	122	0.5288	0.5339	0.5187	k=10, pesos=distancia
Regresión Logística	122	0.5676	0.5811	0.5718	C=0.7800, f=saga
	41	0.5708	0.5785	0.5860	C=0.9484, f=saga
SVC	122	0.5526	0.5831	0.5724	C=0.9678
	39	0.5559	0.5808	0.5862	C=0.9652
Decision Tree	122	0.5639	0.5555	0.5411	ccp_alpha=0.001828, profundidad=7, criterio=entropía
	112	0.5482	0.5394	0.5429	ccp_alpha=0.001663, profundidad=3, criterio=entropía
Random Forest	122	**0.5728**	**0.5974**	0.5756	ccp_alpha=0.0006414, profundidad=4, max_variables=no, árboles=85, criterio=gini
	90	0.5619	0.5892	0.5745	ccp_alpha=0.001237, profundidad=10, max_variables=sqrt, árboles=77, criterio=gini
Red Neuronal	122	0.5607	0.5786	0.5725	activación=logística, alpha=0.9870, nodos=[58], f=lbfgs
Naive Bayes	122	0.5579	0.5611	0.5448	var_smoothing=9.096e-08
AdaBoost	122	0.5676	0.5864	0.5773	aprendizaje=0.07577, árboles=141
	40	0.5579	0.5809	**0.5893**	aprendizaje=0.1013, árboles=454
XGBoost	122	0.5635	0.5850	0.5778	aprendizaje=0.09335, profundidad=1, reg_lambda=0.6033
	74	0.5623	0.5851	0.5785	aprendizaje=0.08782, profundidad=1, reg_lambda=1.054
Aleatorio	0	0.5000			
Local	0	0.5362			

Tabla 19: Resultados de regresión con datos históricos

Modelo	Variables	Acc test	AUC test	AUC CV	Hiperparámetros
k-NN	122	0.5244	0.5354	0.5239	k=10, pesos=distancia
Regresión Lineal	122	0.5510	0.5756	0.5759	alpha=3.212, f=lsqr
	20	0.5591	0.5776	**0.5834**	alpha=1.151 f=sparse_cg
SVR	122	0.5627	0.5821	0.5711	C=0.7361
	46	0.5365	0.5777	0.5814	C=0.5506
Decision Tree	122	0.5607	0.5470	0.5555	ccp_alpha=0.01276, profundidad=4, criterio=friedman_mse
	1	0.5284	0.5248	0.5359	ccp_alpha=0.009876, profundidad=8, criterio=mse
Random Forest	122	0.5664	0.5911	0.5803	ccp_alpha=0.04222, profundidad=4, max_variables=sqrt, árboles=143
	96	0.5728	**0.5929**	0.5813	ccp_alpha=0.01593, profundidad=7, max_variables=sqrt, árboles=83
Red Neuronal	122	0.5526	0.5710	0.5749	activación=relu, alpha=0.8105, nodos=[28, 28, 28], f=adam
AdaBoost	122	0.5510	0.5838	0.5560	aprendizaje=0.6335, árboles=26
	119	**0.5805**	0.5874	0.5716	aprendizaje=0.005820, árboles=428
XGBoost	122	0.5692	0.5883	0.5754	aprendizaje=0.02599, profundidad=3, reg_lambda=1.280
	1	0.5018	0.4917	0.5063	Aprendizaje=0.4225, profundidad=4, reg_lambda=1.484
Aleatorio	0	0.5000			
Local	0	0.5362			

En ambos casos, 10 de los 15 modelos obtienen resultados superiores en cuanto a *accuracy* respecto a aquellos que usan un solo año de datos.

Los modelos que logran tanto el mejor *accuracy* como AUC de todo el estudio utilizan la configuración de variables histórica: la regresión mediante AdaBoost con selección de variables y la clasificación de *random forest* sin selección de variables, respectivamente.

Dicho modelo de AdaBoost logra clasificar correctamente el 58.05% de los partidos del conjunto de test, lo que supone un aumento del 16.10% (8.05 p.p.) frente a la predicción aleatoria de los ganadores con un acierto teórico del 50%, y una mejora del 8.20% (4.40 p.p.) respecto al modelo que predice siempre como ganador al equipo local.

7. DISCUSIÓN

El mejor de los resultados que hemos obtenido, con una tasa de acierto del 58.05% de los partidos, supera en primer lugar a la simple predicción de victoria local independientemente de los datos de entrada en un 8.20%, al obtener este un *accuracy* de 0.5365.

Distintos autores han tomado enfoques diferentes a la hora de generar predicciones en torno al béisbol. Múltiples sistemas podrían usarse de forma conjunta, pero a la hora de analizar la eficacia de nuestro estudio, debemos compararlo con aquellos que abordan el problema de la predicción de partidos de forma similar.

Yang y Swartz [21] utilizan modelos bayesianos para predecir el porcentaje total de victorias para cada equipo en base al porcentaje de victorias entre los equipos, la media de BA de los equipos y ERA de los *pitchers*. La efectividad del modelo ratifica la capacidad predictiva de dichas variables, pero se centran únicamente en el resultado global a final de la temporada y no en partidos concretos.

Estudios como el de **Chen et al.** [22] utilizan los datos de la liga regular para predecir qué equipos ganarán las eliminatorias de la *World Series*, por lo que no son plenamente comparables con nuestros resultados. En nuestro caso, predecimos los resultados de partidos individuales, a diferencia del ganador de las eliminatorias a 5 o 7 partidos. La mayoría de los modelos que probaron obtienen un *accuracy* cercano a 0.5, siendo el mejor de sus resultados el conseguido mediante la regresión logística con datos de ambos equipos durante los años 1975-2016, logrando acertar el 62.50% de los *playoffs*. Concluyen también que unas de las métricas más relevantes son los *strikeouts* recibidos por los bateadores e inducidos por los *pitchers* y otras variables relacionadas con los lanzadores como HAA.

Los trabajos de **Jensen et al.** [23] y **Brown** [24] predicen varias métricas que reflejan el rendimiento ofensivo de los jugadores, las cuales podrían ser usadas como variables adicionales.

Soto-Valero [25] lleva a cabo una experimentación similar a la aquí estudiada, en la que predice los resultados de los partidos utilizando varios métodos como k-NN, redes neuronales, *decision trees* y SVMs, tanto en clasificación como en regresión. Los

resultados que muestra en la Tabla 3 son similares a los que hemos obtenido, con medias de *accuracy* entre 0.5597 y 0.5892. Cabe destacar, sin embargo, que los modelos son entrenados de forma separada para cada equipo, utilizando 10 años previos de datos de forma similar a nuestra configuración de variables históricas.

Elfrink [26] utiliza una serie de variables de la temporada pasada y partidos recientes para resolver los problemas de regresión y clasificación, obteniendo en el mejor de los casos un 55.52% de predicciones correctas.

Finalmente, **Jia et al.** [27] comparan sus resultados utilizando técnicas de clasificación y regresión respecto al modelo trivial que predice siempre la victoria del equipo local y respecto a las probabilidades predichas por las casas de apuestas. Utilizan un conjunto de variables relativas al bateo, *pitching* y estadísticas del equipo similar al que proponemos, incluyendo algunas otras métricas como el número de corredores que alcanzan una base pero no llegan a anotar.

A la hora de calcular las estadísticas de los jugadores, utilizan únicamente los datos de los partidos anteriores desde el inicio de la temporada, lo que dificulta las predicciones de los primeros partidos del año. Tras entrenar modelos de regresión logística, SVM, AdaBoost y LogitBoost, obtienen resultados positivos, con *accuracies* sobre el conjunto de test: 0.557, 0.596, 0.585 y 0.595, respectivamente, por encima de las predicciones de las casas de apuestas que obtienen un 0.569. De forma similar a lo sucedido en nuestro caso, los modelos de regresión son ligeramente inferiores a los de clasificación, con *accuracies* de 0.571 y 0.590 al examinar modelos de regresión lineal y *random forest*. Al entrenar distintos modelos predictivos para cada mes de la temporada, los resultados mejoran hasta llegar al 64.1% de clasificaciones correctas en el mes de septiembre. Al tener unos datos más completos de la temporada actual y al estar la clasificación de la liga más decidida, los meses finales resultan más fáciles de predecir. Por el contrario, sus resultados empeoran en los primeros meses del año.

Los resultados que hemos obtenido, por tanto, resultan similares a estudios anteriores. No solo en la precisión de las predicciones, sino también en las diferencias entre clasificación y regresión o en la importancia de ciertas variables.

8. CONCLUSIONES Y LÍNEAS FUTURAS

El mejor de los modelos que hemos probado ha sido capaz de predecir un 58.05% de los partidos de test correctamente, por encima de los modelos triviales y con resultados comparables al resto de estudios consultados.

Como norma general, los modelos basados en la clasificación producen resultados ligeramente superiores a los de regresión. A pesar de ello, obtenemos el mejor *accuracy* mediante AdaBoost en regresión.

La temporalidad de los datos utilizados es de gran importancia. Como hemos podido comprobar, se consigue una mejora consistente al incrementar la cantidad de partidos que engloban las variables. Empleando los datos históricos se obtienen mejores resultados que en los modelos que incluyen solamente la pasada temporada, y a su vez esos son más eficaces que los que utilizan los últimos 10, 5 y 3 partidos.

A pesar de obtener un error elevado, la predicción de partidos individuales de béisbol resulta un problema complejo en el que muchos estudios obtienen un porcentaje de acierto entorno al 55-60%.

Tras comprobar la mejoría de los modelos al usar conjuntos de datos basados en tiempos más prolongados, consideramos que el uso de datos de más de 10 años puede ser beneficioso. Sin embargo, como Chen et al. [22] comprobaron, los resultados empeoran al incluir las temporadas 1930-1975 debido a los cambios en las reglas del juego y las diferencias históricas en el deporte.

En los casos en los que se emplean únicamente datos de la misma temporada que se predice, es necesario incluir una mayor cantidad de temporadas para compensar la pérdida de ejemplos de entrenamiento.

Por otra parte, podrían incluirse otras variables que estén relacionadas con la probabilidad de victoria, como la edad y años de experiencia de los jugadores como estudiamos anteriormente, u otras estadísticas que involucren más aspectos ofensivos y defensivos como *wins above replacement* que resumen el valor aportado por los jugadores a la victoria del equipo en comparación con un jugador sustituto.

Incorporar otros tipos de datos como la velocidad de los lanzamientos y sus tipos (bolas rápidas, curvas, etc.) junto con la mano dominante del lanzador y bateador podría llevar a obtener mejores resultados, ya que en la mayoría de los casos estos se estudian de forma independiente a las probabilidades de victorias.

BIBLIOGRAFÍA

[1] T. Lepperd, “Official Baseball Rules 2019 Edition.” p. 173, 2019. Accedido: 13-jul-2021. [En línea]. Disponible en: https://content.mlb.com/documents/2/2/4/305750224/2019_Official_Baseball_Rules_FINAL_.pdf.

[2] “Statcast | Glossary | MLB.com.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.mlb.com/glossary/statcast

[3] M. Fast, “What the Heck is PITCHf/x?,” *Hardball Times Baseb. Annu. 2010*, pp. 1–6, 2010.

[4] D. Sheldon, “Baseball Sports Betting Tops $1 Billion at Nevada Sportsbooks in 2016.”, 2016. Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.casino.org/news/major-league-baseball-sports-betting-totals-1-billion-at-nevada-sportsbooks-in-2016/

[5] D. Purdum, “MLB - A record $1 billion was bet on Major League Baseball in Nevada in 2016.”, 2016. Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.espn.com/chalk/story/_/id/18220934/mlb-record-1-billion-was-bet-major-league-baseball-nevada-2016

[6] “Postseason Share | Glossary | MLB.com.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.mlb.com/glossary/miscellaneous/postseason-share

[7] “Download Lahman’s Baseball Database – SeanLahman.com.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://web.archive.org/web/20210609014425/http://www.seanlahman.com/baseball-archive/statistics/

[8] “MLB Stats, Scores, History, & Records | Baseball-Reference.com.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.baseball-reference.com/

[9] “MLB Scores: Real-time baseball scores and highlights.”. Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.mlb.com/scores

[10] “Retrosheet.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.retrosheet.org/

[11] “Baseball Databank.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://github.com/chadwickbureau/baseballdatabank/

[12] “Retrosheet Event Files.” Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.retrosheet.org/eventfile.htm

[13] M. Jafari y N. Ansari-Pour, “Why, When and How to Adjust Your P Values?,” *Cell J.*, vol. 20, no. 4, p. 604, 2019, doi: 10.22074/CELLJ.2019.5992.

[14] S. V. Allen y W. G. Hopkins, “Age of Peak Competitive Performance of Elite Athletes: A Systematic Review,” *Sport. Med. 2015 4510*, vol. 45, no. 10, pp. 1431–1441, jun. 2015, doi: 10.1007/S40279-015-0354-3.

[15] R. Schulz, D. Musa, J. Staszewski, y R. S. Siegler, “The relationship between age and major league baseball performance: Implications for development,” *Psychol. Aging*, vol. 9, no. 2, pp. 274–286, 1994, doi: 10.1037/0882-7974.9.2.274.

[16] J. G. Classé *et al.*, “Association between visual reaction time and batting, fielding, and earned run averages among players of the Southern Baseball League.,” *J. Am. Optom. Assoc.*, vol. 68, no. 1, pp. 43–49, 1997.

[17] S. Kalkman, “Optimizing Your Lineup By The Book - Beyond the Box Score”, 2012. Accedido: 13-jul-2021. [En línea]. Disponible en: https://www.beyondtheboxscore.com/2009/3/17/795946/optimizing-your-lineup-by

[18] R. Olson, “Does batting order matter in Major League Baseball? A simulation approach”, 2012. Accedido: 13-jul-2021. [En línea]. Disponible en: http://www.randalolson.com/2018/07/04/does-batting-order-matter-in-major-league-baseball-a-simulation-approach/

[19] J. H. Friedman, “Greedy function approximation: A gradient boosting machine.,” *https://doi.org/10.1214/aos/1013203451*, vol. 29, no. 5, pp. 1189–1232, oct. 2001, doi: 10.1214/AOS/1013203451.

[20] D. P. Kingma y J. Ba, “Adam: A Method for Stochastic Optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, dic. 2014. Accedido: 27-ago-2021. [En línea]. Disponible en: https://arxiv.org/abs/1412.6980v9.

[21] T. Y. Yang, T. Swartz, y A. L. East, “A Two-Stage Bayesian Model for Predicting Winners in Major League Baseball,” *J. Data Sci.*, vol. 2, pp. 61–73, 2004.

[22] R. Chen, A. Hobbs, y W. Maier, “Predicting Baseball Postseason Results from Regular Season Data,” 2016.

[23] S. T. Jensen, B. B. McShane, y A. J. Wyner, “Hierarchical bayesian modeling of hitting performance in baseball,” Bayesian Anal., vol. 4, no. 4, pp. 631–652, 2009, doi: 10.1214/09-BA424.

[24] L. D. Brown, “In-season prediction of batting averages: A field test of empirical Bayes and Bayes methodologies,” Ann. Appl. Stat., vol. 2, no. 1, pp. 113–152, 2008, doi: 10.1214/07-AOAS138.

[25] C. Soto-Valero, “Predicting win-loss outcomes in MLB regular season games-a comparative study using data mining methods,” Int. J. Comput. Sci. Sport, vol. 15, no. 2, pp. 91–112, 2016, doi: 10.1515/ijcss-2016-0007.

[26] T. Elfrink, “Predicting the outcomes of MLB games with a machine learning approach,” 2018.

[27] R. Jia, C. Wong, y D. Zeng, “Predicting the Major League Baseball Season,” 2013.

ANEXOS

i. Abreviaturas y definiciones

Abreviatura	Inglés	Castellano	Descripción
1B	1: First baseman 2: Single	1: Primera base 2: Sencillo	1: Jugador del equipo defensivo situado cerca de la primera base 2: El bateador avanza hasta primera base tras un bateo hacia la zona de juego
2B	1: Second baseman 2: Double	1: Segunda base 2: Doble	1: Jugador del equipo defensivo situado entre la primera y segunda base 2: El bateador avanza hasta segunda base tras un bateo hacia la zona de juego
3B	1: Third baseman 2: Triple	1: Tercera base 2: Triple	1: Jugador del equipo defensivo situado cerca de la tercera base 2: El bateador avanza hasta tercera base tras un bateo hacia la zona de juego
A	Assist	Asistencia	Otorgada a los jugadores defensivos que han participado en la eliminación de un corredor
AB	At bat	Turno al bate	Turno al bate completado, salvo en las excepciones: SF, SH, BB, HBP, Interferencia, Obstrucción
AL	American League	Liga Americana	Una de las dos divisiones de la MLB
B	Ball	Bola	Lanzamiento que no atraviesa el área de bateo y el bateador no trata de golpear
BA	Batting average	Media de bateo	Cantidad de bateos exitosos dividido entre turnos al bate
BB	Base on balls	Base por bolas	El bateador avanza a primera base de forma automática por haber recibido cuatro bolas
BBP	Base on balls (pitcher)	Base por bolas (lanzador)	Análogo a BB, pero atribuido al lanzador
BF	Batters faced	Bateadores enfrentados	Cantidad de bateadores que completan PAs contra el lanzador
BK	Balk	Balk	Movimiento del lanzador no permitido, tras el cual todos los corredores avanzan una base
C	Catcher	Receptor	Jugador del equipo defensivo situado detrás del bateador. Normalmente, decide los tipos y posiciones de los lanzamientos.
CF	Center fielder	Jardinero central	Jugador del equipo defensivo situado en la zona central del jardín exterior
CPBL	Chinese Professional Baseball League	Liga de Béisbol Profesional China	Liga profesional de China
DH	Designated hitter	Bateador asignado	Jugador cuya función es batear en sustitución del lanzador. Solo está permitido en la AL
	Draft	Draft	Proceso de selección de nuevos jugadores realizado de forma anual
E	Error	Error	Error no forzado de un defensor que causa que al menos un corredor avance una base
ER	Earned Run	Carrera limpia	Carrera permitida por el lanzador, sin que haya habido ningún error defensivo
ERA	Earned Run Average	Media de carreras limpias	Carreras permitidas por el lanzador normalizado por la cantidad de bateadores contra los que se ha enfrentado
FEP	Fielding error percentage	Porcentaje de errores defensivos	Cantidad de errores causados por un jugador dividido entre las jugadas defensivas en las que ha tomado parte (A+PO+E)
H	Hit	Jit, imparable	El bateador llega de forma segura a una base sin error de la defensa tras un bateo hacia la zona de juego
HA	Hits allowed	Hits permitidos	Cantidad de *hits* permitidos por el lanzador
HAA	Hits allowed average	Media de hits permitidos	HA/BF
HB	Hit batsman	Bateador golpeado	Análogo a HBP pero atribuido al lanzador
HBP	Hit by pitch	Golpeado por lanzamiento	El bateador avanza a primera base de forma automática por haber sido golpeado en un lanzamiento
HR	Home run	Jonrón	Bateo que sobrepasa las vallas del campo, permitiendo anotar al bateador y todos los corredores
	Inning	Entrada	División de un partido en el que ambos equipos completan tres eliminaciones
	Interference	Interferencia	Situación en la que una persona altera el curso de la jugada impidiendo llevar a cabo su función a otro jugador

Abreviatura	Inglés	Castellano	Descripción
K	Strikeout	Ponchado	El lanzador consigue eliminar al bateador tras recibir tres strikes
KA	Strikeout average	Media de strikeouts	K/BF
KBO	Korea Baseball Organization	Organización Coreana de Béisbol	Liga profesional de Corea del Sur
LBPRC		Liga de Béisbol Profesional Roberto Clemente	Liga profesional de Puerto Rico
LF	Left fielder	Jardinero izquierdo	Jugador del equipo defensivo situado en la zona izquierda del jardín exterior
LMP	Mexican Pacific League	Liga Mexicana del Pacífico	Liga profesional de México
LVBP	Venezuelan Professional Baseball League	Liga Venezolana de Béisbol Profesional	Liga profesional de Venezuela
MLB	Major League Baseball	Liga Mayor de Béisbol	Principal liga de mayor nivel en los EE.UU.
NL	National League	Liga Nacional	Una de las dos divisiones de la MLB
NPB	Nippon Professional Baseball	Liga Japonesa de Béisbol Profesional	Liga profesional de Japón
OBP	On-base percentage	Porcentaje de embasado	(H+BB+HBP)/(AB+BB+HBP+SF)
	Obstruction	Obstrucción	Situación en la que un defensor sin posesión de la pelota ni en proceso de obtenerla impide el avance de un corredor
OPS	On-base plus slugging	Embasado más slugging	OBP+SLG
P	Pitcher	Lanzador	Jugador del equipo defensivo que inicia las jugadas lanzando la pelota al bateador.
PE	Pitcher errors	Errores del lanzador	Estadística que incluye los resultados negativos causados por el lanzador: (BBP+HB+BK)/BF
PF	Park factor	Factor campo	Estadística que mide lo favorable que es un campo a los bateadores
PG%	Played games percentage	Porcentaje de partidos jugados	Porcentaje de partidos ya disputados de la temporada actual
PIT	Pitch	Lanzamiento	Lanzamiento completado por el *pitcher*
PITA	Pitch average	Media de lanzamientos	PIT/BF
Playoff	Playoff	Eliminatoria	Ronda eliminatoria del campeonato, al mejor de 1, 5 o 7 partidos dependiendo de la ronda
PO	Putout	Eliminación	Otorgado al defensor que completa la eliminación de un corredor
R	Run	Carrera	Punto anotado por un corredor tras recorrer todas las bases
RA	Run average	Media de carreras	R/AB
RF	Right fielder	Jardinero derecho	Jugador del equipo defensivo situado en la zona derecha del jardín exterior
S	Strike	Strike	Lanzamiento que atraviesa el área de bateo sin que el bateador consiga golpearlo o realice un bateo no válido
SBP	Stolen base percentage	Porcentaje de bases robadas	Cantidad de bases robadas (avances antes de un bateo) con éxito dividido entre los intentos de robar bases
SF	Sacrifice fly	*Fly* de sacrificio	Bateo de larga distancia con la intención de avanzar corredores a pesar de quedar eliminado el bateador
SH	Sacrifice hit	Toque de sacrificio	Bateo con la intención de permitir avanzar a otros corredores a costa de la eliminación del bateador
SLG	Slugging average	Media de *slugging*	(1B+2*2B+3*3B+4*HR)/AB
SO	Strikeout	Ponchado	El bateador es eliminado tras recibir 3 *strikes*
SS	Shortstop	Campocorto	Jugador del equipo defensivo situado entre la segunda y tercera base
W%	Win percentage	Porcentaje de victorias	Porcentaje de victorias de un equipo sobre los partidos jugados
WS	World Series	Serie Mundial	Campeonato de postemporada de la MLB

ii. Selección de variables

a) Últimos 3 partidos

Regresión Logística: L_P_SBP, L_P_KA, L_P_ERA, L_P_PE, L_2B_FEP, V_P_KA, V_P_PE, V_RF_FEP

SVC: L_P_KA, L_P_PE, L_3B_FEP, L_CF_SLG, V_P_FEP, V_P_PE, V_3B_FEP, V_SS_RA

Decision Tree (C): PG%, L_P_PITA, L_P_ERA, L_P_PE, V_P_PITA, V_P_PE

Random Forest (C): PG%, PF, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_SOA, L_LF_BA, L_LF_SLG, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_SOA, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_3B_BA, V_3B_SLG, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_SOA, V_LF_BA, V_LF_SLG, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_SOA

AdaBoost (C): L_P_PITA, L_P_KA, L_RF_BA, V_P_PITA, V_P_KA

XGBoost (C): PG%, PF, L_P_SLG, L_P_RA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_SOA, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_SLG, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SBP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_CF_FEP, V_RF_SLG, V_RF_RA, V_RF_SBP

Regresión Lineal: L_P_KA, L_2B_FEP, V_P_KA

SVR: PF, L_P_SBP, L_P_FEP, L_P_KA, L_P_ERA, L_P_PE, L_C_SOA, L_C_SBP, L_1B_RA, L_1B_SBP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_FEP, L_SS_FEP, L_LF_BA, L_CF_RA, L_RF_BA, L_RF_SOA, V_P_RA, V_P_SBP, V_P_FEP, V_P_KA, V_P_HAA, V_P_PE, V_C_RA, V_C_SOA, V_C_FEP, V_1B_FEP, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_CF_RA, V_CF_FEP, V_RF_FEP

Decision Tree (R): L_P_PITA

Random Forest (R): PG%, PF, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_SOA, L_LF_BA, L_LF_SLG, L_LF_SOA, L_CF_SLG, L_CF_SOA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SLG, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_BA, V_2B_SLG, V_2B_SOA, V_3B_BA, V_3B_SLG, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_SOA, V_LF_SLG, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_SOA, V_RF_SLG, V_RF_SOA

AdaBoost (R): L_P_KA, L_LF_SLG, L_CF_RA, V_P_KA, V_RF_SOA

XGBoost (R): V_RF_SLG

b) Últimos 5 partidos

Regresión Logística: L_P_KA, L_2B_FEP, V_P_KA, V_P_PE

SVC: PF, L_P_BA, L_P_SLG, L_P_SOA, L_P_SBP, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_RA, L_C_SOA, L_C_FEP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SBP, L_3B_FEP, L_SS_SOA, L_SS_FEP, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_RA, V_P_SBP, V_P_FEP, V_P_KA, V_P_ERA, V_P_PE, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_RA, V_1B_FEP, V_2B_BA, V_2B_RA, V_2B_FEP, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_LF_FEP, V_CF_RA, V_CF_FEP, V_RF_BA, V_RF_SBP, V_RF_FEP

Decision Tree (C): PG%, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_2B_BA, L_2B_SOA, L_3B_BA, L_SS_BA, L_LF_RA, L_LF_SOA, L_CF_SLG, L_RF_BA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SLG, V_C_SOA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_SOA, V_3B_SLG, V_SS_SLG, V_SS_SOA, V_LF_BA, V_LF_SLG, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_SOA

Random Forest (C): PG%, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_SOA, L_LF_BA, L_LF_SLG, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SLG, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_SOA, V_2B_BA, V_2B_SLG, V_2B_SOA, V_3B_BA, V_3B_SLG, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_SOA, V_LF_BA, V_LF_SLG, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_SOA

AdaBoost (C): L_P_HAA, L_P_ERA, L_P_PE, L_C_RA, L_1B_SLG, L_2B_RA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_FEP, L_SS_BA, L_LF_SOA, L_CF_RA, L_RF_BA, L_RF_SLG, V_P_KA, V_P_HAA, V_P_PE, V_C_SOA, V_2B_SOA, V_2B_FEP, V_3B_FEP, V_SS_BA, V_LF_BA, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_RA, V_CF_FEP, V_RF_BA

XGBoost (C): L_LF_FEP, L_CF_RA, V_P_SLG, V_P_KA, V_P_HAA, V_P_PE, V_C_SOA, V_1B_SOA, V_3B_SOA, V_SS_SLG, V_CF_RA

Regresión Lineal: L_P_KA, L_P_ERA, L_2B_FEP, V_P_KA, V_P_PE

SVR: PF, L_P_SOA, L_P_FEP, L_P_KA, L_P_ERA, L_P_PE, L_C_RA, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_FEP, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_RF_BA, L_RF_RA, L_RF_SOA, L_RF_FEP, V_P_FEP, V_P_KA, V_P_HAA, V_P_PE, V_C_RA, V_C_SOA, V_C_FEP, V_1B_FEP, V_2B_FEP, V_SS_SOA, V_SS_FEP, V_LF_BA, V_CF_RA, V_RF_FEP

Decision Tree (R): L_P_PITA

Random Forest (R): PG%, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_SOA, L_1B_SLG, L_1B_SOA, L_2B_SLG, L_2B_SOA, L_3B_SLG, L_3B_SOA, L_SS_SLG, L_SS_SOA, L_LF_SLG, L_LF_SOA, L_CF_SLG, L_CF_SOA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SLG, V_C_SOA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_SLG, V_2B_SOA, V_3B_SLG, V_3B_SOA, V_SS_SLG, V_SS_SOA, V_LF_SLG, V_LF_SOA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_SLG, V_RF_SOA

AdaBoost (R): PF, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_SBP, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_RA, L_C_SOA, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_SBP, L_2B_BA, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SBP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_RF_SLG, L_RF_SOA, L_RF_SBP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_SBP, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_SBP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA

XGBoost (R): V_RF_SBP

c) Últimos 10 partidos

Regresión Logística: L_P_KA, V_P_KA

SVC: PF, L_P_BA, L_P_SLG, L_P_SOA, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_RA, L_C_SOA, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_FEP, L_3B_BA, L_3B_SOA, L_3B_SBP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_FEP, L_LF_RA, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_FEP, V_P_PITA, V_P_KA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_SOA, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_SOA, V_3B_FEP, V_SS_BA, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_SLG, V_LF_RA, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_SOA, V_CF_SBP, V_CF_FEP, V_RF_SLG, V_RF_FEP

Decision Tree (C): L_P_PITA, L_P_KA, L_P_HAA, L_C_RA, L_C_SOA, L_1B_RA, L_2B_BA, L_3B_BA, L_3B_SOA, L_LF_RA, L_LF_SOA, L_CF_BA, L_RF_SLG, L_RF_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_PE, V_C_RA, V_C_SOA, V_1B_SOA, V_2B_SLG, V_2B_SOA, V_3B_RA, V_3B_SOA, V_SS_RA, V_SS_SOA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SOA, V_RF_SOA

Random Forest (C): PG%, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_SOA, L_1B_SLG, L_1B_SOA, L_2B_SLG, L_2B_SOA, L_3B_SLG, L_SS_SLG, L_SS_SOA, L_LF_SLG, L_LF_SOA, L_CF_SLG, L_CF_SOA, L_RF_SLG, L_RF_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SLG, V_C_SOA, V_1B_SLG, V_1B_SOA, V_2B_SLG, V_2B_SOA, V_3B_SLG, V_3B_SOA, V_SS_SLG, V_SS_SOA, V_LF_SLG, V_LF_SOA, V_CF_SLG, V_CF_SOA, V_RF_SLG, V_RF_SOA

AdaBoost (C): V_P_PE

XGBoost (C): PG%, PF, L_P_PITA, L_P_KA, L_P_ERA, L_P_PE, L_C_BA, L_C_SOA, L_C_FEP, L_1B_SLG, L_1B_RA, L_1B_SOA, L_2B_SOA, L_3B_RA, L_3B_SOA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_LF_SLG, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_RA, V_P_SLG, V_P_PITA, V_P_KA, V_P_ERA, V_P_PE, V_C_SLG, V_C_RA, V_1B_BA, V_1B_SLG, V_1B_SOA, V_2B_BA, V_2B_RA, V_3B_SOA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_LF_SLG, V_CF_SBP, V_RF_SLG, V_RF_RA, V_RF_FE

Regresión Lineal: L_P_KA, L_P_ERA, L_P_PE, L_C_FEP, L_2B_FEP, V_P_KA, V_P_PE

SVR: L_P_BA, L_P_SLG, L_P_FEP, L_P_KA, L_P_ERA, L_P_PE, L_C_RA, L_1B_BA, L_1B_FEP, L_2B_RA, L_3B_BA, L_LF_FEP, L_CF_BA, L_CF_SLG, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_FEP, V_P_FEP, V_P_KA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_1B_RA, V_2B_FEP, V_SS_SOA, V_RF_FEP, V_RF_FEP

Decision Tree (R): V_P_KA

Random Forest (R): PG%, PF, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_BA, V_P_SLG, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_FEP

AdaBoost (R): PF, L_P_BA, L_P_SLG, L_P_SOA, L_P_PITA, L_P_KA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_SOA, L_C_FEP, L_1B_SLG, L_2B_BA, L_2B_SLG, L_2B_SBP, L_3B_BA, L_3B_SOA, L_SS_SLG, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_RA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_SOA, L_RF_SBP, V_P_SLG, V_P_SOA, V_P_SBP, V_P_PITA, V_P_KA, V_P_HAA, V_P_PE, V_C_BA, V_C_RA, V_C_SOA, V_1B_SLG, V_1B_RA, V_2B_SLG, V_2B_RA, V_2B_FEP, V_3B_SLG, V_SS_RA, V_SS_SOA, V_LF_BA, V_CF_SLG, V_CF_RA, V_RF_RA, V_RF_SOA

XGBoost (R): V_RF_FEP

d) Últimos 3 enfrentamientos

Regresión Logística: L_P_PE, V_P_KA, V_P_PE, V_CF_FEP

SVC: L_P_HAA

Decision Tree (C): PG%, L_P_PITA, L_P_KA, L_P_HAA, L_P_PE, L_C_BA, L_1B_SLG, L_3B_SLG, L_SS_BA, L_LF_SLG, L_LF_RA, L_CF_SLG, L_RF_SLG, L_RF_SOA, V_P_PITA, V_P_KA, V_3B_SLG, V_3B_SOA, V_SS_SLG, V_SS_SOA, V_LF_SLG, V_CF_BA, V_CF_SOA, V_RF_SLG

Random Forest (C): PG%, PF, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_SBP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, V_P_BA, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP

AdaBoost (C): L_P_PITA

XGBoost (C): L_P_PITA

Regresión Lineal: L_C_FEP, V_P_KA

SVR: L_P_SBP, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SOA, L_C_SBP, L_1B_BA, L_1B_SLG, L_1B_RA, L_SS_SBP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_FEP, L_CF_SLG, L_CF_FEP, L_RF_BA, L_RF_RA, L_RF_FEP, V_P_SBP, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SOA, V_1B_FEP, V_2B_SOA, V_3B_RA, V_3B_FEP, V_SS_SBP, V_LF_FEP, V_CF_RA, V_CF_FEP

Decision Tree (R): PG%

Random Forest (R): PG%, PF, L_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_SBP, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_SBP, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP

AdaBoost (R): PG%, PF, L_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_SBP, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_SBP, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SBP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_RF_BA, L_RF_SLG, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SOA, V_LF_SBP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA

XGBoost (R): V_RF_SOA

e) Últimos 5 enfrentamientos

Regresión Logística: L_P_KA, L_P_HAA, L_P_ERA, L_C_SOA, L_2B_SOA, L_2B_FEP, L_3B_SOA, V_P_KA, V_P_PE, V_C_FEP, V_2B_FEP, V_3B_FEP, V_SS_FEP, V_LF_FEP, V_CF_RA, V_CF_FEP

SVC: L_P_KA, L_2B_SOA, L_2B_FEP, L_SS_BA, L_SS_SLG, L_SS_FEP, L_RF_RA, V_P_KA, V_P_PE, V_C_SOA, V_C_FEP, V_2B_FEP, V_SS_FEP, V_LF_FEP, V_CF_RA, V_CF_FEP

Decision Tree (C): PG%, L_P_PITA, L_P_KA, L_P_PE, L_C_SOA, L_1B_SOA, L_3B_SLG, L_SS_BA, L_SS_SLG, L_SS_RA, L_LF_BA, L_CF_BA, L_RF_SLG, V_P_PITA, V_P_PE, V_C_BA, V_1B_BA, V_1B_RA, V_2B_BA, V_3B_BA, V_SS_SLG, V_CF_SLG, V_RF_SOA

Random Forest (C): PG%, PF, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP

AdaBoost (C): V_P_PITA, V_P_KA, V_1B_SOA

XGBoost (C): PG%, L_P_SOA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_RA, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_SBP, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_RA, L_3B_SOA, L_3B_SBP, L_SS_SLG, L_SS_RA, L_LF_BA, L_LF_SOA, L_LF_SBP, L_CF_SLG, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_RA, V_P_FEP, V_P_PITA, V_P_KA, V_P_ERA, V_P_PE, V_C_BA, V_C_RA, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_BA, V_2B_SLG, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_LF_BA, V_LF_SLG, V_LF_FEP, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_RF_BA, V_RF_RA, V_RF_SB

Regresión Lineal: L_P_KA, L_C_FEP, L_3B_RA, L_CF_FEP, V_P_KA, V_2B_FEP, V_SS_FEP, V_CF_RA

SVR: L_W%, L_P_FEP, L_P_KA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_FEP, L_2B_SOA, L_2B_FEP, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_SOA, V_P_KA, V_P_PE, V_C_RA, V_C_SOA, V_C_FEP, V_1B_FEP, V_2B_RA, V_2B_FEP, V_3B_BA, V_3B_RA, V_3B_FEP, V_SS_FEP, V_LF_FEP, V_CF_RA, V_CF_FEP, V_RF_FEP

Decision Tree (R): V_P_PITA

Random Forest (R): PG%, PF, L_W%, L_P_BA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_SBP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, V_P_SLG, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_SBP, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP

AdaBoost (R): PF, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SBP, L_1B_BA, L_1B_SLG, L_1B_RA, L_2B_BA, L_2B_SOA, L_2B_SBP, L_3B_BA, L_3B_RA, L_3B_SOA, L_3B_SBP, L_SS_BA, L_SS_SLG, L_SS_RA, L_LF_BA, L_LF_SLG, L_LF_SOA, L_LF_FEP, L_CF_RA, L_CF_SOA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, V_P_SLG, V_P_RA, V_P_PITA, V_P_KA, V_P_HAA, V_P_PE, V_C_BA, V_C_RA, V_C_SOA, V_C_SBP, V_1B_SLG, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_3B_SLG, V_3B_RA, V_3B_FEP, V_SS_BA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SBP, V_CF_SLG, V_CF_SOA, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA

XGBoost (R): V_RF_FEP

f) Últimos 10 enfrentamientos

Regresión Logística: L_P_BA, L_1B_BA, L_1B_FEP, L_LF_RA, L_RF_BA, V_1B_BA

SVC: PG%, PF, L_W%, L_P_BA, L_P_SBP, L_P_FEP, L_P_KA, L_C_FEP, L_1B_BA, L_1B_FEP, L_2B_RA, L_3B_RA, L_3B_SOA, L_3B_SBP, L_LF_RA, L_CF_SLG, L_RF_BA, L_RF_RA, L_RF_SOA, V_P_BA, V_P_SLG, V_P_SOA, V_P_HAA, V_1B_BA, V_1B_SOA, V_2B_RA, V_2B_SOA, V_3B_BA, V_3B_SOA

Decision Tree (C): PG%, PF, L_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_SBP, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_SBP, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_LF_FEP

Random Forest (C): PG%, PF, L_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_SBP, L_P_FEP, L_P_PITA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_SBP, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_SBP, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_RA, V_1B_SOA, V_1B_SBP, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SBP, V_RF_FEP

AdaBoost (C): V_LF_SOA

XGBoost (C): PG%, PF, L_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SBP, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_SOA, L_C_SBP, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_SBP, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_RA, V_1B_SOA, V_1B_SBP, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_SLG, V_CF_SOA, V_CF_SBP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP, V_RF_FE

Regresión Lineal: L_C_FEP, V_P_KA

SVR: L_P_KA, L_2B_BA, L_2B_SOA, L_2B_FEP, L_3B_BA, L_LF_BA, L_RF_RA, V_P_BA, V_P_SLG, V_P_RA, V_P_KA, V_3B_RA, V_LF_FEP, V_CF_FEP

Decision Tree (R): L_RF_SOA

Random Forest (R): PG%, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_SLG, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA

AdaBoost (R): L_P_KA, L_C_SOA, L_CF_SOA, V_P_ERA, V_P_PE, V_3B_SOA, V_RF_BA, V_RF_SOA

XGBoost (R): V_RF_SOA

g) Temporada pasada

Regresión Logística: L_W%, V_W%, L_P_KA, L_P_PE, L_3B_RA, L_SS_FEP, V_P_KA, V_P_PE, V_1B_RA, V_RF_RA

SVC: L_W%, V_W%, L_P_KA, L_P_PE, L_3B_RA, V_P_KA, V_P_PE, V_1B_RA, V_2B_BA, V_RF_RA

Decision Tree (C): PG%, L_W%, V_W%, L_P_BA, L_P_RA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_RA, L_C_SOA, L_1B_BA, L_1B_SLG, L_1B_SOA, L_2B_SLG, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_SOA, L_3B_FEP, L_SS_BA, L_SS_SBP, L_LF_SLG, L_LF_SOA, L_LF_FEP, L_CF_BA, L_CF_RA, L_RF_RA, L_RF_SOA, L_RF_FEP, V_P_SLG, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SOA, V_C_FEP, V_1B_SLG, V_1B_SOA, V_2B_BA, V_2B_RA, V_2B_SOA, V_3B_RA, V_3B_SOA, V_3B_SBP, V_SS_RA, V_LF_BA, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_SOA, V_RF_BA, V_RF_RA, V_RF_SOA

Random Forest (C): PG%, L_W%, V_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_FEP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_FEP

AdaBoost (C): L_W%, V_W%, L_P_BA, L_P_PITA, L_P_KA, L_P_HAA, L_P_PE, L_C_BA, L_C_SLG, L_C_FEP, L_1B_SOA, L_1B_SBP, L_2B_BA, L_2B_SLG, L_3B_RA, L_3B_FEP, L_SS_SLG, L_LF_SOA, L_CF_SLG, L_CF_SOA, L_CF_FEP, V_P_SOA, V_P_KA, V_P_HAA, V_P_ERA, V_C_RA, V_1B_SLG, V_1B_RA, V_2B_BA, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_SLG, V_3B_SBP, V_3B_FEP, V_SS_RA, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_RA, V_RF_SLG, V_RF_RA

XGBoost (C): L_W%, V_W%, L_P_SLG, L_P_KA, L_P_ERA, L_P_PE, L_C_BA, L_C_FEP, L_1B_SLG, L_1B_SOA, L_2B_SOA, L_3B_SLG, L_SS_SLG, L_SS_SBP, L_SS_FEP, L_LF_SBP, L_CF_SLG, L_CF_RA, L_CF_SBP, L_RF_RA, L_RF_SOA, V_P_BA, V_P_SLG, V_P_SOA, V_P_KA, V_P_PE, V_C_RA, V_C_SOA, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SBP, V_2B_FEP, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_SOA, V_SS_FEP, V_LF_RA, V_CF_BA, V_CF_SLG, V_CF_SOA, V_RF_SLG, V_RF_SOA, V_RF_FE

Regresión Lineal: L_W%, V_W%, L_P_KA, L_P_PE, L_C_FEP, L_1B_FEP, L_3B_RA, L_CF_FEP, V_P_KA, V_P_PE, V_2B_FEP, V_RF_RA

SVR: L_W%, V_W%, L_P_KA, L_P_PE, L_C_BA, L_1B_SOA, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_SOA, L_3B_RA, L_CF_SLG, V_P_KA, V_P_PE, V_C_RA, V_1B_BA, V_1B_RA, V_2B_SOA, V_2B_FEP, V_LF_RA, V_CF_RA, V_RF_RA

Decision Tree (R): PG%

Random Forest (R): PG%, L_W%, V_W%, L_P_BA, L_P_RA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_RA, L_C_SOA, L_C_FEP, L_1B_BA, L_1B_RA, L_1B_SOA, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_FEP, L_SS_SLG, L_SS_SOA, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_FEP, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_BA, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_2B_BA, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_FEP, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA

AdaBoost (R): PG%, PF, L_W%, V_W%, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_C_SLG, L_C_RA, L_C_SBP, L_C_FEP, L_2B_SOA, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_FEP, L_SS_FEP, L_LF_BA, L_LF_SBP, L_LF_FEP, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_FEP, L_RF_BA, L_RF_SOA, L_RF_SBP, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_RA, V_C_SOA, V_C_FEP, V_1B_FEP, V_2B_SLG, V_2B_SOA, V_2B_FEP, V_3B_RA, V_3B_SOA, V_3B_FEP, V_SS_RA, V_SS_SOA, V_SS_SBP, V_LF_SLG, V_LF_RA, V_LF_SBP, V_LF_FEP, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_FEP, V_RF_SLG, V_RF_SOA

XGBoost (R): V_RF_FEP

h) Histórico

Regresión Logística: L_W%, V_W%, L_P_BA, L_P_SLG, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_C_BA, L_C_RA, L_1B_SLG, L_1B_SOA, L_2B_BA, L_3B_BA, L_SS_SOA, L_LF_SLG, L_CF_BA, L_CF_SLG, L_CF_RA, L_RF_SLG, L_RF_RA, V_P_RA, V_P_FEP, V_P_KA, V_P_ERA, V_P_PE, V_C_RA, V_C_SOA, V_1B_BA, V_1B_RA, V_2B_BA, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_RA, V_SS_BA, V_SS_FEP, V_LF_BA, V_LF_RA, V_LF_SOA, V_CF_SLG

SVC: L_W%, V_W%, L_P_BA, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_C_BA, L_1B_SLG, L_1B_SOA, L_1B_FEP, L_2B_BA, L_3B_BA, L_SS_SOA, L_LF_SLG, L_CF_BA, L_CF_SLG, L_CF_RA, L_RF_SLG, L_RF_RA, V_P_BA, V_P_RA, V_P_FEP, V_P_KA, V_P_ERA, V_P_PE, V_C_RA, V_C_SOA, V_1B_BA, V_1B_RA, V_2B_BA, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_RA, V_SS_BA, V_SS_FEP, V_LF_RA, V_LF_SOA

Decision Tree (C): PG%, V_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_FEP, V_2B_BA, V_2B_RA, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_SBP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP, V_RF_FEP

Random Forest (C): PG%, V_W%, L_P_SLG, L_P_SOA, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_1B_BA, L_1B_RA, L_1B_SOA, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_SLG, L_3B_RA, L_3B_SOA, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA

AdaBoost (C): L_W%, V_W%, L_P_FEP, L_P_KA, L_P_HAA, L_C_SBP, L_1B_BA, L_1B_SLG, L_1B_SOA, L_2B_BA, L_3B_BA, L_3B_SLG, L_SS_SOA, L_LF_RA, L_LF_SOA, L_CF_BA, L_CF_RA, L_RF_SLG, L_RF_RA, L_RF_SBP, V_P_SOA, V_P_FEP, V_P_KA, V_P_ERA, V_C_RA, V_C_SOA, V_1B_RA, V_1B_SBP, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_RA, V_SS_BA, V_SS_SLG, V_LF_BA, V_LF_SOA, V_LF_SBP, V_CF_SLG, V_RF_BA

XGBoost (C): PF, L_W%, V_W%, L_P_BA, L_P_RA, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_SLG, L_C_RA, L_1B_SLG, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_FEP, L_3B_BA, L_3B_RA, L_3B_SOA, L_3B_SBP, L_SS_SLG, L_SS_RA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_CF_SLG, L_CF_RA, L_CF_SOA, L_RF_BA, L_RF_SLG, L_RF_RA, V_P_BA, V_P_SLG, V_P_RA, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_RA, V_C_SOA, V_1B_SLG, V_1B_RA, V_1B_FEP, V_2B_BA, V_2B_SOA, V_2B_FEP, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_SBP, V_CF_SLG, V_CF_SBP, V_CF_FEP, V_RF_SBP, V_RF_FEP

Regresión Lineal: L_W%, V_W%, L_P_KA, L_P_ERA, L_1B_FEP, L_3B_BA, L_CF_BA, L_CF_SLG, L_CF_RA, V_P_RA, V_P_FEP, V_P_KA, V_P_PE, V_C_RA, V_1B_RA, V_2B_BA, V_2B_RA, V_2B_FEP, V_3B_RA, V_LF_RA

SVR: L_W%, V_W%, L_P_BA, L_P_SLG, L_P_FEP, L_P_KA, L_P_HAA, L_P_ERA, L_C_BA, L_C_RA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_2B_BA, L_2B_SLG, L_3B_BA, L_SS_RA, L_SS_SOA, L_LF_BA, L_LF_SOA, L_CF_BA, L_CF_SLG, L_CF_RA, V_P_BA, V_P_RA, V_P_FEP, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_RA, V_C_SOA, V_1B_BA, V_1B_RA, V_2B_BA, V_2B_RA, V_2B_SOA, V_3B_RA, V_SS_BA, V_SS_SOA, V_SS_FEP, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_FEP, V_CF_RA

Decision Tree (R): V_P_KA

Random Forest (R): PG%, L_W%, V_W%, L_P_SLG, L_P_RA, L_P_SOA, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_FEP, L_1B_BA, L_1B_RA, L_1B_SOA, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_3B_BA, L_3B_RA, L_3B_SOA, L_3B_FEP, L_SS_BA, L_SS_RA, L_SS_SOA, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_CF_FEP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_FEP, V_P_SLG, V_P_RA, V_P_SOA, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_FEP, V_2B_BA, V_2B_RA, V_2B_SOA, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SOA, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SBP, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_FEP

AdaBoost (R): PG%, PF, L_W%, V_W%, L_P_BA, L_P_SLG, L_P_RA, L_P_SOA, L_P_SBP, L_P_FEP, L_P_PITA, L_P_KA, L_P_HAA, L_P_ERA, L_P_PE, L_C_BA, L_C_SLG, L_C_RA, L_C_SOA, L_C_SBP, L_C_FEP, L_1B_BA, L_1B_SLG, L_1B_RA, L_1B_SOA, L_1B_SBP, L_1B_FEP, L_2B_BA, L_2B_SLG, L_2B_RA, L_2B_SOA, L_2B_SBP, L_2B_FEP, L_3B_BA, L_3B_RA, L_3B_SOA, L_3B_SBP, L_3B_FEP, L_SS_BA, L_SS_SLG, L_SS_RA, L_SS_SOA, L_SS_SBP, L_SS_FEP, L_LF_BA, L_LF_SLG, L_LF_RA, L_LF_SOA, L_LF_SBP, L_LF_FEP, L_CF_BA, L_CF_SLG, L_CF_RA, L_CF_SOA, L_CF_SBP, L_RF_BA, L_RF_SLG, L_RF_RA, L_RF_SOA, L_RF_SBP, L_RF_FEP, V_P_BA, V_P_SLG, V_P_RA, V_P_SOA, V_P_SBP, V_P_FEP, V_P_PITA, V_P_KA, V_P_HAA, V_P_ERA, V_P_PE, V_C_BA, V_C_SLG, V_C_RA, V_C_SOA, V_C_SBP, V_C_FEP, V_1B_BA, V_1B_SLG, V_1B_RA, V_1B_SOA, V_1B_SBP, V_1B_FEP, V_2B_BA, V_2B_SLG, V_2B_RA, V_2B_SOA, V_2B_SBP, V_2B_FEP, V_3B_BA, V_3B_SLG, V_3B_RA, V_3B_SOA, V_3B_SBP, V_3B_FEP, V_SS_BA, V_SS_SLG, V_SS_RA, V_SS_SOA, V_SS_SBP, V_SS_FEP, V_LF_BA, V_LF_SLG, V_LF_RA, V_LF_SBP, V_LF_FEP, V_CF_BA, V_CF_SLG, V_CF_RA, V_CF_SOA, V_CF_SBP, V_CF_FEP, V_RF_BA, V_RF_SLG, V_RF_RA, V_RF_SOA, V_RF_SBP, V_RF_FEP

XGBoost (R): V_RF_FEP