Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

SISTEMA de PREDICCIÓN de RESULTADOS DEPORTIVOS BASADO en APRENDIZAJE AUTOMÁTICO

Autor: SERGIO DEL CASTILLO AGUDO
Tutor: JAVIER DE LOPE ASIAÍN

Madrid, Enero 2026

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: SISTEMA de PREDICCIÓN de RESULTADOS DEPORTIVOS BASADO en APRENDIZAJE AUTOMÁTICO

Enero 2026

Autor: SERGIO DEL CASTILLO AGUDO
Tutor: JAVIER DE LOPE ASIAÍN
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

En este Trabajo de Fin de Grado se aborda la predicción del resultado de partidos de la *NBA* mediante técnicas de aprendizaje automático, con el objetivo de estimar la probabilidad de victoria del equipo local a partir de datos históricos de partidos y estadísticas agregadas de los jugadores. La motivación principal del proyecto radica en analizar hasta qué punto modelos de *machine learning* relativamente modestos, entrenados con datos públicos, son capaces de realizar predicciones con un grado de acierto aceptable.

La predicción de resultados deportivos ha sido históricamente estudiada mediante enfoques estadísticos clásicos y, más recientemente, mediante modelos de aprendizaje automático y aprendizaje profundo. En particular, en este proyecto se abordan los modelos basados en árboles de decisión ensamblados, como *XGBoost*, y las redes neuronales recurrentes, como las *LSTM*, por su capacidad para modelar relaciones no lineales y dependencias temporales. El presente trabajo se centra en un escenario realista, utilizando únicamente datos accesibles públicamente y un conjunto de características cuidadosamente seleccionado.

El proyecto se estructura en varias fases. En primer lugar, se realiza un proceso de recolección, limpieza y transformación de datos históricos de partidos de la *NBA*, generando un conjunto de características representativas del rendimiento de los equipos. A partir de estos datos, se implementan dos enfoques para la predicción. El primero consiste en redes neuronales *LSTM* que aprovechan la naturaleza secuencial y cronológica de los partidos. El segundo enfoque se basa en el algoritmo *XGBoost*, que modela el problema como una tarea de clasificación binaria sobre datos tabulares.

Los modelos se entrenan y evalúan siguiendo una praxis común, con una división temporal de los datos en conjuntos de entrenamiento, validación y prueba. Finalmente, se comparan los resultados obtenidos por ambos enfoques, analizando no solo su precisión, sino también otras métricas y valores representativos del rendimiento de los modelos. Los resultados muestran que, aun con un conjunto de datos limitado en comparación con sistemas más complejos, es posible superar una predicción base y obtener estimaciones razonablemente aceptables del resultado de los partidos. Asimismo, el trabajo pone de manifiesto la importancia crítica de la selección de datos y su procesamiento, y deja abiertas diversas líneas de mejora para trabajos futuros.

Abstract

This Final Degree Project addresses the prediction of *NBA* match results using machine learning techniques, with the aim of estimating the probability of victory for the home team based on historical match data and aggregate player statistics. The main motivation for the project is to analyse the extent to which relatively modest machine learning models, trained with public data, are capable of making predictions with an acceptable degree of accuracy.

The prediction of sports results has historically been studied using classical statistical approaches and, more recently, using machine learning and deep learning models. In particular, this project addresses models based on assembled decision trees, such as *XGBoost*, and recurrent neural networks, such as *LSTMs*, due to their ability to model non-linear relationships and temporal dependencies. This work focuses on a realistic scenario, using only publicly available data and a carefully selected set of features.

The project is structured in several phases. First, historical *NBA* game data is collected, cleaned and transformed, generating a set of characteristics representative of team performance. Based on this data, two approaches to prediction are implemented. The first consists of *LSTM* neural networks that exploit the sequential and chronological nature of the games. The second approach is based on the *XGBoost* algorithm, which models the problem as a binary classification task on tabular data.

The models are trained and evaluated following a common practice, with a temporal division of the data into training, validation and test sets. Finally, the results obtained by both approaches are compared, analysing not only their accuracy, but also other metrics and values representative of the models' performance. The results show that, even with a limited dataset compared to more complex systems, it is possible to exceed a baseline prediction and obtain reasonably acceptable estimates of match results. The work also highlights the critical importance of data selection and processing, and leaves open several avenues for improvement in future work.

Tabla de contenidos

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La predicción de resultados deportivos se ha convertido en un asunto cada vez más importante y de mayor interés para la sociedad debido a sus aplicaciones en diversas áreas, como el análisis de datos deportivos, la toma de decisiones deportivas en tiempo real, las apuestas deportivas y las aplicaciones de entretenimiento, con gran acogida entre los fanáticos, como la aplicación para móvil *NBA Fantasy*.

En deportes de equipo, como el baloncesto, el resultado de un partido está influido por un amplio conjunto de variables. Entre ellas, se encuentran factores como el estado de forma del equipo, jugar como local o visitante, el acierto de los jugadores en ese partido, las decisiones técnicas tomadas, el estado psicológico y físico de los jugadores, etc. Además, muchas de estas variables, como la forma del equipo y de los jugadores, evolucionan a lo largo de la temporada, lo que introduce una clara dependencia temporal entre los partidos disputados y el momento en que se realiza la predicción.

Existen variables cuyos valores son prácticamente imposibles de conocer con precisión, como el estado anímico de un jugador y el impacto que puede tener en un partido. Puede haber jugadores que, a pesar de tener un bajo estado anímico, jueguen razonablemente bien y otros que no lo hagan. Esta incertidumbre, junto con la gran cantidad de variables y sus dependencias temporales, no se puede capturar mediante modelos clásicos, lo que deriva en que estos modelos no sean fiables para la predicción y en la búsqueda de modelos más sofisticados.

Los primeros enfoques para predecir resultados se basaban en modelos estadísticos tradicionales y en el uso de un conjunto finito de métricas básicas. Sin embargo, el auge en la medición de datos de los partidos y de los jugadores permite usar métricas más complejas que son capaces de capturar mejor las dependencias dentro de un partido. No obstante, las estadísticas clásicas como los puntos, rebotes y asistencias siguen siendo de gran utilidad y fundamentales para el cálculo de estas estadísticas más complejas.

En los últimos años, las técnicas de *machine learning* han ganado popularidad debido a su habilidad para aprender patrones relevantes a partir de los datos, sin necesidad de definir manualmente relaciones entre las variables. En parti-

cular, los modelos basados en redes neuronales recurrentes permiten capturar dependencias temporales, lo que resulta especialmente adecuado para el análisis de secuencias de partidos a lo largo de una temporada.

1.1. Objetivo Principal

Este proyecto se centra en predecir el ganador de un partido de baloncesto de la NBA apoyándose en redes neuronales recurrentes *Long Short-Term Memory* y en el algoritmo de *machine learning* XGBoost, entrenados a partir de datos históricos de partidos y de datos históricos de jugadores de la *NBA* de dominio público.

1.2. Objetivos Específicos

Además del objetivo general del proyecto, se establecen los siguientes objetivos específicos con el fin de estructurar y guiar el desarrollo del trabajo:

- Analizar el estado del arte en la predicción de resultados deportivos, con especial atención a los modelos utilizados en la *NBA*.
- Estudiar y conocer las estadísticas de baloncesto empleadas actualmente para la predicción del resultado de un encuentro de la *NBA*.
- Desarrollar modelos de aprendizaje automático competentes capaces de predecir con un grado de acierto razonable.

1.3. Estructura del Documento

En cuanto a la estructura del documento, está dividido en seis capítulos:

1. Capítulo 1 introduce el objetivo del proyecto y el contexto en el que se encuentra.
2. Capítulo 2 presenta el estado del arte actual para las predicciones deportivas, haciendo especial énfasis en las relativas a la *NBA*.
3. Capítulo 3 explica los fundamentos teóricos básicos de los modelos utilizados para la predicción.
4. Capítulo 4 muestra la metodología empleada para lograr el objetivo del proyecto. Se explican las herramientas empleadas y los distintos métodos seguidos. Abarca desde la recolección de los datos hasta la obtención de los resultados.
5. Capítulo 5 analiza los resultados obtenidos en cada uno de los modelos implementados.
6. Capítulo 6 reflexiona acerca del trabajo realizado y de futuras vías de mejora.

Capítulo 2

Estado del Arte

En este capítulo se aborda el estado del arte de las predicciones deportivas, en especial de la *NBA*. Se tratan las aplicaciones de las predicciones en el mundo de la *NBA*, los métodos para predecir en la actualidad e históricamente, y las métricas para medir su acierto y validez.

2.1. Usos de las Predicciones Deportivas en la *NBA*

En la última década, el aumento de estadísticas medibles y el uso de grandes cantidades de datos, junto al avance en los modelos de aprendizaje automático, han cambiado el enfoque de la predicción deportiva, especialmente en la *NBA*.
El principal uso que la mayoría de las personas atribuyen a las predicciones deportivas son las apuestas deportivas, debido a la gran expansión y a la explosión económica que han supuesto tanto en España y en Europa [1] como en Estados Unidos [2]. Sin embargo, también presentan gran importancia en el análisis de rendimiento de jugadores y equipos, *scouting* y ajustes tácticos por parte de entrenadores en tiempo real durante partidos [3]. Últimamente, su utilidad se ha visto incrementada también con juegos y aplicaciones que permiten al usuario ser el director ejecutivo de un equipo y dirigir una plantilla de jugadores compitiendo contra otros usuarios, como el *NBA Fantasy* [4].

2.2. Modelos Clásicos

La predicción deportiva se ha basado históricamente en modelos estadísticos clásicos, como la regresión lineal, la regresión logística, los modelos basados en la distribución de Poisson y los árboles de decisión. A pesar de la simplicidad de algunos de estos métodos, ofrecen información útil y sirven como apoyo para métodos más avanzados y modernos.

Uno de los métodos más utilizados y conocidos en la predicción deportiva es la regresión lineal, que nos permite relacionar linealmente variables continuas predictoras con una variable objetivo, también continua, que suele ser la puntuación del partido. La simplicidad y la fácil interpretabilidad de este método

son algunos de los factores de su amplio uso. En este método, se asume que las variables independientes afectan de forma lineal a la variable objetivo o resultado. Los estudios que emplean regresión lineal han observado correlaciones entre estadísticas individuales, como puntos, asistencias y rebotes, y el resultado final de un partido. Sin embargo, su capacidad es limitada. No es útil para modelar interacciones complejas y relaciones no lineales, lo que en escenarios con múltiples factores que interactúan entre sí de formas diferentes y complejas, como puede ser un partido de la *NBA*, no resulta la opción más adecuada por sí sola [5].

La regresión logística es otro método utilizado como base para modelos más avanzados. Este método estima la probabilidad de la ocurrencia de un evento discreto mediante la función *sigmoide*,

$$f(x) = \frac{1}{1 + e^{-x}}$$

que mapea valores reales a probabilidades entre 0 y 1. En el ámbito deportivo de la *NBA*, se utiliza mayoritariamente para predecir resultados binarios, como la victoria o la derrota de un equipo. Este método presenta variantes, como el modelo logístico asimétrico, que introduce un parámetro adicional para permitir una curva *sigmoide* no simétrica. Esta modificación resulta útil cuando el impacto de las variables predictoras sobre la probabilidad del evento no es uniforme a lo largo del dominio, lo que puede reflejar de una forma más realista comportamientos en contextos deportivos más complejos [6]. Debido a esto, la regresión logística se sigue utilizando en modelos más complejos como capa final para generar una salida probabilística binaria.

Otra familia destacada son los modelos basados en la distribución de Poisson, que se refiere al número de veces que ocurre un evento en un intervalo fijo de tiempo o espacio, siempre que los eventos ocurran de manera aleatoria e independiente, con una tasa promedio de aparición constante. En el contexto del baloncesto, la distribución de Poisson se ha utilizado para describir la anotación de un partido, asumiendo que las canastas son eventos aleatorios e independientes. Sin embargo, estudios recientes como [7] muestran que, aunque la distribución de Poisson describe bien la mayoría de las situaciones, en los finales de partidos igualados surgen dinámicas más complejas que no es capaz de predecir correctamente. Esto sugiere que la distribución de Poisson es un buen modelo de referencia, pero que la complejidad del juego en momentos críticos requiere enfoques más flexibles.

Por último, los árboles de decisión. Su estructura jerárquica se asemeja a un diagrama de flujo, donde cada nodo interno representa una prueba sobre una variable, cada rama un posible estado de esa variable y cada nodo hoja un resultado final o clasificación. Su principal ventaja es la interpretabilidad, ya que permite extraer reglas claras sobre qué factores conducen a la victoria. En deportes como el baloncesto, pueden emplearse para clasificar ganadores o predecir márgenes de victoria a partir de estadísticas de equipo, identificando variables clave como pérdidas de balón o puntos anotados [8]. No obstante, un único árbol tiende a tener problemas de sobreajuste, lo que impide que realice predicciones

satisfactorias con datos nuevos. Para solucionar estas limitaciones, se emplean modelos de *ensambles*, como puede ser *XGBoost*.

En resumen, los métodos clásicos han sido muy utilizados durante décadas para la predicción deportiva. Actualmente, no han caído en desuso, sino que sirven como base de modelos más sofisticados o como complemento dentro de estos últimos.

2.3. Modelos Modernos

Debido al aumento de la disponibilidad de los datos deportivos y a la complejidad de estos, han surgido modelos modernos que utilizan métodos de aprendizaje automático y aprendizaje profundo que superan las limitaciones de los modelos clásicos. Dentro de estos métodos, las redes neuronales profundas, en particular las arquitecturas de redes neuronales recurrentes como las *Long Short-Term Memory (LSTM)*, junto con algoritmos avanzados de *ensamble* como *eXtreme Gradient Boosting (XGBoost)*, destacan por su buen rendimiento en la predicción de resultados deportivos.

Las redes neuronales profundas *(Deep Learning)* emplean múltiples capas de neuronas artificiales para aprender patrones de los datos. En particular, las *LSTM* han demostrado un rendimiento superior en la predicción de resultados deportivos de la *NBA*. Según [9], el uso de las *LSTM* permite capturar dependencias temporales y patrones complejos en los datos de tiros. Esta arquitectura es especialmente útil en el contexto de la *NBA*, donde el desempeño de un equipo o jugador en partidos anteriores influye en los resultados futuros. Sin embargo, su implementación requiere una mayor cantidad de datos, recursos computacionales y una cuidadosa selección de variables espaciales y temporales. Además, estudios recientes han comparado las redes *LSTM* con otras arquitecturas modernas, como los modelos *Transformer*, confirmando que las redes *LSTM* siguen siendo una opción robusta y competitiva para tareas de predicción secuencial en el ámbito deportivo [10].

En el ámbito del aprendizaje automático, los métodos *ensemble* (o de conjunto) [11] se han consolidado como una de las estrategias más eficaces para la predicción de resultados deportivos. Estos métodos combinan múltiples modelos base, habitualmente árboles de decisión, con el objetivo de mejorar la precisión y la capacidad de generalización del modelo final. Entre las principales técnicas de *ensemble* se encuentran el *bagging* (como en *Random Forest*) y el *boosting*.

El *bagging* consiste en entrenar varios modelos independientes sobre diferentes muestras del conjunto de datos y luego combinar sus resultados, normalmente realizando un promedio. De esta forma, se reduce el riesgo de que el modelo dependa demasiado de los datos de entrenamiento.

Por otro lado, el *boosting* entrena los modelos uno tras otro, de manera que cada nuevo modelo se centra en los errores que cometieron los modelos anteriores. Así, el conjunto final aprende de sus propios fallos y logra mejorar la precisión de las predicciones.

Dentro de este último enfoque se encuentran algoritmos como *AdaBoost*, *Gradient Boosting* y su versión optimizada, *XGBoost (eXtreme Gradient Boosting)*. Este último ha ganado gran popularidad por su rapidez, eficiencia y rendimiento en el manejo de grandes volúmenes de datos heterogéneos. En el contexto de la *NBA*, diversos estudios han demostrado que los modelos basados en *boosting*, especialmente *XGBoost*, superan a los modelos lineales tradicionales en la predicción de resultados y puntuaciones, obteniendo mejores valores en métricas como *AUC-ROC*, *F1 Score*, *Accuracy* y *Recall* [12].

En definitiva, los métodos modernos de aprendizaje profundo y *ensemble* constituyen el estado del arte en la predicción deportiva de la *NBA* en la actualidad. Estos métodos no solo presentan mayor precisión en sus predicciones que los modelos clásicos, sino que permiten que las predicciones tengan aplicaciones en tiempo real.

Capítulo 3

Fundamentos Teóricos

En este apartado se tratarán los fundamentos teóricos de los modelos de aprendizaje automático *LSTM* y *XGBoost*.

3.1. Funcionamiento Redes Neuronales Artificiales

Para entender las redes neuronales recurrentes, como las *LSTM*, primero debemos comprender las redes neuronales, por qué existen y otros conceptos clave de estas.

Una red neuronal artificial es un modelo computacional que pretende simular el funcionamiento del cerebro humano. Está formada por neuronas artificiales que reciben entradas, las ponderan con pesos (valores que determinan la importancia de cada entrada) y añaden un sesgo que permite ajustar la salida de la neurona para mejorar la capacidad de representación del modelo. Con estas entradas ponderadas y el sesgo, se realiza una operación matemática que depende de la función de activación elegida, y esta función produce la salida de la neurona [13]. Las funciones de activación son las que permiten que la red neuronal aprenda patrones complejos. Hay distintos tipos de funciones de activación, como la función *TanH*, la función *ReLu*, o la función *sigmoide*. Matemáticamente, la salida y de una neurona se puede representar como:

$$y = f(w_1x_1 + w_2x_2 + ... + w_nx_n + b)$$

donde w_i son los pesos de entrada, x_i son las entradas, b es el sesgo y f es la función de activación. Una función de activación muy utilizada es la *sigmoide*:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Por tanto, la salida de la neurona sería:

$$y = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + ... + w_nx_n)}}$$

Esta función transforma la suma ponderada en un valor entre 0 y 1, lo que resulta útil para tareas de clasificación binaria.

Las neuronas se organizan en capas: la capa de entrada, una o varias capas ocultas y la capa de salida.

Para que la red pueda aprender, se requiere optimización, que se refiere al proceso de ajustar los parámetros (pesos y sesgos) para mejorar el rendimiento. El método más común para esta optimización es el descenso del gradiente, que ajusta los parámetros en la dirección que reduce el error. Este proceso se realiza mediante el algoritmo de retropropagación, que calcula el gradiente del error respecto a cada peso de la red y actualiza los valores según:

$$w := w - \eta \frac{\partial E}{\partial w}$$

donde η es la tasa de aprendizaje (*learning rate*) y E es la función de error.

La tasa de aprendizaje es un hiperparámetro que determina el tamaño del paso que damos en cada iteración del descenso del gradiente. Si la tasa es muy alta, la red puede no converger, saltándose el mínimo del error y generando oscilaciones. Si es muy baja, el aprendizaje será muy lento y puede quedarse atrapada en mínimos locales de la función.

Las redes pueden ser unicapa (una sola capa oculta) o multicapa (varias capas ocultas). Las redes unicapa solo pueden resolver problemas linealmente separables, mientras que las multicapa pueden abordar problemas más complejos gracias a la no linealidad introducida por las funciones de activación en las capas ocultas.

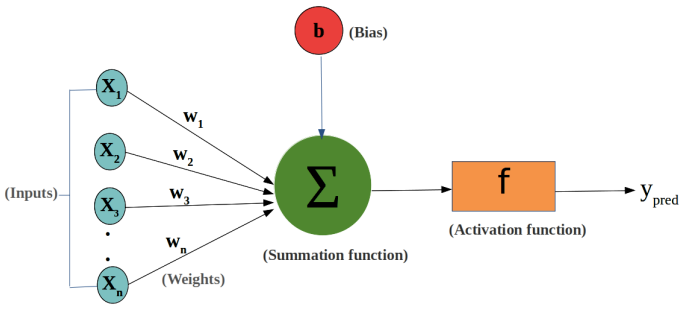

Figura 3.1: Red Neuronal

3.2. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (*RNN*) son un tipo de red neuronal artificial diseñada para procesar y modelar datos secuenciales o temporales. A diferencia de las redes neuronales tradicionales, que tratan las entradas y las salidas como datos independientes, las *RNN* incorporan conexiones recurrentes que permiten que la salida de una neurona en un instante se utilice como entrada en el siguiente. Esto les otorga una memoria temporal que las hace ideales para aplicaciones donde el contexto y el orden de la información son importantes. Sin embargo, las *RNN* clásicas presentan dificultades para aprender dependencias a largo plazo, debido al problema del desvanecimiento o explosión del gradiente durante el entrenamiento. El problema del desvanecimiento del gradiente ocurre cuando los gradientes se vuelven extremadamente pequeños o grandes al propagarse a través de muchas capas temporales. Como consecuencia, la red tiene dificultades para ajustar correctamente sus pesos y pierde la capacidad de recordar información de instantes lejanos en el tiempo.

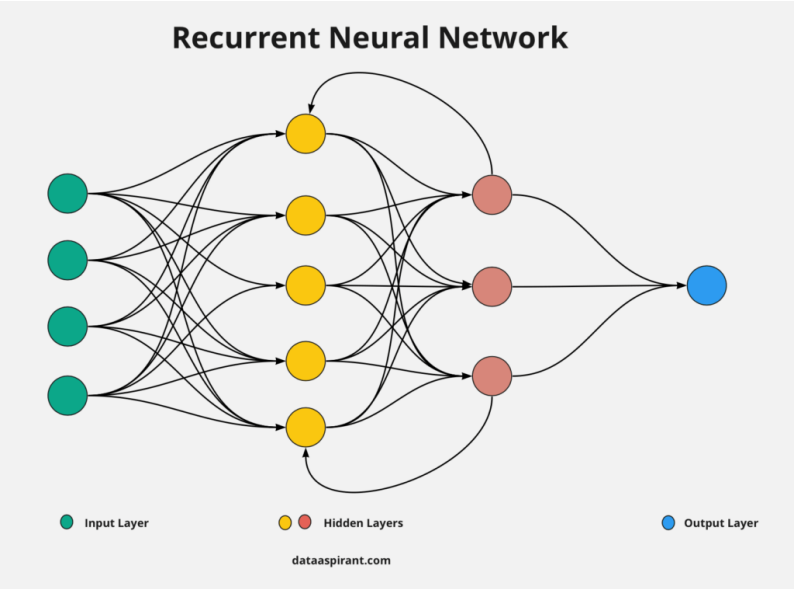

Figura 3.2: Red Neuronal Recurrente

Las redes *Long Short-Term Memory (LSTM)* son un tipo especial de *RNN* que están diseñadas para superar el problema del desvanecimiento del gradiente, por lo que pueden recordar información a largo plazo. Las *LSTM* introducen una estructura interna de celdas de memoria y tres puertas, una de entrada, una de salida y una de olvido que regulan el flujo de información, permitiendo mantener y utilizar información relevante a lo largo de secuencias largas.

- La puerta de entrada decide cuánta información nueva se almacena en la celda.
- La puerta de olvido controla qué información previa debe descartarse, evitando la acumulación de datos irrelevantes.

- La puerta de salida determina qué parte de la información almacenada se utiliza para generar la salida actual.

De esta forma, cada celda *LSTM* mantiene dos estados: el estado de celda (*cell state*), que actúa como una memoria a largo plazo, y el estado oculto (*hidden state*), que funciona como memoria a corto plazo. Estos estados se actualizan dinámicamente en cada paso temporal, permitiendo que la red conserve y utilice información relevante de eventos pasados mientras procesa nuevas entradas. Las principales ventajas de las *LSTM* son su habilidad para recordar información durante largos períodos de tiempo y su eficacia en manejar secuencias complejas con dependencias a largo plazo, por lo que son muy útiles en predicciones dadas una secuencia temporal.

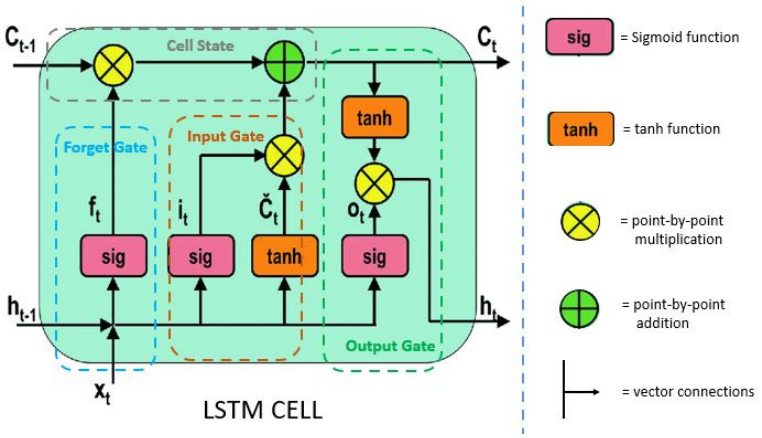

Figura 3.3: Red Neuronal Recurrente Long Short-Term Memory

3.3. Algoritmo *XGBoost*

El *eXtreme Gradient Boosting (XGBoost)* es un algoritmo de aprendizaje automático supervisado que se utiliza para tareas de clasificación y regresión. Se basa en la técnica de *boosting*, que consiste en combinar múltiples modelos débiles para crear un modelo fuerte y de alto rendimiento. En particular, *XGBoost* emplea árboles de decisión como modelos débiles que se entrenan secuencialmente; cada nuevo árbol intenta corregir los errores residuales de los árboles anteriores, es decir, la diferencia entre los valores predichos y los valores reales, mejorando así la precisión del modelo final [12]. El funcionamiento de *XGBoost* puede resumirse en varios pasos:

1. Se realiza una predicción inicial, normalmente una media o un valor constante.

2. Se calculan los errores residuales, que representan la diferencia entre los valores reales y las predicciones.

3. Se entrena un nuevo árbol de decisión para corregir los errores del modelo actual, con el objetivo de mejorar sus predicciones.

4. Las predicciones de este nuevo árbol se combinan con las anteriores mediante una tasa de aprendizaje o *learning rate*, que controla cuánto contribuye cada árbol al modelo final.
5. Este proceso se repite iterativamente hasta alcanzar un número óptimo de árboles o hasta que el error deje de mejorar significativamente.

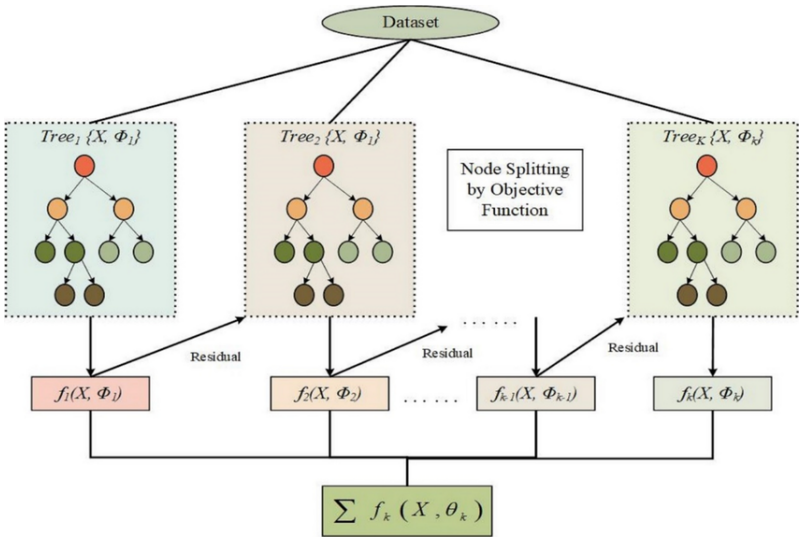

Figura 3.4: Ensemble: XGBoost

3.4. División del Conjunto de Datos

En estos modelos de *machine learning*, el conjunto de datos empleado se suele dividir de la siguiente forma para conseguir el mejor rendimiento posible. El *dataset* se fracciona en tres fases: entrenamiento, validación y evaluación o test. Durante la fase de entrenamiento, el modelo se ajusta utilizando exclusivamente el conjunto de datos de entrenamiento, aprendiendo los patrones presentes en los datos y actualizando los pesos y sesgos de la red. De forma paralela, el conjunto de validación se emplea para supervisar el rendimiento del modelo sobre datos no vistos durante el entrenamiento. Este conjunto resulta clave para el ajuste de hiperparámetros, como la tasa de aprendizaje, y para detectar posibles problemas de sobreajuste. Finalmente, una vez concluido el proceso de entrenamiento, el conjunto de prueba se utiliza para evaluar el rendimiento del modelo frente a datos completamente inéditos, proporcionando una estimación realista de su capacidad de generalización.

3.5. Evaluación de Modelos

A la hora del uso de modelos, es importante saber cómo medir el rendimiento, es decir, la precisión y la robustez de los modelos. Para ello, se emplean una serie de métricas a modo de estándar [14]. Estas métricas varían según el tipo de predicción, distinguiendo principalmente entre problemas de clasificación (por ejemplo, predecir ganador/perdedor) y regresión (por ejemplo, predecir puntuación o diferencia de puntos).
Para comprender las métricas, es importante conocer ciertos términos:

- Verdaderos Positivos: predicciones correctas de la clase positiva. En este caso, se predice que el local gana y se produce la victoria del local.
- Falsos Positivos: predicciones incorrectas de la clase positiva. El modelo predice que gana el local, pero realmente vence el visitante.
- Verdaderos Negativos: predicciones correctas de la clase negativa. En este caso, se predice que no gana el local y resulta vencedor el visitante.
- Falsos Negativos: predicciones incorrectas de la clase negativa. El modelo predice que no gana el local y, en realidad, sí gana.

Entre las métricas más comunes de clasificación se encuentran:

Accuracy: Es la proporción general de predicciones correctas sobre el total de predicciones realizadas por el modelo, tanto para las clases positivas como para las negativas.

$$\frac{AciertosTotales}{TotalPredicciones}$$

Recall (sensibilidad): Mide la proporción de verdaderos positivos sobre la suma de verdaderos positivos y falsos negativos. Es decir, de todos los verdaderos positivos, cuántos acertó el modelo.

$$\frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosNegativos}$$

Precisión: Mide la proporción de predicciones positivas correctas (verdaderos positivos) entre todas las instancias que el modelo clasificó como positivas (verdaderos positivos + falsos positivos).

$$\frac{VerdaderosPositivos}{VerdaderosPositivos + FalsosPositivos}$$

Area Under the ROC Curve (AUC-ROC): Mide la capacidad del modelo para separar correctamente las clases en todos los posibles umbrales de clasificación, siendo útil para evaluar modelos binarios.

F1-Score: Se calcula como la media armónica de la precisión y el *recall*, lo que significa que el resultado se ve fuertemente penalizado si una de las dos métricas es muy baja.

$$(F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall})$$

Para tareas de regresión, se emplean métricas que cuantifican el error entre predicciones y valores reales, tales como:

Coeficiente de determinación R (R^2): Qué proporción de la varianza de la variable dependiente (real) puede ser explicada por el modelo. Es decir, con qué exactitud sigue el modelo los datos reales.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- y_i = valor real
- $\hat{y}_i$ = valor predicho
- $\bar{y}_i$ = media de los valores reales
- $R^2 = 1$, predicción perfecta
- $R^2 = 0$, el modelo no mejora respecto a usar solo la media
- $R^2 < 0$, el modelo es peor que una predicción constante

Por ejemplo, si estás prediciendo puntos anotados, un $R^2 = 0{,}85$ implica que el modelo explica el 85 % de la variabilidad en la anotación total de los equipos.

Mean Absolute Error (MAE): Mide el promedio del error absoluto entre las predicciones y los valores reales.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Por ejemplo, si el modelo predice una media de 3 puntos de error por partido (*MAE* = 3), significa que las predicciones de anotación se desvían en promedio 3 puntos del resultado real.

Root Mean Square Error(RMSE): Similar al *MAE*, pero penaliza más los errores grandes, ya que eleva al cuadrado las diferencias antes de promediarlas.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Por ejemplo, si predices los puntos totales y obtienes, *MAE* = 3.2, *RMSE* = 5.1. El modelo predice bien en general, pero falla bastante en algunos partidos atípicos.

Mean Absolute Percentage Error (MAPE): Expresa el error en porcentaje respecto al valor real.Indica qué porcentaje promedio se desvía la predicción del valor real.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Si obtienes un *MAPE* = 4 %, significa que el modelo predice con un error medio del 4 % respecto al valor real de puntos anotados o margen de victoria.

Capítulo 4

Metodología

En este apartado se tratarán las herramientas utilizadas, las decisiones tomadas y los métodos empleados y seguidos durante el proyecto.
Este ha sido el flujo de trabajo seguido para la realización del proyecto:

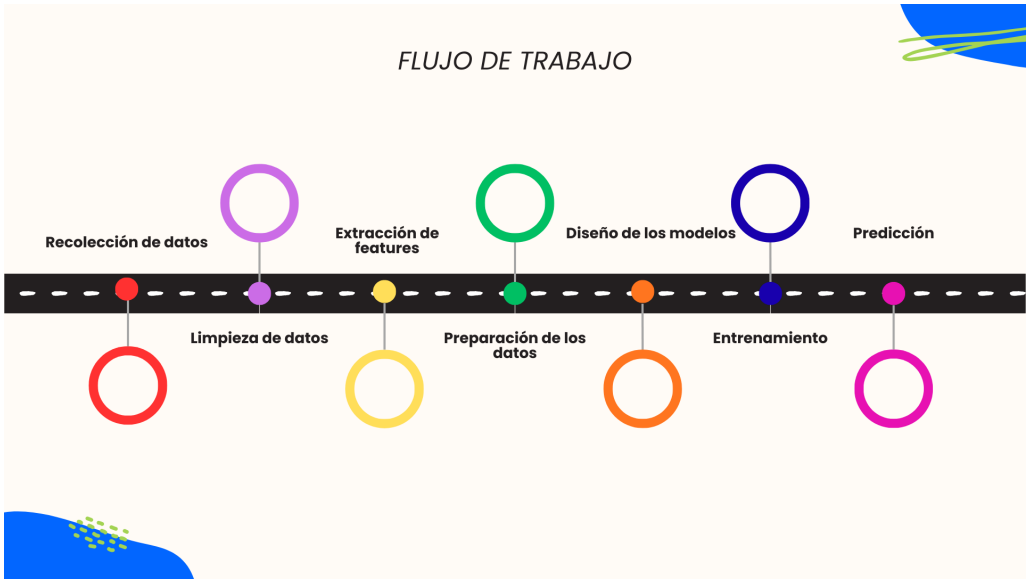

Figura 4.1: Flujo de Trabajo

Entorno y Herramientas Empleadas

Antes de abordar el flujo de trabajo y los procedimientos seguidos, resulta fundamental describir el entorno de trabajo y las herramientas empleadas a lo largo del proyecto, ya que estas han condicionado tanto el diseño como la implementación de las soluciones propuestas.

El lenguaje de programación escogido ha sido `Python`, debido a su gran integración con librerías y paquetes de *machine learning*. Además, es el lenguaje que permite ejecutar el entorno de desarrollo escogido, `Google Colaboratory`

(Google Colab). Este entorno es una plataforma basada en la nube que permite ejecutar código en Python a través de cuadernos interactivos. Además, ofrece varias ventajas relevantes para este tipo de proyectos, entre las que destacan la facilidad de uso, la integración con Google Drive y, especialmente, el acceso gratuito a recursos hardware acelerados como *GPU* y *TPU*. Esto permite entrenar modelos de aprendizaje profundo de forma más eficiente que en un entorno local, reduciendo significativamente los tiempos de entrenamiento y facilitando la experimentación con distintas arquitecturas y configuraciones.

Para la implementación de los modelos de redes neuronales se ha utilizado la librería PyTorch, un *framework* de aprendizaje profundo ampliamente utilizado. En cambio, para la implementación del algoritmo de *XGBoost*, se ha utilizado la librería xgboost. Ambas librerías son fácilmente integrables en el entorno Colab.

Además, se han utilizado diversas librerías complementarias del ecosistema Python para el tratamiento y análisis de los datos. Entre ellas destacan nba_api para la obtención de datos oficiales de la *NBA*, NumPy y Pandas para la manipulación eficiente de los datos obtenidos, Matplotlib para la creación de gráficos de barras, TensorBoard para la visualización de resultados y métricas de las redes neuronales, y scikit-learn para tareas auxiliares como la normalización de variables o el cálculo de métricas de evaluación.

Diseño y Organización de los Modelos

Para facilitar el entendimiento de los métodos empleados en el proyecto, se dedica este espacio a explicar las decisiones tomadas respecto a los modelos implementados.

Como se explica en los apartados siguientes, se recogen datos de partidos y de jugadores. En el proyecto, se realiza la implementación de una red neuronal recurrente *LSTM* entrenada únicamente a partir de los datos de partidos, red 1, y otra entrenada con datos de partidos y jugadores, red 2. Debido a que la red 1 presenta mejores resultados, como se verá posteriormente, se ha decidido probar esta red con varias configuraciones distintas para analizar y comprobar su funcionamiento en profundidad con cada configuración. Finalmente, se opta por quedarse con las dos configuraciones de la red 1 que muestran mejor rendimiento. Desde ahora, en el documento, se hará referencia a estas redes *LSTM* como red 1, configuración 1, y red 1, configuración 2.

Asimismo, debido al bajo rendimiento mostrado por la red 2, entrenada con datos de partidos y jugadores, se ha decidido realizar una implementación del algoritmo *XGBoost* con estos mismos datos para evaluar su rendimiento respecto a la red 2.

4.1. Recolección de Datos

En la predicción de resultados de partidos de la *NBA*, hay una amplia cantidad de datos disponibles debido al gran número de estadísticas que se miden en cada partido. Evidentemente, el uso de unos datos u otros depende de la intención de la predicción. En este caso, se emplean los que ayudan a predecir ganadores y perdedores en un partido [15]. Para ello, hay dos tipos de datos a diferenciar, los del equipo y los del jugador individualmente en un partido.
Habitualmente, los siguientes son los datos más relevantes que se miden dentro de un partido de la *NBA*.

Relativos a los equipos:

- Tiros libres intentados durante el partido (FTA)
- Tiros libres encestados durante el partido (FTM)
- Tiros de campo intentados durante el partido (FGA)
- Tiros de campo encestados durante el partido (FGM)
- Tiros de tres intentados durante el partido (FG3A)
- Tiros de tres encestados durante el partido (FG3M)
- Número de rebotes ofensivos (OREB)
- Número de rebotes defensivos (DREB)
- Número de asistencias (AST)
- Número de robos (STL)
- Número de tapones (BLK)
- Número de pérdidas (TOV)
- Número de faltas personales (PF)
- Puntos totales del equipo (PTS)

Relativos a los jugadores:

- Tiros libres intentados durante el partido (FTA)
- Tiros libres encestados durante el partido (FTM)
- Tiros de campo intentados durante el partido (FGA)
- Tiros de campo encestados durante el partido (FGM)
- Tiros de tres intentados durante el partido (FG3A)
- Tiros de tres encestados durante el partido (FG3M)
- Número de rebotes ofensivos (OREB)
- Número de rebotes defensivos (DREB)

- Número de asistencias (AST)
- Número de robos (STL)
- Número de tapones (BLK)
- Número de pérdidas (TOV)
- Número de faltas personales (PF)
- Puntos totales (PTS)
- Minutos jugados (MP)
- El *Plus-Minus*. Esta estadística se calcula restando a los puntos anotados por el equipo cuando el jugador está en pista los puntos recibidos por el equipo en ese mismo periodo.

Con estos datos, se pueden calcular posteriormente otras estadísticas más útiles y valiosas para los modelos de *machine learning*, como se verá más adelante. Un ejemplo es la estadística avanzada *Net Rating*.

Con el propósito de recoger estos datos para entrenar los modelos, se utiliza la librería de `Python` llamada `nba_api`. Esta librería recoge los datos de la web oficial de la *NBA*, *NBA.com*, realizando *web scraping*. Es la herramienta más recomendada y empleada para la obtención de datos fiables de la *NBA*. Esta librería contiene una gran cantidad de métodos para conseguir información específica relacionada con partidos y jugadores de la *NBA*. Para este caso, se han utilizado métodos de las clases `LeagueGameLogs`, `PlayerCareerStats`, `PlayerGameLogs`, `CommonTeamRoster`, `boxscoretraditionalv3` y `TeamGameLogs`.
Las *API* suelen defenderse de posibles ciberataques *DDoS*, que saturan servidores con una gran cantidad de solicitudes desde múltiples dispositivos al mismo servidor, mediante invalidación temporal de *IP* si detectan que se han realizado numerosas llamadas en un corto espacio de tiempo desde un mismo dispositivo. Para evitar la anulación de la *IP* al extraer datos, se introdujeron tiempos de espera en el código.
Como primera decisión del proyecto, se escogió extraer datos de los partidos desde la temporada 2003-2004, tanto de temporada regular como de *playoffs*. El motivo detrás de esta decisión es que es la temporada en la que debutó el jugador más veterano de la liga actualmente, Lebron James. Sin embargo, para los jugadores, se obtuvieron las plantillas actuales de los equipos de la *NBA* y con estas plantillas se extrajeron cada uno de los partidos jugados históricamente por cada uno de los jugadores y sus datos en estos partidos. Es por eso que aquí hay partidos de 2005, por ejemplo, en los que solo hay un jugador del que se tiene información, porque ningún otro se mantiene en activo. Esto pone de relieve la importancia de limpiar y mantener únicamente datos que realmente sean útiles y representativos.

4.2. Limpieza de Datos

Como se menciona anteriormente, la limpieza de datos es fundamental para que la red aprenda patrones de los datos y características útiles y no se quede con el ruido de los datos o con datos insignificantes.
De la clase de `LeagueGameLogs`, se mantuvieron las siguientes estadísticas para cada equipo, tanto del local como del visitante, dentro del mismo partido:

- Puntos totales
- Rebotes totales, tanto ofensivos como defensivos
- Asistencias totales
- Tapones totales
- Robos totales
- Faltas personales totales
- Número de tiros de tres intentados, anotados y el porcentaje de acierto
- Número de tiros de dos intentados, anotados y el porcentaje de acierto
- Número de tiros libres intentados, anotados y el porcentaje de acierto
- Ganador del partido

Con `LeagueGameLogs` se extraen otros datos también, como `TEAM_ABBREVIATION` o `SEASON_ID`, que no ayudan al modelo a predecir, simplemente dan contexto del partido.
En cuanto a los jugadores, se utilizó la clase `CommonTeamRoster` para conseguir las plantillas actuales de todos los equipos de la *NBA*; después, con la clase `PlayerCareerStats`, se extrajeron todos los partidos que ha jugado un jugador a lo largo de su carrera y, posteriormente, con `PlayerCareerStats`, se obtuvieron las siguientes estadísticas, entre otras, de cada partido jugado por el jugador:

- Puntos
- Asistencias
- Rebotes,tanto ofensivos como defensivos
- Pérdidas
- Robos
- Minutos
- Faltas personales
- Tiros de dos intentados, anotados y porcentaje de acierto
- Tiros de tres intentados, anotados y porcentaje de acierto
- Tiros libres intentados, anotados y porcentaje de acierto

De nuevo, estas son las estadísticas que se decidieron conservar, ya que hay otros datos irrelevantes para los modelos, como `NBA_FANTASY_PTS` o `DD2`, que son los puntos dados en la aplicación `NBA_Fantsy` por el jugador y si hizo o no un doble doble (dobles dígitos en dos estadísticas). Otra decisión destacable es que solo se tuvieron en cuenta partidos en los que se tuvieran al menos 30 minutos de tiempo jugado de los jugadores del equipo local y del equipo visitante. De esta manera, no se entrenan los modelos con datos de partidos y jugadores que no sean representativos; si solo se tienen 12 minutos de datos de los jugadores de un equipo en un partido, es difícil que el modelo pueda aprender cómo influyeron en el resultado del partido las estadísticas de los jugadores.

Es importante resaltar que, para entrenar una red neuronal *LSTM*, es fundamental el orden temporal. Debido a esto, tanto para los datos de los partidos como para los datos de los jugadores en los partidos, se mantiene el orden temporal. Esto se consigue con el dato extraído a través de las clases `LeagueGameLogs` y `PlayerGameLogs`, `GAME_DATE`, que especifica la fecha del encuentro. Los datos recogidos se ordenan mediante la fecha del encuentro.

4.3. Extracción de Características

Las estadísticas anteriormente recolectadas contienen información relevante sobre el desarrollo de un partido. Sin embargo, por sí solas no son interpretables por un modelo de *machine learning*. Ya que el modelo no dispone de conocimiento previo sobre las reglas y el funcionamiento del baloncesto. Por este motivo, es necesario manipular y transformar los datos para que el modelo pueda aprender de ellos.
En lugar de utilizar únicamente las estadísticas anteriores, como los puntos, rebotes y demás, se optó por generar características o *features* derivadas de estas que comparan directamente a ambos equipos. Esto permite al modelo aprender acerca de la superioridad e inferioridad relativa de un equipo respecto al otro en las distintas facetas del juego.
Las primeras características implementadas fueron la diferencia de asistencias, rebotes, tapones, robos, tiros de dos anotados, tiros libres anotados y tiros de tres anotados entre ambos equipos. No se implementó la diferencia de puntos entre ambos equipos, ya que puede introducir *data leakage* al modelo. Adicionalmente, se calcularon las características de *Net rating* y *Streak*. La primera es una estadística avanzada muy utilizada dentro de la *NBA* debido a lo representativa que es. El *Net rating* nos dice la eficiencia neta del equipo, calculada como la diferencia entre los puntos anotados y recibidos por cada 100 posesiones. La última característica calculada fue la racha del equipo como local y como visitante. Es decir, de los diez últimos partidos que el equipo jugó como local, cuántos ganó. Lo mismo como visitante. Estas últimas características nos indican, primero, cuán eficaz ha sido un equipo respecto a otro y, después, cuál es el estado de forma con el que ha llegado un equipo a un partido.

4.4. Preparación del Conjunto de Datos

A la hora de entrenar modelos en *machine learning*, es imperativo preparar los datos para que los modelos sean capaces de leerlos correctamente. Para las redes neuronales *LSTM*, los datos de entrada se estructuran de esta forma `(muestras, longitud de secuencia,características)`. Es por ello que se deben crear las secuencias. Los datos originales se dividen en pares de entrada (secuencia de entrada) y salida (valor o secuencia a predecir). En este caso, para predecir el resultado de un partido, *X*, se emplean los *N* partidos anteriores ordenados cronológicamente como entrada.

El algoritmo de *XGBoost* no está diseñado para trabajar con secuencias temporales, sino con datos tabulares; es por ello que los datos se deben ordenar de otra manera. Los datos empleados son los mismos que para la red 2. Sin embargo, este algoritmo funciona de un modo distinto. Para predecir el partido *X*, emplea los *N* partidos anteriores y calcula resúmenes numéricos de estos partidos. Es decir, las medias de estadísticas como rebotes o pérdidas y las diferencias de estas estadísticas entre los equipos, además de calcular la racha de los equipos. Todo ello se convierte en una única fila de números con la que se entrena el modelo. Con este resumen de los partidos, el modelo predice y es capaz de aprender patrones de los datos. La mayor diferencia con respecto a las redes *LSTM* es que, en los partidos empleados para predecir el partido *X*, no importa su orden para realizar la predicción. Este modelo no tiene en cuenta si un partido ocurrió antes que otro.

4.5. Diseño de los Modelos

Como se ha explicado anteriormente, en este proyecto se realizan cuatro implementaciones de modelos de *machine learning*. Dos configuraciones distintas de la misma red *LSTM*, red 1, configuración 1, y red 1, configuración 2, otra implementación de una red *LSTM* entrenada a partir de datos de partidos y jugadores, red 2, y, por último, una implementación del algoritmo *XGBoost* entrenada con los mismos datos que la red 2.

RED 1

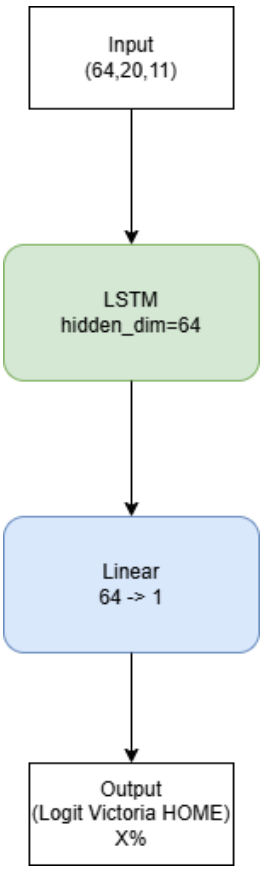

Figura 4.2: Red 1 Configuración 1

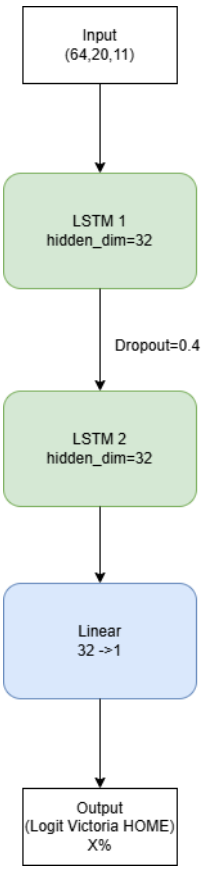

Figura 4.3: Red 1 Configuración 2

En estas imágenes, podemos apreciar las diferencias presentes en ambas configuraciones. Para las dos redes *LSTM*, la entrada es $(64, 20, 11)$. Esto nos indica que el tamaño del *batch* es 64, el número de partidos de una secuencia de la red *LSTM* es 20, y el número de *features* o características de cada partido es 11.

Ambas configuraciones comparten la misma estructura general: un tensor de entrada con los datos explicados en el párrafo anterior, una o varias capas *LSTM* encargadas de modelar la evolución temporal de las características y una capa final *Linear* que produce un único valor de salida. Este valor corresponde al *logit* asociado a la victoria del equipo local, que posteriormente es transformado en una probabilidad mediante la función *sigmoide* en el cálculo de la función de pérdida.

En la primera configuración (Figura 4.2), el modelo está compuesto por una única capa *LSTM* seguida directamente de una capa *Linear*. La capa *LSTM* emplea un tamaño del estado oculto de 64 unidades, lo que le permite capturar relaciones temporales de cierta complejidad manteniendo una arquitectura relativamente simple. Esta configuración presenta un menor número de parámetros, un coste computacional reducido y un menor riesgo de sobreajuste, lo que

la convierte en una arquitectura adecuada como modelo base.

La segunda configuración (Figura 4.3) introduce una mayor profundidad mediante el apilamiento de dos capas *LSTM* consecutivas, cada una con un tamaño de estado oculto de 32 unidades. Entre ambas capas se incorpora una capa de *dropout* con una tasa de 0.4, cuyo objetivo es regular el modelo durante el entrenamiento y mejorar su capacidad de generalización. El uso de dos capas *LSTM* apiladas permite que la primera capa capture patrones temporales más locales, es decir, tendencias a corto plazo, mientras que la segunda puede modelar dependencias temporales más complejas.

Desde el punto de vista funcional, las capas *LSTM* se encargan de procesar secuencias temporales manteniendo un estado interno que permite conservar información relevante de partidos anteriores, mitigando los problemas de desvanecimiento del gradiente presentes en redes recurrentes simples. Mientras que la capa *Linear* final actúa como un proyector que transforma la representación temporal aprendida por la o las *LSTM* en un valor escalar adaptado para un problema de clasificación binaria.

Aunque la segunda configuración implica un aumento en la complejidad del modelo y en el tiempo de entrenamiento, los resultados experimentales mostraron una ligera mejora en el rendimiento respecto a la primera. Por este motivo, se decidió seleccionar la configuración 2 como modelo *LSTM* principal para el resto del estudio.

RED 2

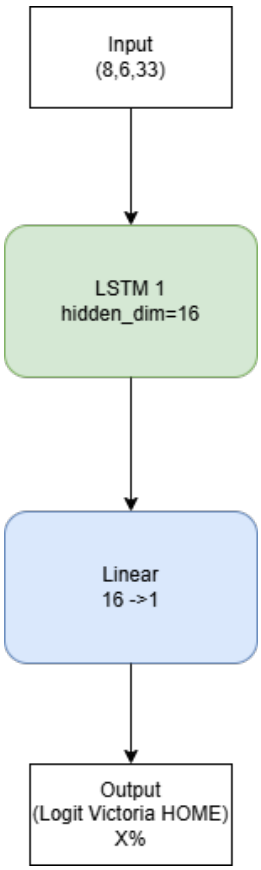

Figura 4.4: Red 2

Por su parte, para la red 2 se observa una entrada diferente, $(8, 6, 33)$.
La diferencia en el tamaño de la secuencia se debe a la diferencia de tamaño del *dataset* empleado por cada red. El *dataset* para la red 1 es de aproximadamente 28.000 partidos. Considerablemente más amplio que el escaso *dataset* de la red 2 de 1913 partidos. Debido al conjunto de datos, también se modifica el tamaño del *batch* a uno acorde. En cambio, se aprecia un tamaño mayor de características gracias al uso de datos no solamente de partidos, sino también de jugadores.

En la imagen se muestra el diseño de la red 2. Esta es una red simple con solamente una red *LSTM* apilada y un tamaño de 16 unidades de estado oculto. Esta decisión viene motivada principalmente por el tamaño reducido del conjunto de datos disponible. La configuración se enfoca en reducir de forma significativa el número de parámetros entrenables, lo que disminuye el riesgo de sobreajuste y favorece un entrenamiento más estable con las características de este *dataset*. El uso de una red más profunda o con una mayor dimensión oculta en este contexto resultaría contraproducente, ya que incrementaría la complejidad del modelo sin disponer de suficientes datos para justificarla. En estos casos, el modelo tendería a memorizar el conjunto de entrenamiento en lugar de aprender

patrones temporales robustos, suponiendo una disminución de su capacidad de generalización sobre nuevos datos.

Algoritmo XGBoost

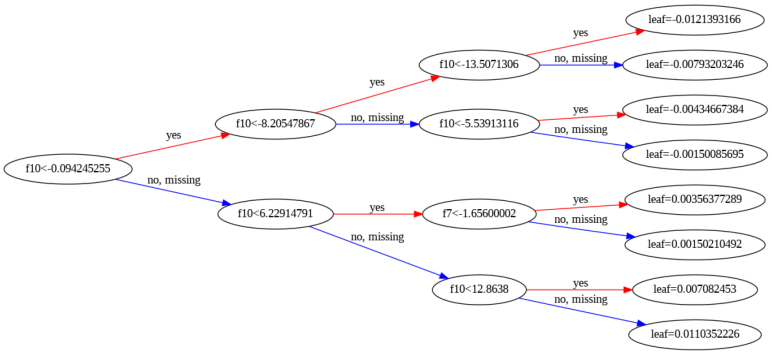

Figura 4.5: Árbol XGBoost

Debido a los resultados de la red 2, discutidos en el apartado de resultados, y a la sencillez e imposibilidad de hacer un modelo más complejo que capturase mejor las dependencias temporales de esta red, se tomó la decisión de realizar un modelo de *XGBoost* y comparar los resultados obtenidos.

Esta imagen muestra un ejemplo representativo de uno de los árboles de decisión que componen el modelo *XGBoost* implementado. El modelo final está formado por un *ensemble* de aproximadamente 400 árboles. En el árbol mostrado se observan distintos nodos de decisión. El árbol de decisión compara valores de las características para ir seleccionando distintas ramas hasta alcanzar una hoja final. Cada hoja contribuye con un valor escalar que forma parte del cálculo final de la probabilidad de victoria del equipo local. En este caso concreto, la mayoría de las divisiones del árbol se realizan en función de la característica *f10*, correspondiente a `TEAM_NETRTG_DIFF`, que representa la diferencia de *Net Rating* entre ambos equipos en un partido. Esto pone de manifiesto la relevancia de dicha característica en el proceso de toma de decisiones del modelo.

4.6. Entrenamiento

Como se explica en el apartado de fundamentos teóricos, otro aspecto fundamental para el aprendizaje del modelo es la división del *dataset*, ya que condiciona tanto el proceso de entrenamiento como la evaluación final del modelo. Los datos utilizados para entrenar estos modelos se dividen en tres subconjuntos: entrenamiento, validación y evaluación.

4.6.1. Redes Neuronales Recurrentes *LSTM*

Para la red 1, que utiliza un mayor conjunto de datos, se realizó una división del *dataset* del 70 % para la fase de entrenamiento, 15 % para la fase de validación y 15 % para la fase de evaluación. En el caso de la red 2, cuyo *dataset* es más reducido, se optó por una división del 80 % para entrenamiento, 10 % para validación y 10 % para evaluación, con el objetivo de disponer de un mayor número de datos para el aprendizaje del modelo.
Para el entrenamiento de ambas redes se empleó una función de pérdida *Binary Cross-Entropy* con *logits* (`BCEWithLogitsLoss`), ampliamente utilizada en problemas de clasificación binaria. Esta función de pérdida combina la función de pérdida `BCELoss` con una capa *sigmoide* interna para devolver la probabilidad de que gane el equipo local. Esta función mide la desviación entre las predicciones del modelo y los resultados reales.
Esta es la fórmula de la función de pérdida `BCELoss` (para un solo partido):

$$\mathcal{L}_{\text{BCE}}(y, \hat{p}) = -\left[y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})\right]$$

Donde $y \in \{0, 1\}$ es la etiqueta real del partido, ganó o perdió el local, y $\hat{p}$ es la probabilidad predicha de victoria del equipo local.

Un aspecto relevante de los *datasets* utilizados es el desbalanceo existente entre victorias locales y visitantes. En concreto, el *dataset* de la red 1 contiene aproximadamente un 58 % de victorias del equipo local, mientras que en el *dataset* de la red 2 este porcentaje es cercano al 52 %. Este desbalanceo puede provocar que el modelo tienda a realizar predicciones sesgadas hacia la clase mayoritaria, es decir, a predecir sistemáticamente la victoria del equipo local.

Para mitigar este efecto, se introdujo (`pos_weight`) en la función de pérdida, penalizando con mayor severidad los errores cometidos cuando el modelo predice la victoria local de forma incorrecta. De este modo, se fuerza a la red a basar sus decisiones en las características de entrada y no únicamente en la distribución estadística del *dataset*, especialmente en situaciones de alta incertidumbre.
Para la optimización de los modelos, se seleccionó el optimizador `AdamW`, una variante del optimizador `Adam` que introduce una regularización más adecuada mediante *weight decay*. Esta elección permite reducir el sobreajuste y mejorar la estabilidad del entrenamiento, corrigiendo algunas de las limitaciones presentes en la versión original de `Adam`. El *weight decay* es una técnica que penaliza los pesos grandes dentro de un modelo, empujándolos a cero para simplificar el modelo y que generalice mejor datos nuevos.
En ambos modelos se utilizó una tasa de aprendizaje inicial de 1×10^{-4}, junto con un *weight decay* de 5×10^{-5}, especialmente importante en la red 2 debido a su menor complejidad y mayor riesgo de sobreajuste.

Adicionalmente, se incorporó un `scheduler`, encargado de ajustar dinámicamente la tasa de aprendizaje durante el entrenamiento. Este `scheduler` reduce el *learning rate* cuando la pérdida en el conjunto de validación deja de mejorar durante un número determinado de épocas, en este caso dos, aplicando un factor de reducción de 0.5. Esta estrategia permite comenzar con una tasa alta para un progreso más rápido y reducirla a medida que se acerca al óptimo, ge-

neralmente en fases finales del entrenamiento, logrando una convergencia más estable y precisa.

4.6.2. Algoritmo *XGBoost*

Para el entrenamiento del modelo *XGBoost* se construyó un *dataset* tabular a partir de estadísticas históricas de equipos y jugadores, calculadas exclusivamente antes de la fecha de cada partido, evitando así cualquier tipo de *data leakage*.
Una vez construido el *dataset* final, los datos se ordenaron cronológicamente y se dividieron de forma secuencial en tres subconjuntos: 80 % para la fase de entrenamiento, 10 % para la fase de validación y 10 % para la fase de evaluación. Esta división temporal garantiza que el modelo se evalúa siempre sobre partidos posteriores a los utilizados durante el entrenamiento, simulando un escenario realista de predicción.
El modelo se entrenó con una función objetivo de clasificación binaria, *binary:logistic*, para estimar la probabilidad de victoria del equipo local. Durante el entrenamiento, se empleó la función de pérdida *log-loss* como métrica de evaluación, ya que penaliza más las predicciones incorrectas realizadas con alta confianza.
El modelo final está compuesto por un *ensemble* de hasta 400 árboles de decisión, cada uno con una profundidad máxima de 3 niveles. Esta configuración permite capturar relaciones no lineales simples entre las características sin aumentar excesivamente la complejidad del modelo, reduciendo así el riesgo de sobreajuste. La tasa de aprendizaje se fijó en $\eta = 0{,}01$.

4.7. Predicción

Una vez el modelo está entrenado, se puede realizar la predicción de un partido, que es el objetivo final del proyecto. Para ello se guardaron los modelos ya entrenados y se utilizaron métodos de las clases `boxscoretraditionalv3` y `TeamGameLogs` de la `nba_api` para recoger la información de los últimos 10 partidos jugados por los dos equipos de los que se quiere predecir el ganador. `Boxscoretraditionalv3` se utilizó para conseguir las estadísticas de los jugadores de cada equipo en los 10 partidos más recientes.
Por su parte, `TeamGameLogs` es utilizado para los últimos 10 partidos de cada equipo. La decisión de escoger estas dos nuevas clases y no utilizar las clases mencionadas con anterioridad en el documento viene motivada porque estas dos clases son actualizadas con mayor frecuencia. Por tanto, permiten obtener información de partidos incluso si se jugaron horas antes de realizar la predicción. Para la red 1, solamente se utilizan métodos de la clase `TeamGameLogs`; en cambio, para el modelo de *XGBoost* se utilizan métodos de las dos clases.

Aquí se puede observar el resultado devuelto de una predicción realizada por el sistema:

Oklahoma City Thunder vs Charlotte Hornets	
Probabilidad HOME	0.613
Probabilidad AWAY	0.387
Conclusión	Es un 61.3% probable que Oklahoma City Thunder gane a Charlotte Hornets

Capítulo 5

Análisis de Resultados

En este apartado se discuten los resultados obtenidos en las distintas métricas de los diferentes modelos evaluados.

5.1. Resultados Obtenidos

Durante el entrenamiento, cada modelo es entrenado durante un número determinado de épocas; sin embargo, cuando se detecta que el modelo deja de aprender durante varias épocas o comienza a sobreajustar, se realiza lo que se denomina como *early stopping* y se mantiene la última época en la que el modelo aprendió como la época de referencia. Se entiende que el modelo deja de aprender cuando el valor de la función de pérdida en esa época no desciende con respecto al mejor valor obtenido. Esto provoca una diferencia visible en las gráficas de este apartado en el número de épocas durante la fase de entrenamiento entre los distintos modelos evaluados.
Hay modelos que, razonablemente rápido, consiguen aprender todos los patrones posibles de los datos del conjunto de entrada, o simplemente tienden a sobreajustar y entonces se realiza el *early stopping*, y, en cambio, hay otros que necesitan un mayor número de épocas para aprender.

Como se menciona y explica anteriormente en el documento, de la red 1 se mide el rendimiento obtenido con las dos configuraciones más óptimas.

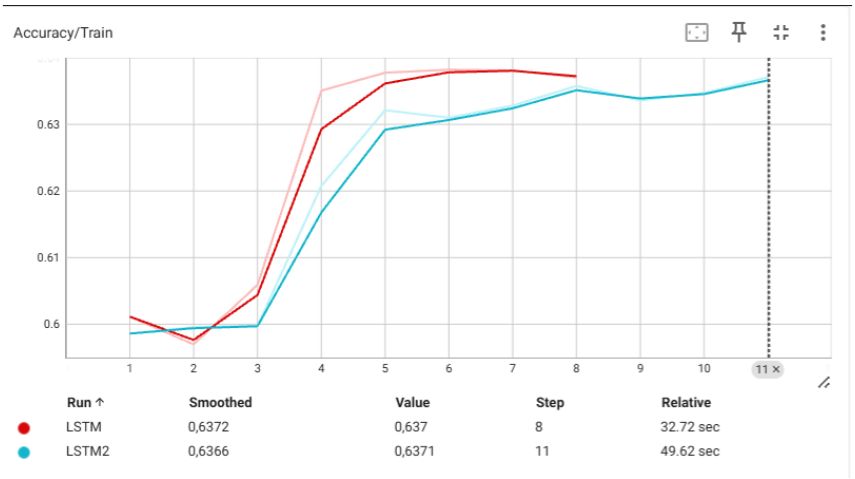

Figura 5.1: *Accuracy* Red 1 Ambas Configuraciones en Entrenamiento

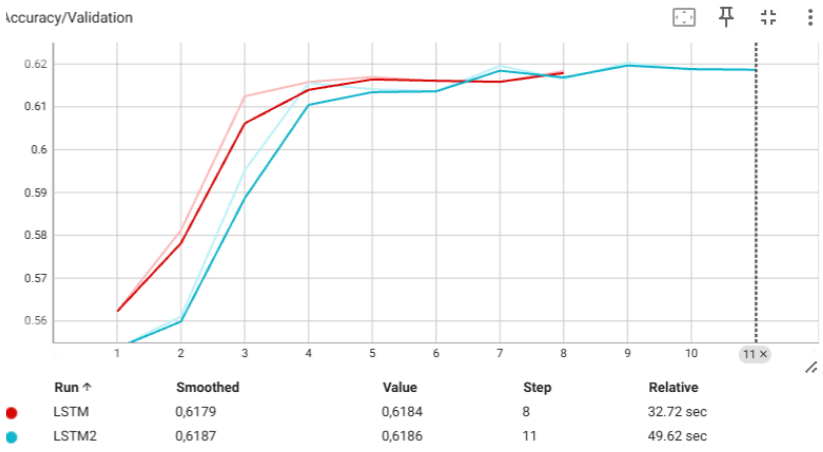

Figura 5.2: *Accuracy* Red 1 Ambas Configuraciones en Validación

En ambas imágenes, la configuración 1, la menos compleja, es la llamada *LSTM* representada de rojo. Por su parte, la configuración 2 se representa mediante la línea azul.
En los dos gráficos se aprecia la diferencia de épocas de entrenamiento, eje X, mostrando un aprendizaje más estable y continuado en el tiempo en la segunda configuración. Se aprecian tanto en el gráfico referente a la fase de entrenamien-

to como en el gráfico referente a la época de validación unos valores similares para las dos configuraciones. No obstante, la configuración 2 muestra un mayor acierto en la fase de validación respecto a la configuración 1. Otra observación es que ninguna de las configuraciones presenta sobreajuste. Los valores en la fase de validación son menos precisos pero dentro de un margen razonable, sin que el modelo llegue a memorizar los patrones de los datos de la fase de entrenamiento.
Se puede concluir que ambas configuraciones son satisfactorias, llegando a ser ligeramente superior la configuración 2, alcanzando casi el 62 % de acierto en la fase de validación. Es importante señalizar que ambas configuraciones mejoran los resultados del *baseline* para este *dataset*. Ambas configuraciones superan tanto en entrenamiento como en validación el 58 % de acierto.

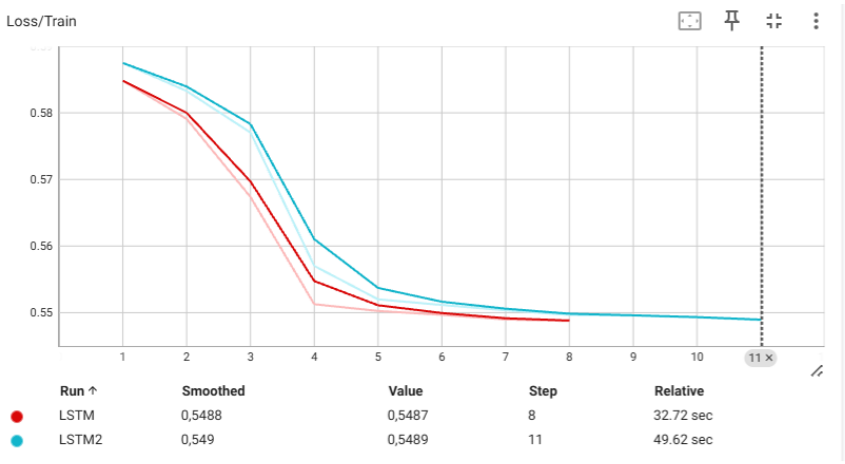

Figura 5.3: Función de Pérdida de Ambas Configuraciones en Entrenamiento

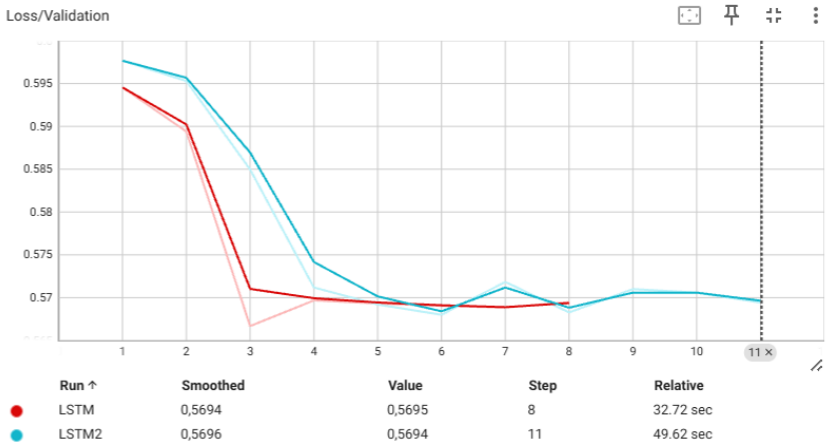

Figura 5.4: Función de Pérdida de Ambas Configuraciones en Validación

Nuevamente, se obtienen resultados similares de la función de pérdida para las dos configuraciones. Sin embargo, otra vez, la configuración 2 demuestra llegar a mejores valores, lo que se traduce en que este modelo cuando predice y acierta, realiza predicciones con un alto grado de confianza, mientras que cuando realiza una predicción y resulta ser errónea, es realizada con un menor grado de confianza, provocando unos valores más bajos en la función de pérdida.
En las tablas inferiores, se observan los resultados obtenidos por ambas configuraciones en distintas métricas durante la fase evaluación:

Red 1 configuración 1 :

Test Loss	Test Accuracy	Test F1	Test Precision	Test Recall	Test AUC
0.5688	0.6102	0.6490	0.6517	0.6464	0.6459

Cuadro 5.1

Red 1 configuración 2:

Test Loss	Test Accuracy	Test F1	Test Precision	Test Recall	Test AUC
0.5699	0.6147	0.6605	0.6492	0.6723	0.6466

Cuadro 5.2

Finalmente, se muestra un gráfico de barras con los resultados obtenidos en *accuracy* por ambas configuraciones en la fase de entrenamiento y en la fase de test:

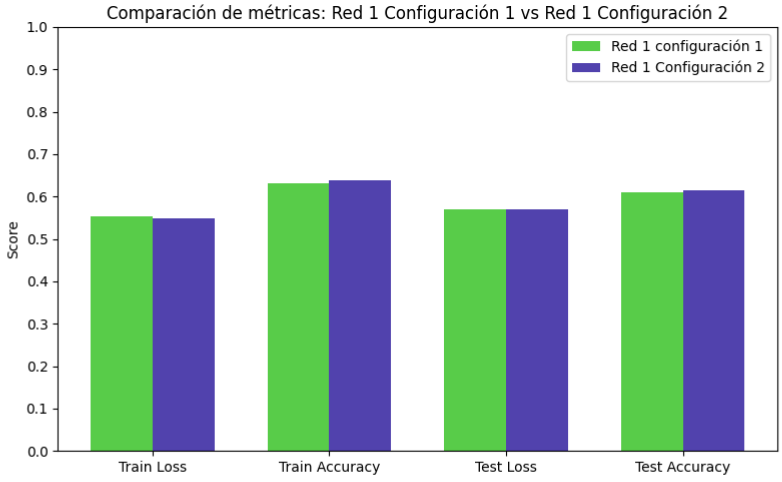

Figura 5.5: Gráfico de Barras Red 1 Configuración 1 vs Configuración 2

Gracias a las tablas (Cuadro: 5.1 y Cuadro: 5.2), se puede apreciar que la función de pérdida en la fase de evaluación presenta valores prácticamente idénticos para ambos casos (0.5688 y 0.5699), lo que sugiere que ambas configuraciones cometen errores de una magnitud semejante al evaluar datos no vistos durante el entrenamiento. Esto refuerza la idea de que no existe una diferencia significativa entre las dos configuraciones.

En cuanto a la precisión, ambas configuraciones presentan valores muy cercanos, 0.6517 y 0.6492, lo que indica que, cuando las configuraciones predicen una victoria, la probabilidad de que dicha predicción sea correcta es similar en ambos casos. No obstante, la configuración 2 destaca por un mejor valor de *recall*, 0.6723 frente a 0.6464, lo que implica que esta configuración presenta una mayor capacidad para identificar correctamente las victorias del equipo local y reducir las predicciones erróneas de que el equipo local no gana.
La métrica *F1*, que combina precisión y *recall*, muestra los siguientes valores, 0.6605 frente a 0.6490. Nuevamente, la configuración 2 resulta obtener mejor rendimiento en esta métrica. Este resultado corrobora las métricas comentadas en el párrafo anterior de precisión y *recall*.
Por último, el área bajo la curva *AUC-ROC* presenta valores prácticamente equivalentes (0.6459 y 0.6466), lo que indica que ambas configuraciones poseen una capacidad discriminatoria muy similar a la hora de diferenciar correctamente entre victorias y derrotas del equipo local, sin que ninguna destaque de forma clara sobre la otra.

El gráfico de barras (Figura: 5.5) pone de manifiesto la similitud de ambas configuraciones. Muestra unos valores muy similares de *accuracy* tanto en la fase de entrenamiento como en la fase de evaluación. Una vez más, la configuración 2 muestra una ligera superioridad, obteniendo un valor de 61.47 % frente al 61.02 % de la configuración 1. Aunque esta diferencia es mínima, indica que esta configuración es capaz de aprender patrones más útiles a partir de los datos debido a su diseño de dos capas *LSTM* apiladas.

En cuanto al modelo de la red 2 se obtienen los siguientes resultados:

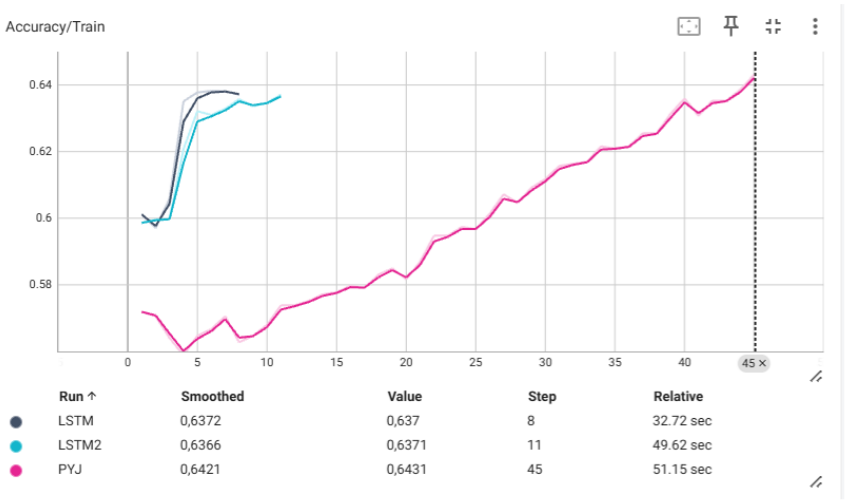

Figura 5.6: *Accuracy* de Ambas Redes Comparadas en Entrenamiento

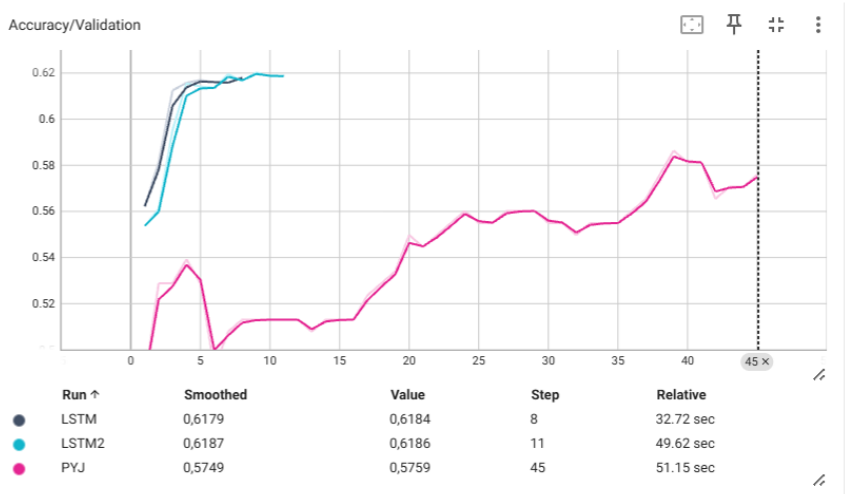

Figura 5.7: *Accuracy* de Ambas Redes en Validación

En ambas imágenes (Figura: 5.6 y Figura: 5.7) se observa claramente que este modelo realiza más épocas de entrenamiento que ambas configuraciones de la red 1, lo cual podría ser un buen indicador de que la red 2 realiza un aprendizaje continuado de los datos a lo largo del tiempo.

Sin embargo, en este caso se observa que el modelo no es ni consistente ni estable. También se aprecia que los resultados obtenidos son bastante inferiores a los de las configuraciones 1 y 2 de la red 1. No obstante, lo más representativo de esta red es su tendencia a sobreajustar. Se observa que a medida que avanza el entrenamiento, la *accuracy* de la fase de entrenamiento es considerablemente mayor a la *accuracy* en la fase de validación. Llegando a obtener valores superiores de *accuracy* en la fase de entrenamiento que las dos configuraciones de la red 1, pero mostrando un rendimiento muy inferior en la fase de validación. En el entrenamiento llega a obtener un 64 % de acierto, mientras que en validación no pasa del 57 %. Es decir, frente a partidos no vistos durante el entrenamiento, obtiene un acierto levemente mejor que el *baseline* para este conjunto de datos.

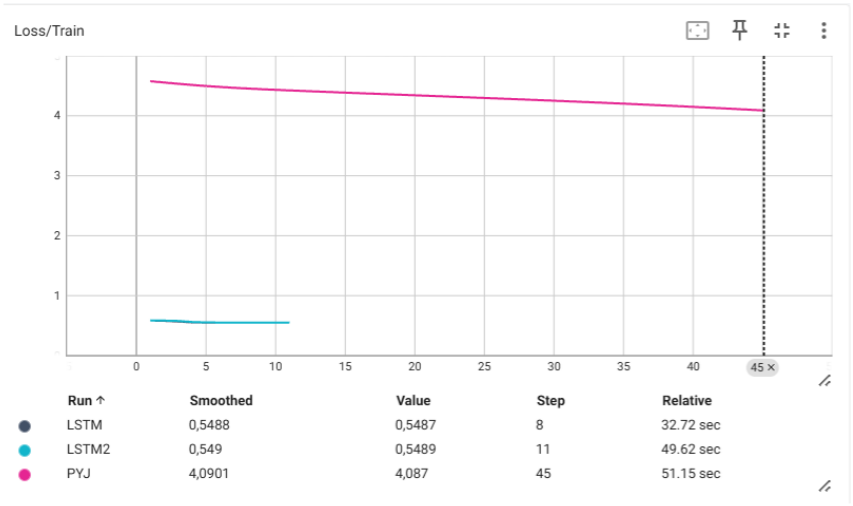

Figura 5.8: Función de Pérdida de Ambas Redes en Entrenamiento

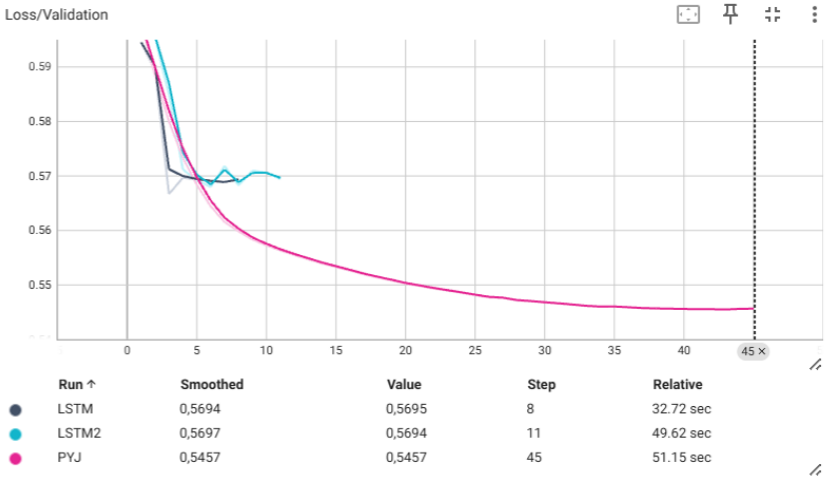

Figura 5.9: Función de Pérdida de Ambas Redes en Validación

Las imágenes (Figura: 5.8 y Figura: 5.9) muestran la evolución de la función de pérdida durante la fase de entrenamiento y durante la fase de validación para los distintos modelos evaluados de redes *LSTM*. La red 2 presenta valores de pérdida significativamente más elevados que los de ambas configuraciones de la red 1 durante el entrenamiento, situándose alrededor de un valor de 4.
Este comportamiento no se debe a una diferencia en la función de pérdida utilizada, ya que todos los modelos emplean la misma, sino principalmente a las características del conjunto de datos sobre el que se entrena esta red.
De hecho, este conjunto de datos es considerablemente más reducido y presenta una mayor variabilidad estadística, lo que provoca que los errores individuales tengan un mayor impacto en el valor medio de la pérdida. Como consecuencia, el modelo necesita realizar correcciones más intensas durante el entrenamiento,

lo que se refleja en valores de pérdida más elevados, especialmente en las primeras épocas.
No obstante, se observa una tendencia de disminución de la curva, lo que muestra aprendizaje durante esta fase.
En la fase de validación, muestra resultados incluso mejores que la red 1, volviendo a demostrar la inestabilidad e inconsistencia de esta red.

En esta red, se obtuvieron los siguientes resultados con el conjunto de datos de evaluación:

Red 2

Test Loss	Test Accuracy	Test F1	Test Precision	Test Recall	Test AUC
0.5683	0.4844	0.1239	0.7000	0.0680	0.5956

Cuadro 5.3

Por último, vemos un gráfico de barras que compara los resultados obtenidos por ambas redes *LSTM* en el conjunto de datos de prueba. Para esta comparación se utiliza la configuración 2 de la red 1:

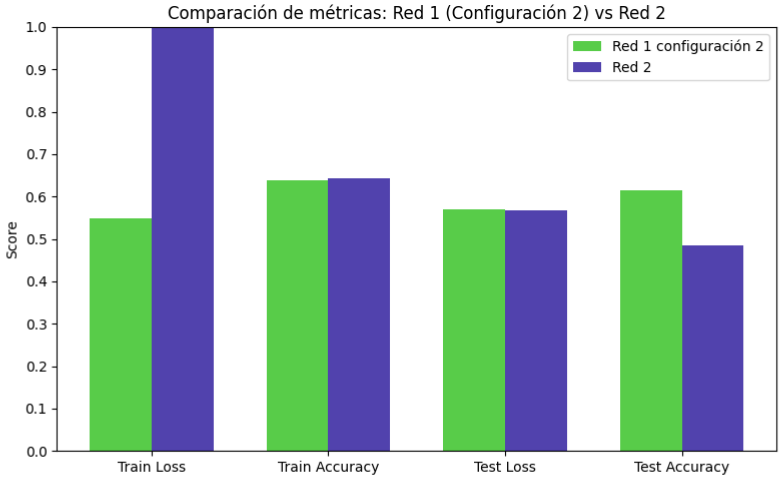

Figura 5.10: Gráfico de Barras Red 1 Configuración 2 vs Red 2

En lo relativo a la red 2, las métricas obtenidas sobre el conjunto de evaluación muestran un comportamiento claramente distinto al observado en la red 1, con métricas mucho más dispares.
En cuanto al valor obtenido de la función de pérdida, sigue la línea de los valores obtenidos durante la fase de validación. Es un valor aceptable y similar a los obtenidos por la red 1 configuración 2. Esto, nuevamente, desentona con el valor obtenido en su fase de entrenamiento.

El valor extremadamente bajo de la métrica *F1*, 0.1239, evidencia un desequilibrio significativo entre precisión y *recall*. Aunque la precisión alcanza un valor elevado, 0.7000, el *recall* es muy reducido, 0.0680. Esto indica que el modelo suele acertar cuando predice que el equipo local vence, pero falla en la mayoría de las ocasiones cuando predice que el local no gana.
Este comportamiento sugiere que la red 2 adopta una estrategia altamente restrictiva, realizando predicciones positivas únicamente cuando la confianza en la victoria del equipo local es muy elevada. Como consecuencia, se reduce el número de falsos positivos, pero se incrementa notablemente el número de falsos negativos, lo que penaliza severamente a la métrica *F1-score*.
Por último, el valor del área bajo la curva *AUC*, 0.5956, indica que, a pesar del bajo rendimiento en términos de clasificación, el modelo discrimina mejor que el azar, lo que se reflejaría en un valor del 0.5 en esta métrica.

Con los resultados de la tabla (Cuadro: 5.3) y el gráfico de barras (Figura: 5.10), se puede apreciar la diferencia de acierto en la fase de entrenamiento y en la fase de evaluación de la red 2, tanto por sí sola como respecto a la configuración 2 de la red 1.
La *accuracy* alcanza un valor del 48.44 % en la fase de test, lo que indica que el modelo ni siquiera es capaz de superar la predicción del *baseline* para este conjunto de datos. Este valor es cercano al porcentaje que se obtiene de manera azarosa, alrededor del 50 %. Esto confirma la inestabilidad de la red y su bajo rendimiento a la hora de predecir correctamente el ganador de un partido de baloncesto de la *NBA*.

En cuanto a la implementación de *XGBoost*, se obtuvieron los siguientes resultados:

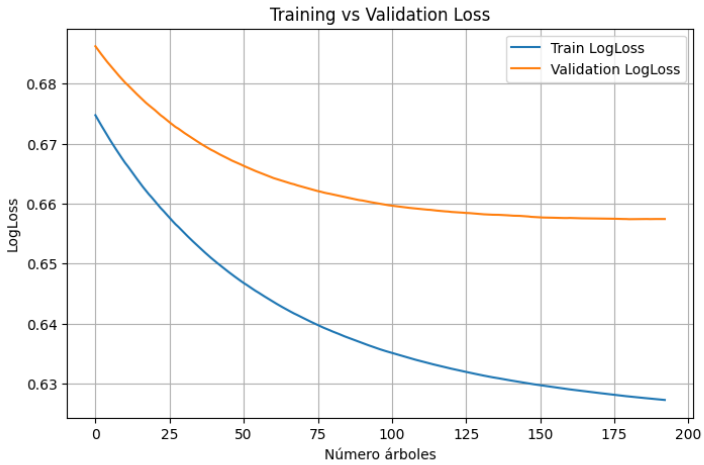

Figura 5.11: Función de Pérdida en Entrenamiento y Validación *XGBoost*

Se observa claramente que el valor de la función de pérdida desciende, lo que indica que el modelo aprende patrones y cada vez realiza predicciones más cercanas a la correcta. Sin embargo, se aprecia una notable diferencia entre los valores de pérdida de este modelo respecto a los valores de la red 1. Este modelo presenta valores cercanos al 0.63 % en la fase de entrenamiento y 0.66 % en la fase de validación, mientras que la mejor configuración de la red 1 obtiene valores cercanos al 0.55 % en el entrenamiento y 0.56 % en la fase de evaluación. Esto se traduce en que, a pesar de que este modelo aprende y mejora durante las distintas épocas del entrenamiento, la red 1 cuando acierta la predicción es con mayor grado de confianza y cuando se equivoca, su predicción no presenta una confianza tan alta como sí presenta este modelo.

Para este modelo se obtuvieron los siguientes datos en el conjunto de test:

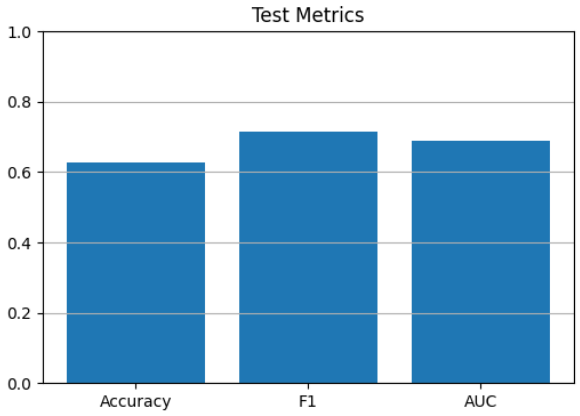

Figura 5.12: Gráfico de Barras Resultados Obtenidos en Test *XGBoost*

Esta imagen muestra una *accuracy* que alcanza un valor cercano al 63 %, lo que indica que el modelo clasifica correctamente más veces que ningún modelo *LSTM* anterior; en concreto, este valor resulta en una mejora de casi el 2 % en sus predicciones.
Por su parte, la métrica *F1* presenta un valor superior, en torno a 0.71, lo que sugiere un buen equilibrio entre precisión y *recall*. Este resultado indica que el modelo no solo realiza predicciones correctas, sino que también mantiene una relación adecuada entre la detección de casos positivos y la reducción de errores de clasificación.
Finalmente, el área bajo la curva *AUC* alcanza un valor aproximado de 0.69, lo que evidencia una capacidad discriminatoria superior de este modelo frente a la de las redes *LSTM*. Este valor indica que el clasificador es capaz de diferenciar entre las clases con una probabilidad significativamente superior al azar.

En conjunto, las métricas reflejan un comportamiento consistente del modelo, con un rendimiento equilibrado y estable en el conjunto de test, siendo especialmente destacable el buen valor obtenido en la métrica *F1*.
Este modelo presenta una característica realmente útil: es posible consultar qué características han contribuido más al aprendizaje y a las predicciones del modelo.

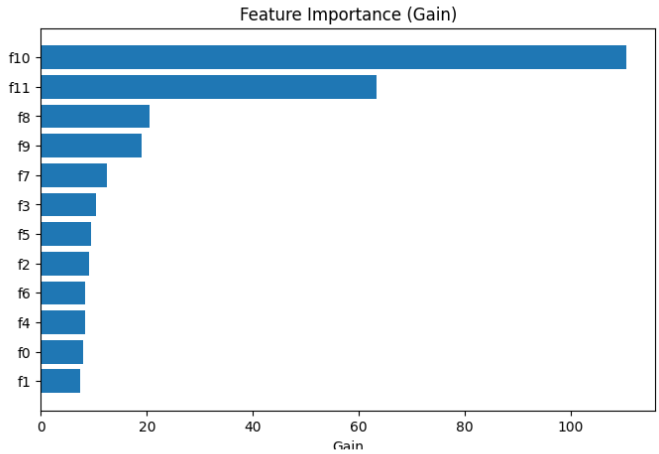

Figura 5.13: Importancia Características en la Predicción de *XGBoost*

En esta imagen se aprecia que las características más destacadas son la *f10* y la *f11*, que en este caso equivalen a `TEAM_NETRTG_DIFF` y a `TEAM_STREAK_DIFF` respectivamente. Es decir, este modelo aprende principalmente de la diferencia de *Net Rating* entre ambos equipos y de la diferencia de las rachas de victorias entre ambos equipos.
Gracias a este gráfico, también observamos que son bastante más relevantes las características y estadísticas empleadas relacionadas con el equipo que las referentes a los jugadores, que ni se muestran en el gráfico. Las 11 características presentes son las mismas 11 características que emplea la red 1. Esto permite determinar la dirección que deben tomar futuros trabajos con el objetivo de mejorar la precisión de estos modelos.

Capítulo 6

Conclusiones y Trabajos Futuros

El proyecto explora las capacidades predictivas de distintos modelos de *machine learning* entrenados con datos de dominio público. El objetivo principal fue implementar un modelo que prediga con cierto grado de acierto el ganador y el perdedor de un partido de la *NBA*.

Los resultados obtenidos por los distintos modelos demuestran que mejoran el *baseline* de cada conjunto de datos con el que fueron entrenados. Si bien el porcentaje de acierto máximo alcanzado, 63 %, se sitúa por debajo de herramientas de predicción más sofisticadas, como la que tiene la cadena de televisión estadounidense *ESPN*, que alcanza hasta un 72 % de acierto, es importante dar valor a estos resultados obtenidos y ponerlos en contexto. Dicha herramienta cuenta con acceso a un volumen de datos significativamente mayor, más detallado y de mayor calidad, incluyendo información interna de los equipos y de los jugadores que es muy probable que no esté disponible públicamente. Esto pone de manifiesto la importancia de los datos para este proyecto. No solo la utilidad y cuán descriptivos son los mismos, también su fiabilidad y proximidad en el tiempo. Otra consecuencia directa que producen los datos en *machine learning*, es el diseño del modelo. Un conjunto de datos pequeño limita el diseño y, por tanto, el aprendizaje de un modelo.

6.1. Trabajos Futuros

Dadas estas conclusiones y el posible margen de mejora de los modelos, se abren distintas líneas de trabajo futuro con el propósito de mejorar el acierto en los modelos. Uno de los factores más determinantes para mejorar la precisión de los modelos, como se ha mencionado en el párrafo anterior, es la selección y diseño de las características empleadas para entrenar el modelo. En el mundo del *machine learning* se ha popularizado una frase coloquial: *'Basura que entra, basura que sale'*, que ejemplifica que un modelo, por muy pulido y complejo que sea, no va a aprender de los datos, salvo que estos sean válidos, útiles y adecuados para el modelo en particular. Aunque las características utilizadas en este proyecto aportan información relevante, existen otros datos que no han

sido considerados y que podrían mejorar el acierto predictivo de los modelos empleados. Entre estos posibles datos, se encuentra el cansancio de los equipos con estadísticas medibles como los *back-to-back*, que es cuando un equipo juega dos días consecutivos. Más datos determinantes relacionados con el cansancio son: la distancia recorrida por el equipo, si es un viaje a la otra costa del país que conlleva una duración de viaje más larga de lo habitual, el *jet-lag* producido por ese viaje, etc. Todo esto afecta a los jugadores y, por tanto, a los equipos y al resultado. Otros datos que no se tienen en cuenta en este proyecto son el *Plus-Minus* de los jugadores, la vuelta de un jugador tras una lesión o una ausencia prolongada, el *ranking* del equipo en diferentes categorías, el propio *ranking* del equipo en la clasificación de la conferencia o incluso la posición en el *ranking* global de la *NBA*. En determinados casos, un equipo que ya no tiene posibilidad de acceder a los *playoffs* puede decidir *tankear* (finalizar en la peor posición posible en la clasificación de la *NBA* para optar a una mejor opción en el *draft* del año siguiente) y es muy probable que juegue con menos ánimos que el rival o incluso buscando la derrota. Esto podría reflejarse en los resultados de los partidos en los que juega este equipo.
Es importante destacar que los datos medidos durante los partidos también evolucionan y, consecuentemente, las estadísticas calculadas a partir de estos. Últimamente en el mundo *NBA*, se ha empezado a plantear el posible uso de una nueva estadística llamada *'gravity'*. Esta estadística mide cuánta atención atrae un jugador de la defensa, incluso sin tener el balón, cuantificando su impacto en la creación de espacio y oportunidades para sus compañeros.

Otro aspecto relevante a considerar en futuros trabajos es el diseño de la arquitectura del modelo. En este proyecto se aprecia que el modelo *XGBoost* es bastante eficaz cuando no se obtienen una cantidad suficiente de datos para diseñar una red *LSTM* competente. Sin embargo, con una cantidad de datos relevante y que muestren una secuencialidad entre ellos, las redes *LSTM* siguen presentando ventajas. No obstante, en los últimos años, las arquitecturas basadas en *Transformers* han demostrado un gran potencial en el modelado de dependencias complejas gracias a su mecanismo de autoatención, que permite asignar distinta importancia a las características en función de su relevancia. La exploración y experimentación con este tipo de modelos podría conllevar una mayor capacidad predictiva de los modelos.

En conjunto, este trabajo demuestra que es posible obtener predicciones competitivas a partir de datos públicos mediante un diseño cuidadoso del proceso de extracción de características y del modelo empleado. Asimismo, sienta las bases para futuras investigaciones orientadas a mejorar tanto la calidad de los datos como la complejidad de los modelos utilizados.

Bibliografía

[1] L. P. Carcedo, «El Mercado de las Apuestas Deportivas», *Departamento de Economía- Universidad de Oviedo. Facultad de Comercio, Turismo y Ciencias Sociales*, 2009.

[2] O. Economics, «Economic Impact of Legalized Sports Betting», *Oxford Economics*, págs. 22-26, 2017.

[3] Z. Zhou, «Real-Time Decision Modeling of Basketball Game Tactics Based on Video Analytics», *Journal of Electrical Systems*, 2024.

[4] J. Döpke, T. Köhler y L. Tegtmeier, «Are they worth it? – An evaluation of predictions for NBA 'Fantasy Sports'», *Journal of Economics and Finance (2024) 48:142–165*, pág. 1, 2024.

[5] W. Peng, «Machine Learning Models for Nba Game Prediction», *ITM Web of Conferences*, págs. 3-5, 2025.

[6] J. M. Pérez–Sánchez, E. Gómez–Déniz y N. Dávila–Cárdenes, «A Comparative Study of Logistic Models Using an Asymmetric Link: Modelling the Away Victories in Football», *Symmetry 2018, 10, 224*, págs. 2-6, 2018.

[7] J. M. Martín-González, Y. de Saá Geurra, J. M. García-Manso, E. Arriaza y T. Valverde-Estévez, «The Poisson model limits in NBA basketball: Complexity in team sports», *Physica A: Statistical Mechanics and its Applications*, 2016.

[8] M. Gifford y T. Bayrak, «A predictive analytics model for forecasting outcomes in the National Football League games using decision tree and logistic regression», *Science Direct*, pág. 4, 2023.

[9] C. KOLLWITZ, «ENHANCING THE PREDICTION OF SHOT SUCCESS IN NBA BASKETBALL GAMES USING MACHINE LEARNING TECHNIQUES – LSTM NEURAL NETWORK», *Nova School of Business and Economics*, págs. 52-60, 2023.

[10] M. I. Habib, «Forecasting NCAA Basketball Outcomes with Deep Learning: A Comparative Study of LSTM and Transformer Models», *Université Côte d'Azur*, 2025.

[11] M. R. Molina, «Técnicas de ensembles de modelos de Deep Learning aplicadas a la predicción de series temporales», *Escuela Politécnica Superior de Jaén*, págs. 33-39, 2022.

[12] Y. O. adn Xuewei Li, W. Zhou, W. Hong, W. Zheng, F. Qi y L. Peng, «Integration of machine learning XGBoost and SHAP models for NBA game outcome prediction and quantitative analysis methodology», *PLOS One*, pág. 5, 2024.

[13] F. Izaurieta y C. Saavedra, «Redes Neuronales Artificiales», *Departamento de Física Universidad de Concepción Concepción Chile*, 2000.

[14] C. Li, H. Zhang, Y. Zhang, J. Shen y R. An, «The application of artificial intelligence techniques in predicting game outcomes of professional basketball league: A systematic review», *PLOS One*, pág. 12, 2025.

[15] K. Puranmalka, «Modelling the NBA to Make Better Predictions», *MASSACHUSETTS INSTITUTE OF TECHNOLOGY*, págs. 15-19, 2013.

Anexos

Anexo A: Glosario

- ***Feature***: característica o variable utilizada como entrada para un modelo de *machine learning*.
- ***Data leakage***: situación en la que información del conjunto de datos usado para entrenar se filtra durante el entrenamiento del modelo, produciendo que el modelo aprenda los datos y no los patrones de los datos. Lo que da como resultado que frente a datos nuevos no sea capaz de actuar adecuadamente.
- **Lote(*batch*)**: número de muestras de entrenamiento procesadas conjuntamente en una sola iteración antes de actualizar los pesos del modelo.
- **Dataset**: conjunto de datos empleados en entrenamiento, validación y evaluación de los modelos de aprendizaje automático. En este proyecto, está compuesto por partidos de la *NBA* y sus características asociadas.
- **Hiperparámetro**: configuración externa de una red neuronal cuyo valor se fija antes del entrenamiento y no se aprende a partir de los datos, como la tasa de aprendizaje, el tamaño del lote o la dimensión del estado oculto.
- **Sobreajuste(*overfitting*)**: situación en la que el modelo se ajusta excesivamente a los datos del set de entrenamiento, lo que provoca una pérdida de capacidad de generalización y menor acierto sobre datos nuevos.
- ***Baseline***: modelo de referencia utilizado para evaluar el rendimiento de los modelos propuestos. En este proyecto, se refiere al porcentaje de acierto que se obtendría si siempre se predijera que gana el equipo local en el conjunto de datos utilizado para ese modelo.
- ***Dropout***: método empleado entre capas de redes neuronales que desactiva ciertas neuronas para evitar que la red dependa en exceso de alguna neurona y así se promueve el aprendizaje de patrones. Ayuda a evitar el sobreajuste.
- ***Hidden dimension***: tamaño del estado interno de una capa *LSTM*, que determina cuánta información temporal puede almacenar y procesar el modelo. En este trabajo, una mayor *hidden dimension* permite representar con mayor precisión la evolución del rendimiento de un equipo de la *NBA* a lo largo de varios partidos, a costa de una mayor complejidad del modelo.

- ***Logit***: logaritmo de la probabilidad de que un evento ocurra frente a la probabilidad de que no ocurra. El valor del *logit* puede variar desde $-\infty$ a $+\infty$. En este caso, el *logit* modela la confianza del sistema en la victoria del equipo local, se define: $logit(P) = ln(\frac{P}{1-P})$
- ***Ensemble***: técnica que combina los resultados de múltiples modelos individuales para obtener un modelo predictivo más preciso. En este proyecto, se trata un *ensemble* de árboles de decisión utilizado en el modelo de *XGBoost*.
- **Época**: una iteración completa en la que el modelo de *machine learning* ha procesado todo el *dataset* al completo.

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Thu Jan 15 19:13:45 CET 2026
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)