Modelado de Goles Esperados en Fútbol: Análisis Predictivo y Agrupación mediante Aprendizaje Automático.
Expected Goal Modelling in Football: Predictive Analysis and Clustering using Machine Learning.

Trabajo de Fin de Grado
Curso 2024–2025

Autor
Beñat Perez de Arenaza Eizaguirre

Director
Rubén Fuentes Fernández

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Calificación: 9.5

Modelado de Goles Esperados en Fútbol: Análisis Predictivo y Agrupación mediante Aprendizaje Automático.
Expected Goal Modelling in Football: Predictive Analysis and Clustering using Machine Learning.

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Beñat Perez de Arenaza Eizaguirre

Director
Rubén Fuentes Fernández

Convocatoria: *Junio 2025*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

16 de junio de 2025

Resumen

Modelado de Goles Esperados en Fútbol: Análisis Predictivo y Agrupación mediante Aprendizaje Automático.

Los goles esperados se han convertido en una herramienta fundamental para el análisis del fútbol. En este trabajo se presenta un planteamiento experimental y comparativo de la creación de modelos de goles esperados. Para ello, se han considerado dos líneas principales: en la primera de ellas se ha tomado como referencia el modelo ofrecido por *StatsBomb*, con el objetivo de replicar su enfoque. En la segunda, se ha creado un modelo desde cero, sin apoyarse en propuestas previas. En ambos enfoques se han comparado diferentes técnicas y formas de preprocesamiento y se ha llevado a cabo un análisis de reducción de variables. Además, se han incorporado variables posicionales de los jugadores y se ha analizado el impacto que han tenido en el rendimiento de los modelos.

Palabras clave

Goles esperados, aprendizaje supervisado, fútbol, clasificación binaria, posicionamiento de los jugadores, algoritmos de clasificación, regresión, algoritmos de regresión.

Abstract

Expected Goal Modelling in Football: Predictive Analysis and Clustering using Machine Learning.

Expected goals have become a fundamental tool for football analysis. This work presents an experimental and comparative approach to the creation of expected goals models. For this purpose, two main lines have been considered: in the first one, the model offered by StatsBomb has been taken as a reference, with the objective of replicating its approach. In the second, a model has been created from scratch, without relying on previous proposals. In both approaches, different techniques and forms of preprocessing have been compared, and a variable reduction analysis has been carried out. Additionally, positional variables of the players have been incorporated, and the impact they have had on the performance of the models has been analyzed.

Keywords

Expected goals, supervised learning, football, binary classification, player positioning, classification algorithms, regression, regression algorithms.

Índice

Índice de figuras

Capítulo 1

Introducción

El fútbol es un deporte que se inventó en Inglaterra durante el siglo XIX (Goldblatt, 2008). Pronto se extendió por Europa y después por el resto del mundo hasta convertirse en uno de los deportes más practicados y seguidos de todo el mundo. En sus orígenes, el deporte era muy ofensivo, y los jugadores se centraban casi exclusivamente en atacar. Conforme avanzó el tiempo, la sofisticación del juego también evolucionó. Los entrenadores empezaron a crear formaciones más equilibradas y se comenzaron a detallar posiciones más específicas (Wilson, 2013).

Fue a mediados del siglo XX cuando se empezaron a anotar las jugadas con el objetivo de encontrar patrones de juego que optimizaran las posibilidades de ganar los partidos (Anderson y Sally, 2013). Inicialmente se extrajo la conclusión, errónea según muchos, pero pionera en cuanto al análisis del deporte, de que había que tener un juego más directo.

A finales del siglo pasado, se comenzaron a usar las grabaciones de los partidos para extraer las mejores conclusiones posibles sobre qué tipos de jugadas eran las más efectivas. Asimismo, se empezaron a crear las primeras anotaciones de los eventos más significativos de los partidos (pases, tiros, faltas...).

Con la llegada del nuevo milenio, de la mano de Opta Sports (Barnett, 2014) se empezaron a anotar de forma detallada las estadísticas más relevantes de cada partido, lo cual supuso un antes y un después en la forma de entender el fútbol. Hasta entonces, todas las revoluciones tácticas que se desarrollaron en el fútbol fueron fruto de las acertadas intuiciones y del gran entendimiento del juego que tuvieron los grandes entrenadores de las décadas pasadas. Sin embargo, a partir de este punto, todos los equipos empezaron a incorporar todos estos datos para moldear su tipo de juego.

Finalmente, en la década del 2010, la irrupción de modelos estadísticos más avanzados como los goles esperados (xG del inglés *expected goals*) o asistencias esperadas (xA del inglés *expected assists*) abrió la puerta a la forma actual de analizar el fútbol (Fernández et al., 2019).

1.1. Goles esperados (xG)

Con la llegada de grandes volúmenes de datos y con el desarrollo de las técnicas de análisis de datos, una de las grandes revoluciones fue la aparición del concepto de goles esperados (xG) (Magazine, 2022). Aunque este concepto no se le pueda atribuir a ningún autor, en los primeros años del siglo XXI se desarrollaron los primeros modelos de goles esperados tanto por investigadores académicos (Ensum et al., 2005) como por empresas especializadas en el análisis de datos deportivos (Knutson, 2018) .

Hasta entonces, la forma de analizar el fútbol era ciertamente resultadista, o aunque no lo fuera en todos los casos, no había herramientas accesibles para evitar este tipo de análisis.

Llevando el reduccionismo al extremo, se puede analizar quién ha jugado mejor, meramente en función del resultado. No obstante, resulta evidente que este enfoque acarrea ciertos problemas. Podría suceder que un equipo pierda un partido a pesar de haber jugado mejor y habérselo merecido más que el rival, y viceversa. O podría suceder que un equipo, aun habiendo jugado mucho mejor que el rival, solamente ganara por un gol.

Siendo un poco menos reduccionistas, se puede analizar quién se ha merecido ganar un partido en función de la cantidad de tiros que ha hecho. Esto resulta algo más realista, puesto que *a priori* si un equipo ha disparado mucho más que otro, se puede suponer que ha tenido más oportunidades de ganar el partido. Afinando este análisis un poco más, también se podrían analizar los tiros a puerta que ha hecho cada equipo, ya que resulta evidente que hay tiros que apenas causan peligro, y que con el enfoque anterior se estarían teniendo en cuenta.

No obstante, todavía se puede intuir una gran imprecisión al medir quién debería haber ganado un partido teniendo en cuenta solamente los tiros a puerta. Es aquí, precisamente, donde entra el papel del concepto de los goles esperados. Aunque este concepto sigue siendo una simplificación, se trata de un enfoque mucho más esclarecedor de lo que ha sucedido en un partido.

Los goles esperados buscan atribuir una probabilidad de gol a cada tiro, de tal forma que se soluciona el problema detallado anteriormente de la diferencia de peligrosidad de los tiros. Es decir, un tiro desde el medio del campo no debería contabilizarse de la misma forma que un penalti o un tiro a portería vacía, puesto que la probabilidad de que cada uno de estos tiros termine dentro, y por tanto la peligrosidad de cada uno de estos tiros, no es para nada la misma. En caso de seguir el enfoque de los tiros o de los tiros a puerta, se puede apreciar que la diferencia de peligrosidad no se estaría teniendo en cuenta, lo cual llevaría a un análisis mucho más pobre y menos representativo de la realidad.

De esta forma, si se suman todas las probabilidades de gol de los tiros que ha hecho un equipo durante un partido, se obtiene lo que se conoce como los goles esperados. Resulta evidente que este enfoque será más preciso que los mencionados anteriormente, y es por eso que hoy en día se trata de la estadística más representativa del fútbol.

Cabe reiterar que este enfoque sigue dando lugar a imprecisiones en algunos

casos. Por ejemplo, podría suceder que un equipo hubiera llegado a posiciones muy cercanas a la portería del rival, pero que por imprecisiones en el último pase, o por intercepciones antes del disparo, la estadística de goles esperados no reflejara fielmente la realidad. No obstante, resulta una perspectiva mucho más fidedigna de lo acaecido durante un partido. Es por ello que, junto con el concepto de goles esperados, han aparecido otros conceptos como el de asistencias esperadas para tratar de reflejar más fielmente la realidad de lo ocurrido durante un partido.

En suma, la importancia de este concepto radica en que los partidos pueden ser analizados de forma más realista. Esto da lugar a que los entrenadores puedan hacer ajustes tácticos más detallados, puesto que son conocedores de la realidad de forma más precisa. De lo contrario, si, por ejemplo, se tomara el enfoque más reduccionista expresado anteriormente, un entrenador podría pensar que no debe hacer ningún ajuste puesto que ha ganado el partido. Esto podría llevarlo a creer que han ganado gracias a su propuesta táctica, cuando la realidad sería algo más cercana a que han ganado a pesar de su propuesta táctica.

Hoy en día son el equipo técnico de los clubes, los periodistas deportivos y los analistas deportivos los que emplean esta métrica para obtener un conocimiento más profundo de los partidos. No obstante, son los aficionados los principales beneficiados, dado el incremento del nivel que ha habido en los partidos gracias a la incorporación de estas herramientas de análisis.

1.2. Motivación

A raíz de la importancia creciente que está tomando el análisis de datos en muchas áreas, como es el caso de los deportes, y en especial en el fútbol, resulta interesante profundizar en este aspecto, que es uno de los que más ha cambiado el deporte en los últimos años. En el caso del fútbol, el concepto de goles esperados es uno de los más importantes en este tipo de análisis.

Como apasionado de los deportes y en especial de la parte táctica de los mismos, pienso que el desarrollo de este trabajo me puede aportar tanto profesional como personalmente.

Profesionalmente, el desarrollo de este trabajo me aportará conocimientos sobre el análisis de los datos y me servirá como introducción a este mundo. Podré familiarizarme con algunas de las herramientas que se usan en este campo y potenciar algunas de mis habilidades técnicas. Por otro lado, el desarrollo de este trabajo podría tener un impacto directo sobre el campo, en caso de obtener algún conocimiento nuevo sobre el tema.

En cuanto a la parte personal, podría empezar a ver los partidos con otros ojos. Por ejemplo, podría empezar a disfrutar de jugadas que, aunque no hayan terminado en gol, hayan generado mucho peligro. Esto permitiría convertirme en un espectador más curioso y más interesado en los pequeños detalles de los partidos. Además, podré difundir una idea menos resultadista del fútbol a la gente de mi alrededor, de la cual se podrían ver beneficiados.

1.3. Objetivos

El objetivo principal de este trabajo es crear un modelo que a cada tiro le atribuya una probabilidad de terminar en gol. Para ello, el trabajo de aprendizaje máquina para los goles esperados de un partido se subdividirá en objetivos más pequeños que servirán de subobjetivos para completar el trabajo:

- Recolectar datos de disparos.
- Preprocesar el conjunto de datos.
- Entrenar distintos modelos desde el enfoque de clasificación.
- Entrenamiento de distintos modelos desde el enfoque de regresión.
- Ajustar los mejores modelos.
- Estudiar el efecto de las variables posicionales, los distintos tipos de preprocesamiento de los datos, y de los distintos modelos.

El plan de trabajo (ver Fig. 1.1) ha sido como se describe a continuación: en primer lugar, se ha llevado a cabo la elección del conjunto de datos que ha servido de base para crear el modelo. Para ello,se tuvieron en cuenta, entre otras cosas, las variables que tiene el conjunto y el número de filas que tiene. Si presentara un número muy limitado de variables, o si las variables no fueran significativas, entonces difícilmente se podrá crear un modelo preciso. Asimismo, si el número de tiros que ofrece el conjunto de datos fuera muy reducido, resultaría complicado sacar conclusiones generales.

A continuación, se ha preprocesado el conjunto de datos para que se pueda utilizar de la forma más eficiente posible. Es decir, es probable que de primeras el conjunto de datos no esté listo para poder utilizarlo, por lo que se deben realizar una serie de modificaciones, entre las cuales se encuentran la eliminación de filas con valores no conocidos, la eliminación de variables poco significativas, la creación de nuevas variables representativas o la estandarización y/o normalización de los datos, entre otros.

Después de esto, el conjunto de datos ya se puede utilizar para entrenar un modelo. Para ello, se han seguido dos enfoques. Por un lado, se ha creado un modelo de probabilidad de gol sin ninguna referencia previa. Es decir, sin tener una probabilidad de gol preexistente, se ha tratado de crear un modelo que atribuya una probabilidad de gol a cada tiro.

Por otro lado, se ha tratado de calcular la probabilidad de gol de un tiro a partir de valores existentes de la probabilidad de gol que se tomarán de un modelo de referencia. En ambos casos se han empleado métricas para obtener información del rendimiento de los modelos.

Por lo tanto, el primer enfoque se corresponde con la construcción de un modelo desde cero, mientras que el segundo trata de replicar el funcionamiento de un modelo existente.

Acto seguido, se tomaron los mejores modelos y se ajustaron de tal forma que se maximizara el rendimiento. Para ello, se trató fundamentalmente de ver si aún había variables no significativas.

Después han extraído conclusiones de los resultados obtenidos y se han mostrado con el objetivo de plasmar los conocimientos extraídos del proceso descrito.

Finalmente, se ha redactado esta memoria una vez terminados los pasos anteriores.

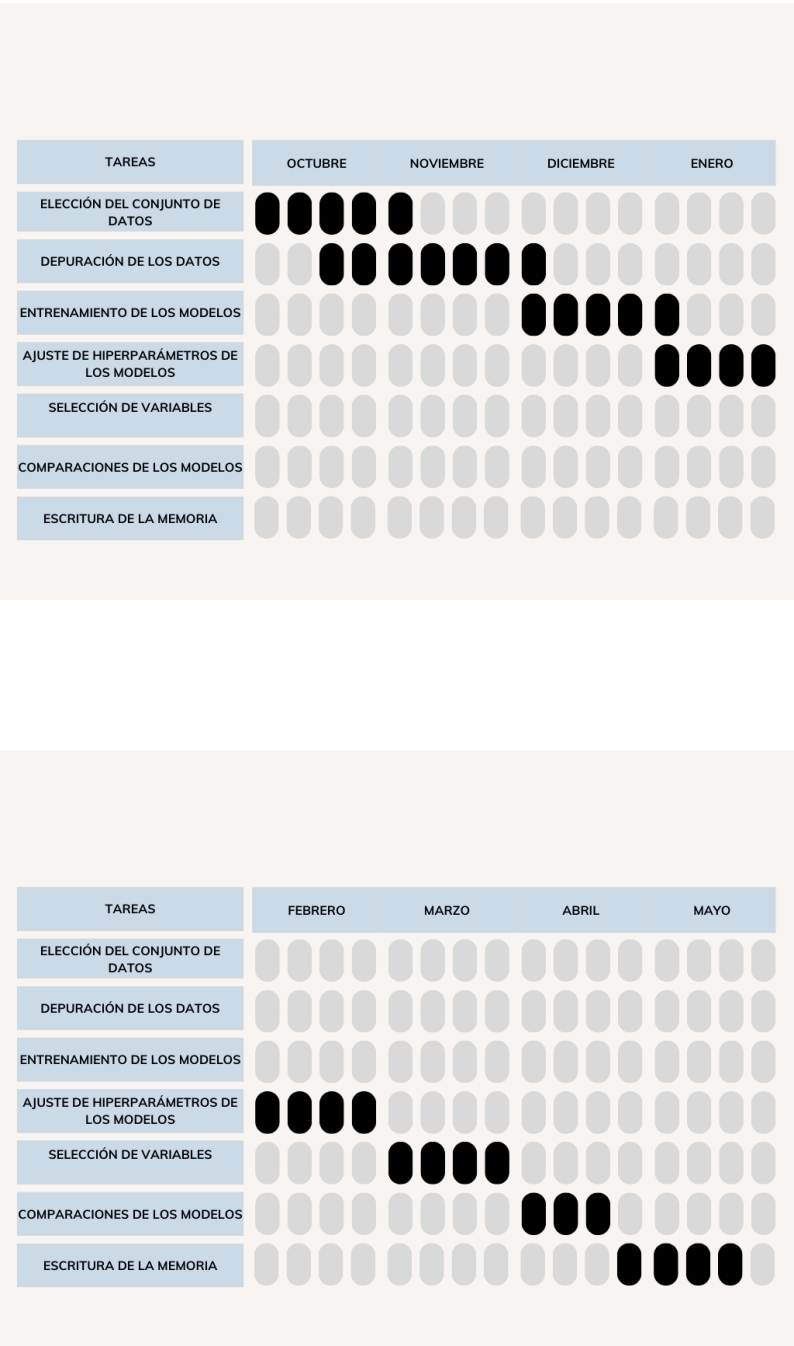


Figura 1.1: Diagrama de la organización temporal del trabajo

1.4. Estructura del documento

Durante los próximos capítulos se abordarán las siguientes cuestiones. En primer lugar, en el siguiente se tratarán los conceptos técnicos básicos necesarios para entender el desarrollo del trabajo, las técnicas y herramientas empleadas para el mismo, y finalmente se presentarán algunos de los trabajos previos de una naturaleza similar a este, así como la principal diferenciación del presente trabajo respecto a estos últimos (ver Capítulo 2).

A continuación, en el capítulo de infraestructura se detallarán las herramientas técnicas específicas que se han empleado para desarrollar el trabajo (ver Capítulo 3).

Posteriormente, se mostrará todo el proceso de la creación de los modelos de goles esperados, desde la creación del conjunto de datos, el procesamiento del mismo, hasta los distintos enfoques empleados para construir los modelos (ver Capítulo 4).

Antes de finalizar, se mostrarán los resultados obtenidos de los modelos generados, así como la comparativa entre los mismos (ver Capítulo 5).

Finalmente, en el último capítulo se plasmarán las principales conclusiones extraídas de la realización del trabajo, así como posibles cambios o mejoras que podría sufrir el mismo en una futura continuación del trabajo (ver Capítulo 6).

Capítulo 2

Estado de la Cuestión

El objetivo de este capítulo es contextualizar el problema, mostrar los antecedentes relevantes y posicionar mi propuesta en el marco de la investigación actual. Para dicha contextualización, se presentarán unas nociones básicas sobre los conceptos, técnicas y herramientas relacionadas con el problema.

La estructura que se va a seguir durante el capítulo va a ser la siguiente: primeramente se van a presentar los conceptos técnicos básicos que serán indispensables para entender el desarrollo del trabajo (ver sección 2.1). A continuación, se presentarán las bases de algunas de las técnicas que se emplearán en el desarrollo de los modelos (ver sección 2.2). Finalmente, se presentarán algunos trabajos que han servido de referencia (ver sección 2.3) y se concluirá con la diferenciación que presenta este trabajo respecto a los anteriores (ver sección 2.4).

2.1. Conceptos básicos

En esta sección se mostrarán las bases teóricas de los modelos estadísticos que se emplean en este tipo de problemas, lo cual facilitará la comprensión de los trabajos previos, así como la de este trabajo.

2.1.1. Aprendizaje automático

El Aprendizaje Automático es un campo de la Inteligencia Artificial que usa datos, experiencias e información previa para mejorar el rendimiento de un sistema (Alpaydin, 2021). Dentro del Aprendizaje Automático existen tres áreas: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

El aprendizaje supervisado busca predecir una variable partiendo de otras variables (que se conocen como atributos) (Hastie et al., 2009a). Para ello, usa datos de los cuales se conoce el valor de la variable a predecir (lo que se conoce como datos etiquetados). De esta forma, para un nuevo dato cuya variable a predecir se desconoce, se puede estimar dicho valor.

El aprendizaje no supervisado busca encontrar patrones, estructuras o agrupaciones en los datos sin utilizar datos etiquetados.(Hastie et al., 2009b).

En el aprendizaje por refuerzo, un agente aprende a tomar decisiones mediante recompensas y castigos para tratar de maximizar las recompensas acumuladas durante el tiempo (Wiering y Van Otterlo, 2012).

Como se verá más adelante en este capítulo, todos los trabajos que desarrollan un modelo de goles esperados emplean aprendizaje supervisado. Es por ello que en el presente trabajo se va a emplear el mismo enfoque.

A continuación se discutirán aspectos del aprendizaje supervisado con el objetivo de facilitar que el desarrollo del trabajo se pueda entender de forma más sencilla y completa.

2.1.2. Aprendizaje supervisado

El problema a tratar se enfocará desde el punto de vista del aprendizaje supervisado. Como ya se ha introducido, su objetivo es tratar de predecir una variable de salida mediante otras variables de entrada, haciendo uso de datos de los cuales se conoce la variable a predecir (Hastie et al., 2009a). Es decir, si se quiere predecir el precio de una casa en función de sus características, las variables de entrada serían las características de la casa (tamaño, número de habitaciones...) y la variable a predecir sería el precio.

2.1.3. Tipos de problemas

Dentro del aprendizaje supervisado existen dos tipos de problemas: clasificación y regresión. En el primero, la variable de salida solamente puede tomar una cantidad discreta de valores. Por ejemplo, si se quiere crear un detector de *spam* la variable de salida solamente podría tomar dos valores: *spam* o no *spam*.

En cambio, en los problemas de regresión, la variable de salida puede tomar infinitos valores. Quizás incluso es una variable continua, como es el caso de predecir la temperatura de los próximos días en función de datos de los días anteriores.

Esta distinción resulta especialmente relevante porque el trabajo, entre otras cosas, va a emplear ambos enfoques para la construcción de los modelos.

2.1.4. Función de pérdida

El objetivo de las diferentes técnicas que se presentarán posteriormente será encontrar una función que a cada dato de entrada le atribuya un valor de salida. Para ello, en algunas de las técnicas de aprendizaje supervisado, se hará uso de una función de pérdida que se tratará de minimizar (Raximov et al., 2022). Es decir, si las técnicas presentan parámetros que ajustar, la mejor combinación de parámetros será la que minimice la función de pérdida, lo cual dará lugar a la función que asigna datos de entrada con valores de salida con la mejor precisión posible.

Esto resulta vital para entender cómo se lleva a cabo el ajuste de los modelos en aquellas técnicas que precisen de ello.

2.1.5. Creación y ajuste de los modelos

Existen modelos que dependen de ciertos parámetros. Para que el rendimiento del modelo sea el mejor posible, se deben ajustar dichos parámetros para tratar de buscar la mejor configuración de los mismos. Esto es lo que se conoce como el ajuste de los parámetros.

Adicionalmente a los parámetros que pueda tener un modelo, en algunas de las técnicas que se van a emplear existen otras variables, conocidas como hiperparámetros, que deben determinarse antes de ajustar dichos parámetros. Por tanto, antes de encontrar la mejor configuración de parámetros y posteriormente evaluar el rendimiento del modelo con dicha configuración, se deben predeterminar los valores de los hiperparámetros (Arnold et al., 2024).

Relacionado con esto, una de las formas más habituales de elegir diferentes combinaciones de hiperparámetros es la búsqueda en rejilla (*grid search*), de la que se hará uso más adelante. Esta búsqueda consiste en entrenar el modelo con cada una de las posibles combinaciones de hiperparámetros que se proporcionen.

2.1.6. Particiones del conjunto de datos

El objetivo principal de estas técnicas es entender relaciones entre variables de entrada y salida para nuevos datos. Es decir, de nada sirve que se prediga correctamente la variable de salida de todos los datos de los que se dispone, si al utilizar como entrada un dato desconocido, no es capaz de predecir correctamente su salida. Es por esta razón que nace la necesidad de dividir el conjunto de datos en dos partes. Por un lado, se destinará una parte para ajustar el modelo. Por otro lado, tras realizar ese ajuste, se empleará el otro conjunto para probar el verdadero rendimiento predictivo sobre datos no vistos anteriormente. Estos conjuntos se conocen como conjunto de entrenamiento y conjunto de prueba, respectivamente (Arnold et al., 2024).

Con el objetivo de elegir la mejor configuración de hiperparámetros, al igual que sucedía con la necesidad de separar el conjunto con el que se ajustan los parámetros y el conjunto con el que se evalúa el rendimiento del modelo, se debe separar un nuevo conjunto que servirá para determinar la mejor configuración de valores de los hiperparámetros. De esta necesidad surge un tercer conjunto que se conoce como conjunto de validación (Burzykowski et al., 2023).

De esta forma, para cada configuración de hiperparámetros, se ajustan los parámetros del modelo con el conjunto de entrenamiento, y después se evalúa el rendimiento del modelo creado a partir de cada configuración distinta de hiperparámetros en el conjunto de validación. Así, se obtiene el modelo cuya elección de hiperparámetros ha obtenido un mayor rendimiento. Finalmente, una vez elegidos los hiperparámetros y los parámetros del modelo, se evalúa el rendimiento con el conjunto de prueba.

Estos tres conjuntos resultarán relevantes puesto que, dependiendo de cómo se haga la partición, los modelos resultantes pueden diferir considerablemente entre sí.

2.1.6.1. Validación cruzada

Resulta evidente que si se dispone de muy pocos datos, el hecho de tener que prescindir de algunos de ellos para la validación o para el test, podría llevar a que el poder predictivo del modelo se viera mermado. Además, dependiendo de cómo se haga la partición de los datos, los resultados pueden variar. Por ello, con el objetivo de utilizar la máxima cantidad de datos para cada fase y con el objetivo de reducir la aleatoriedad de las distintas posibilidades de particionar los datos, se utiliza lo que se conoce como validación cruzada.

Para ello, se divide el conjunto de datos en entrenamiento y en prueba. A su vez, el conjunto de prueba se divide en k particiones (también conocidas como $folds$). A continuación, se entrena el modelo k veces, utilizando en cada iteración $k-1$ particiones del conjunto de datos como conjunto de entrenamiento, y dejando la partición restante como conjunto de validación. Así, cada partición se utiliza exactamente una vez como conjunto de validación. Al finalizar este proceso, se hace una media de los rendimientos de las k validaciones. Esto se hace para cada configuración de hiperparámetros.

2.1.7. Sobreajuste y subajuste

Tras los procesos de entrenamiento, validación y prueba descritos, podría suceder que el modelo tuviera un buen rendimiento en validación, es decir, que para los datos con los que se entrena predijera con mucha precisión la variable de salida, pero que con el conjunto de prueba el rendimiento no fuera tan bueno. En este caso, el modelo se habría ajustado demasiado a los datos de entrenamiento, capturando el ruido de los mismos. Es decir, no se habría limitado a encontrar el patrón general de los datos, sino que habría ido más allá tratando de captar las particularidades que presentan los datos de entrenamiento. No obstante, cuando se presenta un nuevo dato, no sería capaz de predecir correctamente su variable de salida, puesto que se habría ceñido demasiado a los primeros datos. Esto es lo que se conoce como sobreajuste (o $overfitting$ del inglés).

Por el contrario, si el modelo no es capaz de encontrar el patrón general que siguen los datos, el rendimiento del modelo también sería defectuoso. En este caso, no se estaría ajustando demasiado a los datos, sino que se habría ajustado tan poco a ellos que ni siquiera habría encontrado un patrón general que siguen todos ellos. Esto es lo que se conoce como infraajuste (o $underfitting$ del inglés).

Estos conceptos resultarán importantes a la hora de crear los modelos de goles esperados, puesto que un modelo que con los datos conocidos tenga un buen rendimiento pero que ante datos nuevos no haga predicciones acertadas carecerá de utilidad. De igual forma, un modelo que no capture el patrón general de los datos y que, por tanto, carezca de poder predictivo también carecerá de utilidad.

Para evitar esto, se emplean distintas técnicas como la validación cruzada, la incorporación de datos nuevos que abarcan nuevas zonas del espacio de características, o lo que se conoce como regularización, que se presentará a continuación.

2.1.8. Regularización

La flexibilidad de un modelo mide la capacidad del mismo para aproximar funciones. Si un modelo es muy flexible, entonces podrá captar el ruido de los datos, ya que se podrá ajustar mucho a los mismos. Mientras que si es menos flexible, no podrá captar las peculiaridades de cada dato, y solamente podrá capturar el patrón general que siguen. La regularización es, precisamente, lo que controla la flexibilidad de un modelo (Tian y Zhang, 2022).

Para ello, en la función de pérdida previamente introducida, se añade un elemento de regularización que penaliza a los modelos con más flexibilidad. De esta forma, para minimizar la nueva función de regularización se primarán modelos menos flexibles, es decir, que no se ajusten tanto a los detalles particulares de los datos. Así, al ajustar los parámetros del modelo, se buscará un equilibrio entre ajustar correctamente los datos, pero sin capturar el ruido de los datos, para captar el patrón general de los mismos.

2.1.9. Métricas

En esta sección se ha hablado del rendimiento de los modelos. Para poder comparar distintos modelos entre sí, se necesitan unas mediciones que permitan discriminar cuál es mejor para cierto propósito. Es aquí donde entran en juego las métricas (Naidu et al., 2023). Existen distintas métricas y a continuación se detallarán las principales para clasificación y para regresión.

2.1.10. Métricas para clasificación

La métricas presentadas a continuación serán para el caso binario, es decir, el caso en el que solamente hay dos clases, como sucederá con el caso de los tiros, que tomarán la etiqueta de "gol" o "no gol". Para definir las métricas para clasificación, primero se deben presentar los siguientes conceptos:

- **TP (True Positives)**: número de ejemplos positivos correctamente clasificados como positivos.
- **TN (True Negatives)**: número de ejemplos negativos correctamente clasificados como negativos.
- **FP (False Positives)**: número de ejemplos negativos clasificados incorrectamente como positivos.
- **FN (False Negatives)**: número de ejemplos positivos clasificados incorrectamente como negativos.

A partir de estos conceptos se definen las siguientes métricas:

- **Exactitud (*Accuracy*):**

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN}$$

Muestra la proporción de individuos que han sido clasificados correctamente. No obstante, dependiendo del contexto podría no ser suficientemente representativo. Por ejemplo, si se quiere predecir si un paciente tiene una enfermedad poco común, y solamente un 1 % de los pacientes lo tienen, un predictor que para cualquier individuo predijera la clase negativa, es decir, que no tiene la enfermedad, tendría un 99 % de exactitud, pero resulta evidente que no tiene utilidad alguna para el caso. Es de aquí que surge la necesidad de utilizar otras métricas.

- **Precisión:**

$$\text{Precisión} = \frac{TP}{TP + FP}$$

Muestra la proporción de individuos realmente positivos entre todos aquellos predichos como tal. Resulta particularmente importante cuando los falsos positivos son costosos. Por ejemplo, si un tratamiento médico es muy costoso o acarrea ciertos riesgos, se debe tratar de minimizar el número de individuos clasificados como positivos cuando realmente no lo son.

- **Exhaustividad (Recall):**

$$\text{Exhaustividad} = \frac{TP}{TP + FN}$$

Mide la proporción de verdaderos positvos entre todos los individuos de la clase positiva. Resulta importante cuando los falsos negativos suponen un coste o perjuicio. Por ejemplo, en un control de dopaje, si existen falsos negativos estos individuos podrían tener una ventaja injusta respecto de otros competidores, por lo que resulta de vital importancia minimizar el número de los mismos.

- **F1 :**

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Exhaustividad}}{\text{Precision} + \text{Exhaustividad}}$$

Es una métrica que combina las dos anteriores.

- **Área bajo la curva ROC (AUC-ROC):**

Mide el área bajo la curva ROC (ver fig 2.1). Esta curva se construye de la siguiente forma. Se determina un umbral $u \in [0, 1]$ a partir del cual se clasifican los individuos como positivos y se mide la tasa de verdaderos positivos(exhaustividad) y la tasa de falsos positivos (donde la tasa de falsos positivos mide la proporcion de falsos positivos respecto del número total de casos negativos). La curva *ROC* es, por tanto, el conjunto de puntos obtenidos al varías el umbral u en el intervalo $[0, 1]$. El *AUC* es el área bajo esta curva. La clasificación perfecta es el punto (0,1) y en (0,5, 0,5) es aleatoria. Por tanto, cuanto más se acerque el área a 1, mejor será el modelo discerniendo entre instancias positivas y negativas.

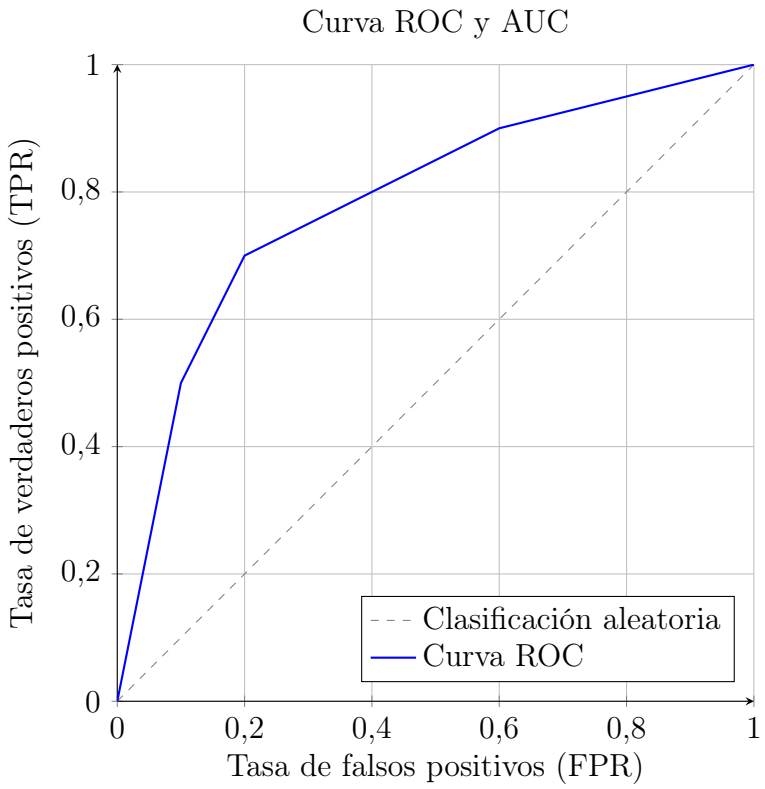


Figura 2.1: Curva ROC con el área bajo la curva (AUC) resaltada en azul.

2.1.10.1. Métricas para regresión

A continuación se detallarán algunas de las métricas que más se emplean para problemas de regresión. En la siguiente discusión y_i hace referencia al valor real de la variable de salida para el dato de entrada x_i y $\hat{y}_i$ hace referencia al valor predicho para esa misma entrada.

- **Error absoluto medio (MAE):**

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

  Mide la media del valor absoluto de las diferencias de los valores predichos y los valores reales.

- **Error cuadrático medio (MSE):**

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

  Mide la media del cuadrado las diferencias entre los valores predichos y los valores reales.

- **Raíz del error cuadrático medio (RMSE):**

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

  Es la raíz cuadrada de MSE.

- **Coeficiente de determinación (R^2):**

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

  Mide qué proporción de la varianza de los datos reales se puede explicar usando las predicciones.

- **Varianza explicada (Explained Variance):**

$$\text{ExplainedVariance} = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$$

  Mide cúanto se ha reducido la varianza de las diferencias entre el valor real y el valor predicho respecto de la varianza de los valores reales.

- **Error absoluto mediano (Median Absolute Error):**

$$\text{MedAE} = \text{mediana}\left(|y_i - \hat{y}_i|\right)$$

  Se trata del valor mediano del valor absoluto de las diferencias de los valores predichos y los valores reales.

2.2. Técnicas y algoritmos

En esta sección se abordarán algunas de las técnicas empleadas en la construcción de modelos de goles esperados existentes. Los trabajos que se presentarán más adelante hacen uso de algunas de estas técnicas, y son las más comunes para la creación de este tipo de modelos. Adicionalmente, se mostrarán los fundamentos de las técnicas de redicción de variables que se emplearán posteriormente.

2.2.1. Regresión logística

Se trata de un modelo que se emplea para calcular la probabilidad que tiene un dato de entrada de pertenecer a una clase (Nick y Campbell, 2007). Para ello, se emplea la función logística, que atribuye una probabilidad de que un dato de entrada pertenezca a la clase positiva.

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

Donde:

- $P(y = 1 \mid x, \beta)$: Probabilidad de que x pertenezca a la clase 1.
- $X = (x_1, x_2, \ldots, x_n)$: es el dato de entrada.
- $\beta = (\beta_0, \beta_1, \ldots, \beta_n)$: son los parámetros del modelo que deben ser ajustado durante el entrenamiento.

La probabilidad de que el dato de entrada no pertenezca a la clase positiva es el complementario de que pertenezca a la clase positiva.

Para tratar de encontrar los mejores valores de los parámetros, se usa la función de verosimilitud:

$$L(\beta) = \prod_{i=1}^{n} P(y = 1 \mid X, \beta)^{y_i}(1 - P(y = 1 \mid X, \beta))^{1-y_i}$$

Se tratarán de encontrar los valores de β que maximicen la función de verosimilitud.

2.2.2. Árboles de decisón

Los árboles de decisión buscan hacer una partición del espacio de características buscando regiones homogéneas respecto de la variable de salida y sirven tanto para clasificación como para regresión (De Ville, 2013). En el caso binario, los árboles resultantes también son binarios. Cada nodo no terminal representa el cumplimiento/incumplimiento de una restricción. Es decir, un hijo representa la región del espacio de características resultante de cumplir la condición del nodo, y el otro hijo la región del espacio de características resultante de no cumplir dicha condición. Las hojas del árbol contienen todos los datos de entrenamiento.

Para saber a qué hoja pertenece un nuevo dato, se debe empezar desde la raíz, y se debe descender por el árbol por la derecha o por la izquierda en cada nodo dependiendo de si se cumple o no la condición, hasta llegar a una hoja.

En caso de estar frente a un problema de clasificación, la clase predicha para un dato de entrada coincide con la clase mayoritaria de la hoja del árbol a la que pertenece ese dato. En el caso de la regresión, una posibilidad que se utiliza es hacer una media de los datos de entrenamiento que pertenecen a dicha hoja, y el nuevo dato de entrada toma esa media como predicción.

En la construcción del árbol, se parte de todos los datos de entrenamiento y se elige la variable y el valor de dicha variable que más disminuya la falta de homogeneidad entre los valores de salida asociados a los datos en los nodos hijo del árbol. Para ello se usan funciones de impureza como la entropía o el índice de *gini*. Se opera de esta forma hasta llegar a alguna condición de parada, entre las cuales se encuentran la profundidad máxima del árbol, la reducción mínima de la impureza para dividir un nodo, o el número mínimo de datos por nodo.

2.2.2.1. Random Forest

Esta técnica consiste en generar varios árboles de decisión y combinar sus resultados para generar una mejor predicción (Rigatti, 2017). Para ello, a diferencia

de los árboles de decisión, donde los árboles se generaban teniendo en cuenta todos los datos de entrenamiento, Random Forest hace uso del muestreo aleatorio con reemplazo. Esto es, para cada árbol, se eligen aleatoriamente una serie de datos del conjunto de entrenamiento, y si un dato se elige para la construcción de un árbol, puede ser elegido para la construcción de otros. Además, para construir cada árbol, en vez de elegir todas las variables, se elige un subconjunto de las mismas. De esta forma se genera un conjunto (bosque) de árboles de decisión.

En el caso de clasificación, para elegir cuál es la predicción de un dato de entrada, se elige la clase mayoritaria entre todas las clases predichas por cada uno de los árboles. En caso de la regresión, se hace una media de los valores predichos por cada uno de los árboles.

2.2.3. Redes neuronales

En esta sección se abordará un tipo de red neuronal específico que es el perceptrón multicapa (Singh y Banerjee, 2019). Este está compuesto por una serie de neuronas que se sitúan en 3 capas: la capa de entrada, las capas intermedias y la capa de salida. Cada neurona recibe un vector de valores y le aplica una combinación lineal mediante otro vector de pesos. Después, al valor obtenido se le aplica una función, que se conoce como función de activación, que da como resultado el valor de salida de dicha neurona.

En la capa de entrada hay tantas neuronas como variables. En la capa intermedia puede haber una cantidad variable de capas de neuronas. Cada neurona en las capas intermedias recibe la salida de todas las neuronas de la capa anterior y su salida va a todas las neuronas de la capa siguiente. Finalmente, en la capa de salida, si se trata de un problema de clasificación habrá tantas neuronas como clases distintas. Cada clase se corresponderá con una neurona y la clase asociada a la neurona que produzca una salida con mayor valor será la clase predicha. En un problema de regresión, en cambio, habrá una única neurona, y directamente su salida será el valor predicho. En el caso de la clasificación, a las salidas generadas por la capa de salida también se les puede aplicar una función de activación.

2.2.4. Reducción de variables

A continuación se presentarán las técnicas de reducción de variables que se van a emplear más adelante.

SelectKBest

Selecciona las k variables más relevantes según un criterio estadístico.

SelectFromModel(Random Forest)

Esta técnica emplea el modelo *Randmo Forest* para determinar qué variables son determinantes. Para ello, se observa cuánta mejora genera cada variable en la impureza del árbol cuando se usa para dividir los nodos.

SelectFromModel(Lasso)

Esta técnica es similar a la anterior, pero el modelo que se usa es una regresión lineal con regularización.

RFE(Recursive Feature Elimination)

Elimina recursivamente las variables menos relevantes según un modelo como la regresión logística o *Random Forest*.

2.3. Análisis comparativo de sistemas

A continuación se mencionarán algunos de los trabajos de referencia, así como las diferencias que presentan entre sí.

En primer lugar, (Mead et al., 2023) desarrollan un modelo de goles esperados que va más allá de los modelos convencionales. Ponen el foco en la importancia del nivel del jugador y del equipo que efectúa el disparo, así como en los factores psicológicos de los mismos. Incorporan estos factores en la creación de un modelo de goles esperados y comparan su rendimiento con el de un modelo convencional que no tiene en cuenta los factores citados. De este estudio se concluye que algunos de los parámetros incorporados resultan beneficiosos para la creación de los modelos. En concreto, el valor de mercado de los jugadores (como variable que representa la calidad del jugador) y la diferencia en el marcador en el momento del tiro (que se relaciona con el estado psicológico del lanzador) son las variables más reseñables.

En este caso se emplean las siguientes técnicas: regresión logística, perceptrón multicapa, *AdaBoost*, *random forest* y *xGBoost*. El trabajo crea un modelo distinto por cada una de las grandes ligas europeas de fútbol. La regresión logística consigue ser la mejor técnica para crear los modelos de la liga española y la liga francesa. El perceptrón multicapa se emplea para crear el mejor modelo para la liga inglesa. Finalmente, los mejores modelos para la liga italiana y para la liga alemana se consiguen utilizando *xGBoost*. En suma, las tres técnicas que un mejor rendimiento han obtenido han sido la regresión logística, el perceptrón multicapa y el *xGBoost* .

Por otro lado, (Hewitt y Karakuş, 2023) ponen el foco en la diferencia que existe en la probabilidad de que un tiro termine en gol en función de la posición del jugador. Esto es, que la probabilidad de que un mismo tiro por parte de un defensa y un delantero termine en gol tiene distinta probabilidad. Ante esto, desarrollan un modelo de goles esperados ajustado por posición, que en este caso se diferencia entre delanteros, mediocampistas y defensas. Se concluye que los jugadores más efectivos ante un mismo tiro son los delanteros, seguidos de los mediocampistas, y en última posición los defensas.

Para el desarrollo de este trabajo se emplea un modelo base que utiliza regresión logística, y otro modelo mejorado que emplea árboles de decisión y variantes del mismo.

Por último, (Madrero Pardo, 2020) presenta una comparación entre dos modelos de goles esperados: uno que no tiene en cuenta la calidad del jugador, y otro que sí la tiene en cuenta. Para ello, emplea valoraciones generales del jugador así como va-

loraciones del tiro. Tras la comparación, se puede concluir que el modelo que atiende a las individualidades técnicas de los jugadores consigue un mejor rendimiento.

Otra vez más, las técnicas empleadas en este trabajo son la regresión logística, el perceptrón multicapa y *xGBoost*.

Se puede observar que cada uno de los trabajos aborda el tema poniendo el foco en aspectos diferentes del fútbol: el nivel de jugador y el estado psicológico, la posición del jugador, y la calidad general del jugador y su calidad de tiro. Todos ellos concluyen que las características en las que hacen hincapié ayudan, en efecto, a mejorar el modelo de goles esperados.

Al centrarse en un aspecto en concreto, se desatienden los demás, por lo que los tres estudios toman unas variables base a las que les añaden las respectivas variables que creen oportunas según el foco que le quieren poner. Por tanto, si se comparan los tres enfoques, las ventajas de uno son los defectos de los otros, y viceversa.

Aunque se empleen diferentes técnicas en cada uno de los trabajos citados, el foco radica en ver la influencia que tienen las variables nuevas incorporadas.

Por otro lado, tras la revisión de las técnicas que se emplean en estos trabajos de referencia, se puede justificar la elección inicial de modelos que se ha hecho anteriormente.

2.4. Diferenciación de la propuesta

En el presente trabajo, siguiendo la línea de los estudios citados, se tratará de añadir algunas variables con el objetivo de ver si mejoran el rendimiento del modelo. Dichas variables serán principalmente posicionales y tendrán en cuenta a los jugadores cercanos al sitio del disparo, así como al posicionamiento del portero. Esto es, se tendrá en cuenta el posicionamiento de los jugadores que tienen una influencia directa o indirecta en el tiro.

Adicionalmente, se llevará a cabo una comparación del rendimiento de distintos modelos de aprendizaje automático con el objetivo de ver cuál de ellos resulta más conveniente en este caso particular. Además, se compararán las diferencias entre distintos procesamientos de los datos.

Para concluir, todo lo mencionado se abordará desde dos enfoques: uno de clasificación y otro de regresión (ver Capítulo 4). El primero seguirá la línea de los trabajos de referencia. Es decir, se creará un modelo desde cero y se tratará de predecir si un tiro va a terminar en gol o no. Es decir, se seguirá un enfoque de clasificación binaria, donde las dos clases posibles serán "gol" y "no gol". Posteriormente, se obtendrá la probabilidad de que cada tiro pertenezca a la clase "gol". Esta probabilidad será la salida del modelo.

El segundo enfoque, en cambio, se basará en las probabilidades calculadas por el modelo de *statsbomb* y se tratará de crear un modelo lo más parecido posible a este con un enfoque de regresión. Es decir, la variable a predecir será la probabilidad que le atribuye *statsbomb* a cada tiro.

Por tanto, por un lado se creará un modelo desde cero (enfoque de clasificación), y por otro, se tratará de construir un modelo que trate de asemejarse al modelo de referencia de *statsbomb* (enfoque de regresión).

2.5. Conclusiones

Para recapitular, a lo largo de este capítulo se han introducido algunos conceptos fundamentales del aprendizaje automático, más en concreto del aprendizaje supervisado. Además, se han presentado algunas de las técnicas que se van a emplear para el desarrollo del trabajo, muchas de las cuales coinciden con aquellas empleadas en los trabajos de referencia.

Se han discutido también trabajos de referencia cuyo foco reside en la incorporación de nuevas variables para mejorar el rendimiento de un modelo base. Finalmente, se ha presentado la diferenciación que ofrece este trabajo respecto de los de referencia: la incorporación de nuevas variables posicionales y la comparación de rendimiento entre diferentes técnicas y diferentes preprocesamientos de los datos. También, se van a tratar las diferencias entre crear los modelos con un modelo de referencia (enfoque de regresión) y sin él (enfoque de clasificación).

Capítulo 3

Infraestructura

En este capítulo se detallarán las herramientas y tecnologías que se han empleado para llevar a cabo el trabajo.

Para ello, la estructura que se va a seguir va a ser la siguiente: en primer lugar, se presentarán los lenguajes de programación empleados para el desarrollo del trabajo (ver sección 3.1). A continuación, se mostrarán las librerías específicas empleadas (ver sección 3.2), y finalmente, se detallará el entorno de desarrollo empleado (ver sección 3.3).

3.1. Lenguaje de programación

El lenguaje utilizado para el desarrollo del trabajo ha sido *python* por su facilidad de uso y por lo adaptado que está para crear modelos de aprendizaje supervisado. Alternativamente, se podría haber utilizado *R*, pero ofrece menos facilidades para crear modelos.

3.2. Librerías

Dentro de las librarías que ofrece *python* se han hecho uso de las siguientes.

3.2.1. Mplsoccer

Para la adquisición y presentación de los datos se ha empleado *mplsoccer* (Rowlinson y Durgapal, 2021). Se trata de una librería de *python* diseñada para la manipulación y visualización de los datos relacionados con el fútbol (aunque solamente se ha destinado para el primer propósito). Ofrece la posibilidad de acceder a eventos de los partidos, entre los que se encuentran los tiros, que son una pieza clave para el desarrollo del trabajo.

También se podrían haber usado otras librerías como *understatpy* (Varano, 2020) o *statsbombpy* (StatsBomb, 2020), que están diseñadas para el análisis de datos de los partidos de fútbol.

3.2.2. Manipulación de datos

Para la manipulación de datos se han usado las librerías *pandas* (McKinney, 2010) y *numpy* (Harris et al., 2020) que son las más utilizadas para este propósito. Entre las principales manipulaciones se incluyen la limpieza de los datos (eliminar valores nulos o corregir datos, entre otros), transformación de los datos (normalizar o estandarizar, por ejemplo) y la creación de nuevas variables.

Estas modificaciones resultan vitales porque los datos de primeras suelen estar presentados de formas muy diversas, y las técnicas de aprendizaje supervisado deben ser capaces de procesarlos correctamente. Es por ello que previamente se efectúan una serie de cambios a los datos para que, independientemente de cómo estén presentados, el algoritmo pueda interpretarlos de forma correcta y se saque el máximo partido del mismo.

Pandas

Se trata de una librería orientada a la manipulación de datos estructurados. Su principal estructura es el $dataframe$, una tabla formada por filas y columnas que facilita la agrupación, eliminación, inserción, unión y filtrado de los datos.

Numpy

Se trata de una librería de cálculo numérico que ofrece *python*. Ofrece funcionalidades de álgebra lineal y estadística y se emplea con el objetivo de optimizar las operaciones realizadas. Su utilidad aumenta conforme el volumen de los datos aumenta.

3.2.3. Modelos

Para la implementación de los modelos de aprendizaje supervisado, se ha empleado la librería $scikit-learn$ (Pedregosa et al., 2011) que es la más utilizada dado el amplio rango de modelos y funcionalidades que ofrece. Adicionalmente, se ha usado la librería *xgboost* (Chen y Guestrin, 2016) debido a la ausencia de este modelo en $scikit-learn$.

Scikit-learn

Se trata de una librería de *python* destinada a crear modelos de aprendizaje supervisado y no supervisado. Ofrece la posibilidad de crear una variedad amplia de estos modelos de forma sencilla. Además, ofrece funcionalidades para llevar a cabo el preprocesamiento de los datos, el ajuste de hiperparámetros o la validación cruzada. Cabe destacar que se integra perfectamente con las librerías empleadas para la manipulación de los datos.

xgboost

Es una librería que implementa el algoritmo de *extreme gradient boosting* que es una mejora del algoritmo del *gradient boosting*. Este algoritmo crea un árbol de decisión y realiza varias iteraciones donde mejora ese árbol reduciendo el error cometido.

3.3. Entorno de desarrollo

El trabajo se ha desarrollado en *google colab* (Research, 2025) con el objetivo de ser accesible desde cualquier dispositivo y por la facilidad que ofrece para desarrollar este tipo de trabajos.

Se trata de un entorno basado en la nube que sirve para ejecutar código de *python*. Los archivos se estructuran en celdas de diferentes tipos: código, resultados y *markdown* para texto con formato, lo cual resulta útil para tareas de análisis de datos y aprendizaje automático.

No obstante, al tratarse de un recurso remoto, la capacidad de la CPU y de la RAM son limitadas, lo cual podría suponer un problema en caso de que la demanda de los recursos fuera alta.

Es por esto último que se podría haber optado por otras alternativas, como trabajar en local desde Jupyter Notebook. No obstante, dado que los requerimientos computacionales no han sido excesivos, se ha optado por la comodidad que ofrece la primera opción.

3.4. Conclusiones

En suma, el lenguaje de programación que se va a emplear es *python*, como es común en trabajos similares. Adicionalmente, la biblioteca de adquisición de los datos será *mplsoccer* y las bibliotecas principales para la manipulación de dichos datos serán *pandas* y *numpy*. Para la creación de los modelos, se emplearán las librerías $scikit-learn$ y *xgboost*. Para finalizar, el entorno de desarrollo elegido ha sido *colab*, ya que cumple con los requerimientos necesarios y resulta práctico.

Capítulo 4

Modelos de aprendizaje máquina

En este capítulo se detallará paso a paso el proceso que se ha seguido para construir los modelos de goles esperados.

En primer lugar, se ha creado el conjunto de datos (ver sección 4.1), y se han añadido las nuevas variables posicionales. A continuación, se ha preprocesado dicho conjunto (ver sección 4.2).

Una vez hecho esto, se cumplen las condiciones necesarias para entrenar los modelos. Para ello, como se ha presentado al final del capítulo 2, se han seguido dos enfoques.

Para el enfoque de clasificación, se ha creado un modelo donde la variable objetivo ha sido si el disparo ha terminado en gol o no (ver sección 4.3. De esta forma, se ha creado un modelo de goles esperados desde cero, sin tomar ningún modelo preexistente como referencia.

Para el enfoque de regresión, se ha creado un modelo con la variable de goles esperados que ofrece *statsbomb* como variable objetivo (ver sección 4.4. Es decir, el objetivo ha residido en crear un modelo lo más similar posible al modelo de referencia de *statsbomb*.

En ambos enfoques se han entrenado distintos modelos y después se han ajustado para tratar de mejorar su rendimiento. Además, se han creado los modelos descritos de dos formas diferentes: solamente con las variables ofrecidas por el conjunto de datos que se ha usado, y con las variables nuevas posicionales, además de las anteriores.

Finalmente, se ha realizado el análisis de reducción de variables (ver sección 4.5).

4.1. Creación del conjunto de datos

Para comenzar, se han obtenido los datos de los partidos de La Liga desde la temporada 2004/2005 hasta la temporada 2020/2021 mediante el repositorio (StatsBomb, 2025). En él se encuentran algunos de los partidos de esas temporadas, en concreto, 834 partidos. Cabe destacar que no contiene todos los partidos, pues por cada temporada hay 38 jornadas con 10 partidos por jornada, por lo que hay un total de 380 partidos por temporada. Para ello, se han extraído los identificadores de los

partidos de la liga española entre las fechas mencionadas.
A continuación, se han extraído todas las variables significativas de todos los tiros de los partidos que se han obtenido, tal y como se mostrarán a continuación.
Adicionalmente, se han añadido las variables posicionales que se presentarán más adelante. De esta forma, se tendrán todas las variables en un mismo conjunto de datos, con lo cual será más sencillo trabajar.

4.1.1. Variables del conjunto de datos

Las variables que se han obtenido del repositorio han sido las siguientes:

- `freeze`: Lista de jugadores cercanos al disparo. Será clave para crear las variables posicionales que se detallarán más adelante.
- `id`: Identificador único del disparo.
- `match_id`: Identificador del partido.
- `team_id`: Identificador del equipo.
- `team_name`: Nombre del equipo del lanzador.
- `player_name`: Nombre del lanzador.
- `minute`: Minuto del partido.
- `play_pattern_id`: Identificador del tipo de jugada.
- `position_id`: Identificador de la posición del lanzador.
- `body_id`: Parte del cuerpo utilizada en el disparo.
- `technique_id`: Técnica de disparo.
- `sub_type_id`: Identificador del subtipo del tiro.
- `sub_type_name`: Nombre del subtipo del tiro.
- `outcome_id`: Resultado del disparo.
- `x`, `y`: Coordenadas del sitio de disparo.
- `end_x`, `end_y`, `end_z`: Coordenadas del destino del disparo.
- `under_pressure`: Indica si el lanzador estaba bajo presión.
- `shot_statsbomb_xg`: Valor de gol esperado (xG) calculado por StatsBomb.
- `shot_first_time`: Si el disparo fue de primeras.
- `aerial_won`: Si el disparo fue tras ganar un duelo aéreo.
- `shot_one_on_one`: Mano a mano contra el portero.

- `shot_open_goal`: Disparo a puerta vacía.
- `shot_deflected`: Disparo desviado por otro jugador.
- `shot_redirect`: Cambio de trayectoria de otro tiro.
- `shot_follows_dribble`: Disparo justo después de un regate.

Cabe destacar que las dimensiones del campo son 120 en el eje x y 80 en el eje y. Además, no se va a hacer uso de las variables end_x, end_y, $shot\_redirect$ y $shot\_deflected$, puesto que no se conocen al momento del disparo, sino que se conocen después, y el modelo de goles esperados solamente atiende a aquellas variables que son conocidas en el momento del disparo. A raíz de este detalle, han surgido otros tipos de métricas similares a los goles esperados, pero que tienen en cuenta las variables conocidas después del disparo.

4.1.2. Nuevas variables

Además de las variables detalladas, se han creado las siguientes variables posicionales con el objetivo de mejorar el rendimiento del modelo.

- `distance`: Distancia desde el lugar del disparo al centro de la portería.
- `angle`: Ángulo formado entre los dos postes de la portería y el punto de disparo.
- `cuantos_cerca`: Número de jugadores rivales cercanos al lanzador, ponderado por distancia (los más próximos tienen mayor peso).
- `cuantos_en_trayectoria`: Número de jugadores rivales en la trayectoria del disparo hacia la portería, también ponderado por distancia.
- `distancia_portero_porteria`: Distancia del portero a la portería.
- `distancia_portero_jugador`: Distancia del portero al lanzador.
- `angulo_desviado_portero`: Grado de desviación del portero respecto al ángulo ideal de tiro del lanzador.

4.2. Preprocesamiento de los datos

Una vez se ha obtenido el conjunto de datos en su forma original, se deben hacer algunas modificaciones antes de poder usarlo para entrenar los modelos.

4.2.1. Limpieza de datos

En primer lugar, se han eliminado las variables que no se van a emplear para las creaciones de los modelos, pues nada tienen que ver con las variables del disparo. Estas variables son *freeze*, *id*, *match_id*, *team_name*, *team_id*, *player_name*, *player_id* y *sub_type_name*. La variable *freeze* ya se ha utilizado para calcular las nuevas variables posicionales, por lo que ya no es necesaria. La variable *sub_type_name* no resulta muy relevante porque esa información se extrae de *play_pattern_id*. Las demás variables, por el contrario, son identificadores y nombres que no van a tener relevancia en la forma de predecir la probabilidad de que un tiro termine en gol.

Completar los valores faltantes

A continuación, se han analizado los valores de las variables y se ha observado que en el caso de las variables *under_pressure*, *shot_first_time*, *shot_deflected*, *shot_redirect*, *shot_follows_dribble*, *shot_one_on_one*, *aerial_won* y *shot_open_goal* solamente aparecen indicadas en caso de ser ciertas. Por tanto, en caso de que no aparezca nada se han completado con 0 y en caso de aparecer el valor 1 o *True* se ha optado por emplear el 1 para que todas estas variables queden con el mismo formato de 1 si son ciertas en el tiro, y 0 si no lo son.

Eliminar los valores nulos

Después de hacer las modificaciones previas, se han procedido a eliminar las filas con valores nulos. Se ha pasado de 20.431 filas a 20.233, es decir, se han eliminado un total de 198 filas. Esto supone el 0,97 % del total de filas.

4.2.2. Análisis de las variables

Para conocer en más detalle las variables de las que se dispone, se muestran el diagrama de dispersión (ver Fig. 4.1) y la matriz de correlación (ver Fig. 4.2) de las variables. Además, se muestran los valores numéricos de la dispersión de las variables del conjunto de datos (ver Fig. 4.3). Para las variables categóricas, se muestran las frecuencias de cada valor (ver Fig. 4.4). Todas estas figuras representan un subconjunto de todos los datos con el objetivo de que se puedan apreciar fácilmente.

En cuanto a la distribución de las variables, como se puede apreciar en la diagonal principal de la figura 4.1, las variables siguen distribuciones dispares, lo cual será importante tener en cuenta en algunas de las transformaciones que se efectuarán posteriormente.

Se puede apreciar que existen ciertas parejas de variables cuya correlación es superior al 0,5 en valor absoluto. Como es el caso de (*aerial_won*, *under_pressure*), (*shot_open_goal*, *shot_statsbomb_xg*), (*distance*, *x*), (*distance*, *shot_statsbomb_xg*), (*angle*, *x*), (*angle*, *shot_statsbomb_xg*), (*angle*, *distance*), (*distancia_portero_jugador*, *x*), (*distancia_portero_jugador*,*shot_statsbomb_xg*), (*distancia_portero_jugador*, *distance*), (*distancia_portero_jugador*, *angle*) . En espe-

cial, las tres variables *distance*, *distancia_portero_jugador* y x están muy correlacionadas, lo cual podría indicar que, incluyendo una de las tres, se captaría la amplia mayoría de la información que ofrecen. No obstante, no se va a optar por eliminar ninguna hasta el análisis de reducción de variables que se detallará más adelante.

En cuanto a la distribución de las variables, entre otras cosas se puede destacar que la media de disparos se lanza desde las coordenadas (103,72, 39,47), es decir, a unos 16 metros de la portería y prácticamente centrado, teniendo en cuenta las dimensiones del campo (120 x 80).

Tras analizar la frecuencia de los valores de las variables categóricas, se puede apreciar que en la mayoría de ellas hay valores preponderantes y valores infrarrepresentados. Idealmente, se podría optar por equilibrar estas descompensaciones. Una forma para abordarlo podría ser el submuestreo de la clase principal. No obstante, esto podría llevar a que las filas del conjunto de datos se redujeran significativamente, lo cual podría degradar el rendimiento final de los modelos. Es por ello que no se ha decidido hacer nada ante esto. Resulta especialmente importante la variable *outcome_id* que en el enfoque de clasificación será la variable a predecir. Se puede apreciar que la clase de "*gol*" está descompensada respecto a las demás clases, lo cual se debe tener en cuenta a la hora de analizar los resultados.
Tras estas modificaciones, el conjunto de datos consta de 20,232 filas y de 20 variables que se han obtenido de la fuente citada, y 7 variables posicionales que han sido creadas desde cero.

4.2.3. Codificación

Se ha aplicado $one - hot\ encoding$ a las variables categóricas. La razón detrás de esto es que potencialmente algunas técnicas empleadas se pueden ver beneficiadas por esta transformación de los datos. Esto hace que para cada valor distinto de las variables categóricas se cree una nueva columna. Por tanto, si un dato tiene un valor x en una variable categórica, en la nueva columna x de la variable aparecerá como $True$ y en el resto de columnas creadas para la misma variable tomará el valor $False$.

En este caso aplicar esta transformación puede resultar beneficioso de cara a mejorar el rendimiento de los modelos resultantes, ya que no existe una ordenación inherente en las variables categóricas.

Esto se hará para todas las variables categóricas. Por tanto, el número de columnas aumentará considerablemente. Es por ello que se creará un conjunto de datos sin aplicar esta técnica y otro aplicándola, para ver si compensa.

4.2.4. Normalización y estandarización

A las variables numéricas se les ha aplicado una normalización o estandarización, puesto que no todas las variables se encuentran en el mismo rango de valores y las distribuciones que siguen son dispares. Otra vez, para ver qué transformación es la que más compensa, se ha creado un conjunto con los datos normalizados, y otro con los datos estandarizados.

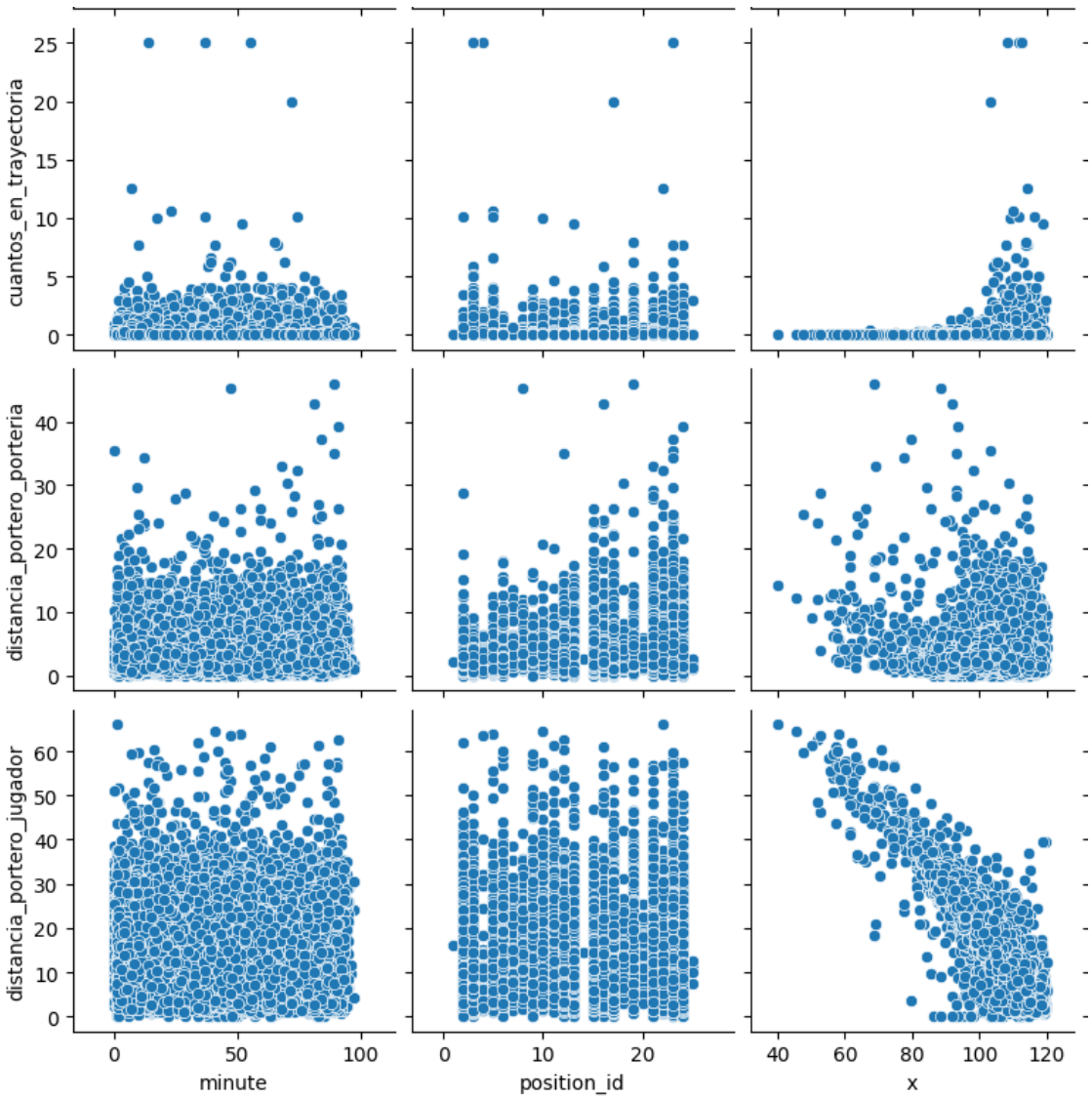


Figura 4.1: Diagrama de dispersión

Figura 4.2: Diagrama de correlación

	minute	play_pattern_id	position_id	body_part_id	sub_type_id	x	y
count	20232.000000	20232.000000	20232.000000	20232.000000	20232.000000	20232.000000	20232.000000
mean	48.117734	2.756228	16.419187	38.934658	85.438513	103.723596	39.469790
std	26.403233	1.849186	6.549703	1.900568	6.053783	8.800204	10.104286
min	0.000000	1.000000	1.000000	37.000000	61.000000	40.000000	0.700000
25%	26.000000	1.000000	12.000000	38.000000	87.000000	97.600000	31.700000
50%	49.000000	3.000000	17.000000	40.000000	87.000000	105.100000	39.500000
75%	71.000000	4.000000	22.000000	40.000000	87.000000	110.700000	47.100000
max	97.000000	9.000000	25.000000	70.000000	88.000000	120.000000	78.800000

Figura 4.3: Tabla de distribución de las variables

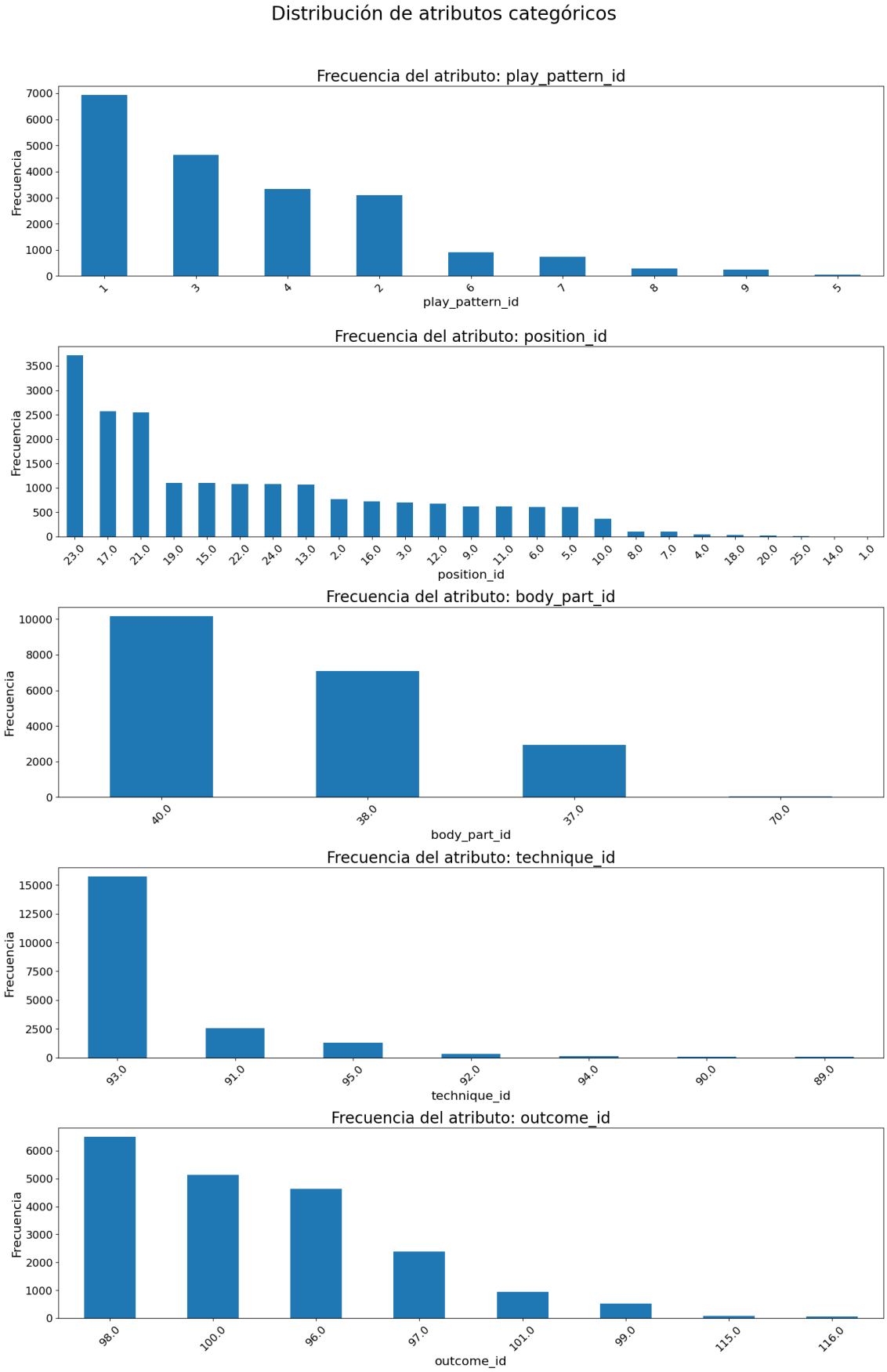


Figura 4.4: Frecuencia de valores de variables categóricas

4.2.5. Todos los conjuntos de datos

En suma, los conjuntos de datos que se han creado han sido los siguientes: tanto para el $one-hot\ encoding$ como para la estandarización y normalización, se han creado dos conjuntos de datos distintos, uno con las variables iniciales y otro con las variables posicionales. Por tanto, en función de si se le ha aplicado $one-hotencoding$, si ha sido estandarizado o normalizado y si usa todas las variables o solo las iniciales, se han obtenido 12 conjuntos de datos distintos (6 con variables del conjunto de datos y 6 con variables nuevas), con los cuales se probará el rendimiento de los modelos y se verá si hay diferencias entre las distintas configuraciones.

4.3. Clasificación

A continuación se mostrará el proceso de entrenamiento y de ajuste de los modelos que no toman como referencia el valor de goles esperados calculado por *statsbomb*. Como se ha explicado anteriormente, este enfoque se tratará mediante un enfoque de clasificación binaria("gol" o "no gol"). Cada técnica tendrá como objetivo predecir si un tiro ha terminado en gol o no. Para ello, internamente calcularán la probabilidad que tiene cada tiro de pertenecer a la clase positiva "gol", y precisamente esa probabilidad será la que se tome como el valor de gol esperado de cada tiro. Es decir, lo importante no va a ser saber si un tiro se predice como gol o no gol, sino obtener la probabilidad que tiene de gol.

Cabe mencionar que las variables que se emplearán serán las siguientes:*minute*, *play_pattern_id*, *position_id*, *body_id*, *sub_type_id*, *sub_type_name*, *x*, *y*, *under_pressure*, *technique_id*, *shot_first_time*, *aerial_won*, *shot_one_on_one*, *shot_open_goal*, *shot_follows_dribble*, *outcome_id*

Estas variables han sido explicadas en las secciones 4.1 y 4.2. En este sentido, cabe destacar que a la variable *outcome_id* se le ha aplicado una transformación para el caso de clasificación. Como va a ser la variable a predecir, se tomará como clase "*gol*" aquellos tiros que han terminado en gol y como "*nogol*" todas las demás clases.

4.3.1. Entrenamiento

En primer lugar, se ha elegido una serie de modelos distintos para probar las diferencias de rendimiento entre ellos. Para ello, se han entrenado con los parámetros por defecto los siguientes modelos.

- Perceptrón multicapa
- Árbol de decisión
- Random Forest
- XGBoost
- KNN

- Regresión logística

Cada uno de los modelos ha sido entrenado con todos los conjuntos de datos creados, con el objetivo de ver las diferencias que se presentan entre los modelos y entre los distintos preprocesamientos de los datos.

Por otro lado, la partición que se ha hecho de entrenamiento y prueba ha sido del 70 %/30 %, puesto que da lugar a suficientes datos de entrenamiento, y también se prueba sobre una cantidad considerable de los datos. Se podría haber cambiado esta proporción, por ejemplo al 80 %/20 %, o a alguna combinación intermedia.

4.3.2. Ajuste

Tras realizar el entrenamiento de los modelos en conjuntos de datos preprocesados de distinta forma, se ha obtenido el conjunto de datos con el que se ha logrado un mejor rendimiento, y se han ajustado sus hiperparámetros para tratar de obtener un mejor rendimiento. Para ello, se ha llevado a cabo una búsqueda en rejilla de hiperparámetros, como se aprecia en la figura 4.5.

La proporción de datos de entrenamiento y de prueba ha sido la misma, es decir, 70 % para entrenamiento y 30 % para prueba. Para llevar a cabo la validación se ha optado por usar validación cruzada con 5 $folds$. Por tanto, solamente se ha empleado el conjunto de entrenamiento y el conjunto de prueba.

A continuación se detallarán algunos de los hiperparámetros y valores sobre los que se ha hecho la búsqueda en rejilla para cada técnica.

Para el perceptrón multicapa se han considerado el número de capas y el número de neuronas por capa, la función de activación, la constante de regularización y la tasa de aprendizaje.

Para el KNN el número de vecinos y si se pondera por distancia o no. En el caso de la regresión logística se han considerado el método de regularización y el inverso de la constante de regularización.

En *xGBoost* y *Random Forest* el número de árboles, la máxima profundidad de ellos y el tamaño mínimo de las hojas. Para los árboles de decisión también se ha tenido en cuenta la máxima profundidad de los árboles y el tamaño mínimo de las hojas.

Los valores que se han elegido han surgido de una serie de pruebas para tratar de encontrar el rango óptimo de los valores de cada hiperparámetro (ver Fig. 4.5).

4.4. Regresión

En esta sección se tratará de crear un modelo con el valor de gol esperado que ofrece *statsbomb* como variable a predecir. Por ello, se tratará desde un enfoque de regresión.

4.4.1. Entrenamiento

Las variables que se van a emplear van a ser las mismas que en el caso del enfoque de clasificación, a excepción de 2 variables. Por un lado, se hará uso de la varia-

```
'MLP': {
    'model': MLPClassifier(random_state=42),
    'params': {
        'hidden_layer_sizes': [(100,50,25)],
        'activation': ['relu'],
        'solver': ['adam'],
        'alpha': [ 0.01, 0.001],
        'max_iter': [50],
        'tol': [1e-5],
        'n_iter_no_change': [100],
        'learning_rate_init': [0.0001, 0.001],
        'verbose': [True]
    }
},

  'Decision Tree': {
      'model': DecisionTreeClassifier(random_state=42),
      'params': {
      'max_depth': [None, 10, 20],
      'min_samples_split': [2, 5, 10],
      'min_samples_leaf': [1, 2, 4],
      'criterion': ['gini']
      }
  },
  'Random Forest': {
      'model': RandomForestClassifier(random_state=42),
      'params': {
      'n_estimators': [100],
      'max_depth': [None, 10, 20],
      'min_samples_split': [2, 5, 10],
      'min_samples_leaf': [1, 2, 4],
      'criterion': ['gini']
      }
  },
  'XGBoost': {
      'model': XGBClassifier(random_state=42),
      'params': {
      'n_estimators': [100],
      'max_depth': [3, 5, 7],
      'learning_rate': [0.01, 0.1, 0.2],
      'subsample': [0.8, 0.9, 1.0],
      'colsample_bytree': [0.8, 0.9, 1.0]
      }
  },
  'KNN': {
      'model': KNeighborsClassifier(),
      'params': {
      'n_neighbors': [3, 5, 7],
      'weights': ['uniform', 'distance'],
      'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
      }
  },
  'Logistic Regression': {
      'model': LogisticRegression(random_state=42),
      'params': {
      'C': [0.001, 0.01, 0.1],
      'penalty': ['l1', 'l2'],
      'solver': ['liblinear', 'saga']
      }
```

Figura 4.5: Hiperparámetros para la búsqueda en rejilla

ble *shot_statsbomb_xg*. Además, no se va a hacer uso de *outcome id* ya que no sería lícito predecir la probabilidad de que un tiro termine en gol sabiendo cómo ha terminado. Por tanto, estas serán las variables que se usarán en esta sección: *minute*, *play_pattern_id*, *position_id*, *body_id*, *sub_type_id*, *sub_type_name*, *x*, *y*, *under_pressure*, *technique_id*, *shot_first_time*, *aerial_won*, *shot_one_on_one*, *shot_open_goal* y *shot_follows_dribble*
En este caso, se van a emplear las mismas técnicas que en el caso de la clasificación, a diferencia de la regresión logística, que se va a reemplazar por la regresión lineal, dado que en este caso se va a emplear para regresión. En el resto de técnicas se empleará la versión de regresión en vez de la de clasificación.

Como en el enfoque de clasificación, cada uno de los modelos ha sido entrenado con todos los conjuntos de datos que se han creado, con el objetivo de ver las diferencias que se presentan entre los modelos, entre los distintos preprocesamientos de los datos, y entre la incorporación de las nuevas variables que se han creado.

4.4.2. Ajuste

Tras realizar el entrenamiento de los modelos en conjuntos de datos preprocesados de distinta forma, se ha obtenido el conjunto de datos con el que mejor rendimiento, y se han ajustado sus hiperparámetros para tratar de obtener un mejor rendimiento. Para ello, se ha llevado a cabo una búsqueda en rejilla de hiperparámetros, como se aprecia en la figura 4.6.

Nuevamente, se ha seguido la misma proporción de datos para el conjunto de entrenamiento y prueba, y se ha hecho uso de validación cruzada. En este caso, se han considerado los mismos hiperparámetros que en el caso de clasificación. Para la regresión lineal (que antes no se ha usado) no se ha considerado ningún hiperparámetro para la búsqueda en rejilla, puesto que son muy específicos y no se ha visto necesario hacerlo.

4.5. Reducción de variables

Dado el alto número de variables que hay en el conjunto de datos, se va a llevar a cabo un análisis de las variables más importantes y se va a probar cómo varía el rendimiento en el mejor modelo obtenido en cada uno de los dos enfoques. Para ello, se ha optado por una serie de técnicas diversas de reducción de variables para, de esta forma, ver cuál funciona mejor para este problema. En particular se han seleccionado:

- SelectKBest
- SelectFromModel(Random Forest)
- SelectFromModel(Lasso)
- RFE

```
'MLP':{
    'model': MLPRegressor(random_state=42),
    'params': {
        'hidden_layer_sizes': [(100,100,100), (20,20,20,20,20)],
        'activation': ['relu', 'tanh'],
        'solver': ['adam'],
        'alpha': [0.01],
        'max_iter': [1000],
        'tol': [1e-5],
        'n_iter_no_change': [100],
        'learning_rate_init': [0.0003],
        'verbose': [True]
    }
},
'KNN': {
    'model': KNeighborsRegressor(),
    'params': {
        'n_neighbors': [3, 5, 7, 9, 11],
        'weights': ['uniform', 'distance'],
        'p': [1, 2]
    }
},

'RandomForest': {
    'model': RandomForestRegressor(random_state=random_state),
    'params': {
        'n_estimators': [50, 100],
        'max_depth': [None] + list(range(5, 10, 5)),
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto', 'sqrt', 'log2']
    }
},

'XGBoost': {
    'model': xgb.XGBRegressor(objective='reg:squarederror', random_state=random_state),
    'params': {
        'n_estimators': [200],
        'learning_rate': [0.065],
        'max_depth': range(7,8),
        'min_child_weight': range(3,4),
        'subsample': [0.8],
        'colsample_bytree': [0.7],
        'tree_method': ['auto'],
        'booster': ['gbtree']

    }
},
'DecisionTree': {
    'model': DecisionTreeRegressor(random_state=random_state),
    'params': {
        'max_depth': [None] + list(range(5, 10, 5)),
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['auto', 'sqrt', 'log2'],
        'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
    }
},
'LinearRegression': {
    'model': LinearRegression(),
    'params': {}
}
```

Figura 4.6: Hiperparámetros para la búsqueda en rejilla

Dado que en los trabajos de referencia no se mencionan técnicas de reducción de variables, la elección de estas ha sido ciertamente arbitraria, simplemente procurando que fueran diversas.

4.6. Conclusión

A lo largo de este capítulo se ha detallado el proceso realizado para crear los modelos. En primer lugar, se han mostrado qué variables se han elegido para formar el conjunto de datos. Además, se han concretado las variables posicionales que se han creado para tratar de mejorar el rendimiento de los modelos. Acto seguido, se han presentado las modificaciones que se le han hecho a estos datos: eliminación de valores nulos, adición de valores ausentes, codificación de los datos, normalización y estandarización. Después se han explicado los dos enfoques que se han seguido: clasificación y regresión. En cada uno de ellos se han detallado las técnicas que se van a usar, cómo se van a usar y cómo se va a llevar a cabo el ajuste de los modelos resultantes. Finalmente, se ha mencionado cómo se va a realizar la reducción de variables.

Capítulo 5

Evaluación

En este capítulo se mostrarán los resultados de los modelos descritos durante el capítulo anterior. En primer lugar, se tratará el enfoque de clasificación y después el de regresión. En cada uno de los dos enfoques se empezará por mostrar los resultados con las variables extraídas de la fuente utilizada. Después se analizará qué sucede cuando se añaden las nuevas variables posicionales. Finalmente se verá el impacto que tiene en los modelos resultantes la reducción de variables.

5.1. Clasificación

- Exactitud
- Precisión
- Exhaustividad
- F1
- AUC-ROC

Cabe destacar que el modelo de *statsbomb* se va a mostrar como referencia para poder apreciar la diferencia de rendimiento que existe entre los modelos que se han creado y el de referencia.

5.1.1. Sin variables nuevas

A continuación se mostrarán las métricas de los modelos creados a partir de las variables que ya se encontraban en la base de datos de la que se ha obtenido la información de los tiros. Como se ha introducido anteriormente, se mostrarán los resultados para los distintos conjuntos de datos.

Conjunto base

Evaluación del conjunto de datos sin normalizar, estandarizar y sin emplear *one − hot encoding*. Los resultados se muestran en la figura 5.1.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000714
MLP	0.893081	0.808824	0.079595	0.144928	0.779229	2.106997
Decision Tree	0.819110	0.243380	0.279305	0.260108	0.583880	0.121335
Random Forest	0.891928	0.649573	0.109986	0.188119	0.758109	2.008609
XGBoost	0.889786	0.556122	0.157742	0.245772	0.768821	0.322362
KNN	0.878583	0.392523	0.121563	0.185635	0.659163	0.788090
Logistic Regression	0.886491	0.550000	0.015919	0.030942	0.743282	0.147750

Figura 5.1: Métricas del conjunto base con variables base

Conjunto con one-hot encoding

Evaluación del conjunto de datos empleando $one - hot\ encoding$. Los resultados se muestran en la figura 5.2.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000668
MLP	0.892092	0.700000	0.091172	0.161332	0.764282	13.643185
Decision Tree	0.830478	0.271003	0.289436	0.279916	0.594709	0.158448
Random Forest	0.891433	0.615942	0.123010	0.205066	0.747146	4.095863
XGBoost	0.887974	0.527094	0.154848	0.239374	0.767703	1.233455
KNN	0.881384	0.431280	0.131693	0.201774	0.667412	3.316995
Logistic Regression	0.892257	0.728395	0.085384	0.152850	0.736410	2.030346

Figura 5.2: Métricas del conjunto con one-hot encoding

Conjunto estandarizado

Evaluación del conjunto de datos estandarizado. Los resultados se muestran en la figura 5.3.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000631
MLP	0.893245	0.811594	0.081042	0.147368	0.780147	3.215814
Decision Tree	0.818616	0.241814	0.277858	0.258586	0.582971	0.102537
Random Forest	0.891928	0.652174	0.108538	0.186104	0.758490	2.398519
XGBoost	0.889786	0.556122	0.157742	0.245772	0.768821	0.247017
KNN	0.880066	0.402116	0.109986	0.172727	0.642733	2.314388
Logistic Regression	0.890445	0.782609	0.052098	0.097693	0.744784	0.179903

Figura 5.3: Métricas del conjunto estandarizado

Conjunto normalizado

Evaluación del conjunto de datos normalizado. Los resultados se muestran en la figura 5.4.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000622
MLP	0.893245	0.802817	0.082489	0.149606	0.716765	2.259326
Decision Tree	0.819110	0.242731	0.277858	0.259109	0.583250	0.136421
Random Forest	0.892257	0.658120	0.111433	0.190594	0.758010	2.789653
XGBoost	0.889786	0.556122	0.157742	0.245772	0.768821	0.229299
KNN	0.878583	0.342466	0.072359	0.119474	0.595619	1.111124
Logistic Regression	0.893245	0.802817	0.082489	0.149606	0.725343	0.123635

Figura 5.4: Métricas del conjunto normalizado

Conjunto estandarizado con one-hot encoding

Evaluación del conjunto de datos estandarizado y empleando one-hot encoding. Los resultados se muestran en la figura 5.5.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000709
MLP	0.873806	0.390029	0.192475	0.257752	0.739472	29.380828
Decision Tree	0.831137	0.272480	0.289436	0.280702	0.595080	0.151901
Random Forest	0.891598	0.618705	0.124457	0.207229	0.747427	2.326077
XGBoost	0.887974	0.527094	0.154848	0.239374	0.767703	0.462470
KNN	0.878913	0.386598	0.108538	0.169492	0.640304	1.851220
Logistic Regression	0.893081	0.791667	0.082489	0.149410	0.759005	0.098837

Figura 5.5: Métricas del conjunto estandarizado con one-hot encoding

Conjunto normalizado con one-hot encoding

Evaluación del conjunto de datos normalizado empleando one-hot encoding. Los resultados se muestran en la figura 5.6.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000644
MLP	0.883855	0.465347	0.136035	0.210526	0.707938	29.318050
Decision Tree	0.830643	0.270748	0.287988	0.279102	0.594171	0.154782
Random Forest	0.891598	0.618705	0.124457	0.207229	0.746379	2.333511
XGBoost	0.887974	0.527094	0.154848	0.239374	0.767703	0.436370
KNN	0.879407	0.354610	0.072359	0.120192	0.601139	1.960067
Logistic Regression	0.893081	0.791667	0.082489	0.149410	0.758251	0.182338

Figura 5.6: Métricas del conjunto normalizado con one-hot encoding

En este enfoque de clasificación, la métrica a la que más importancia se le va a dar va a ser a $ROC - AUC$ ya que da una representación más real de cómo de bien distingue los tiros que terminan en gol de los que no. Si se priorizara otra métrica como el $F1$, el rendimiento podría ser engañoso porque es más sensible a la descompensación de clases. Y como se ha visto anteriormente, en este caso las clases están desbalanceadas.

Se puede apreciar que las diferencias que existen entre los distintos tipos de preprocesamiento dependen del modelo. Por ejemplo, *xGBoost* apenas cambia una milésima desde el mejor rendimiento hasta el peor en $ROC - AUC$, por tanto, se

puede concluir que en este caso el tipo de preprocesamiento no es apenas relevante. En cambio, en otros modelos como KNN sí que existen diferencias algo más notorias. En concreto, en el peor de los casos obtiene 0,59 y en el mejor 0,66. El conjunto de datos con el que mejores rendimientos se ha conseguido ha sido con el estandarizado, pero tampoco dista mucho de los rendimientos obtenidos con el conjunto base. Por tanto, en general en este caso las transformaciones de los datos no han supuesto un cambio significativo en la mayoría de los modelos.

En cuanto a las diferentes técnicas empleadas, se puede apreciar que sí existe una mayor disparidad entre ellas. El mejor modelo se ha obtenido con el perceptrón multicapa y el $XGBoost$, $RandomForest$ y la regresión logística también han tenido un desempeño similar. El árbol de decisión y el KNN han obtenido el peor rendimiento.

Cabe destacar que ninguno ha obtenido un rendimiento muy cercano al modelo de referencia, aunque no se han quedado muy lejos. La mejor configuración ha obtenido un valor de $ROC - AUC$ de 0,78 con el perceptrón multicapa sobre el conjunto estandarizado, mientras que el modelo de referencia ha obtenido 0,82.

Los hiperparámetros se han ajustado sobre el conjunto estandarizado, que es donde se ha obtenido un mejor rendimiento. Los resultados de los modelos ajustados de dicho ajuste se pueden apreciar en la figura 5.7.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE:	ROC AUC	Tiempo
MLP	0.893081	0.828125	0.076700	0.140397	0.783136	22.311504
Decision Tree	0.862109	0.332569	0.209841	0.257320	0.645020	0.265787
Random Forest	0.892422	0.672727	0.107091	0.184769	0.765047	5.677519
XGBoost	0.894399	0.701613	0.125904	0.213497	0.791617	8.129715
KNN	0.880066	0.402116	0.109986	0.172727	0.642733	1.054627
Logistic Regression	0.892916	0.788732	0.081042	0.146982	0.747098	0.363277

Figura 5.7: Métricas de los modelos ajustados con variables base

Se puede apreciar que tras el ajuste de hiperparámetros ha habido un ligero aumento en el rendimiento de algunos de los modelos. Sobre todo, cabe destacar el rendimiento que ha obtenido $XGBoost$, puesto que ha pasado del 0,768 al 0,791 en la métrica $ROC - AUC$. En el resto de técnicas, en cambio, no se ha podido apreciar una gran diferencia.

5.1.2. Con variables nuevas

A continuación se mostrarán las métricas de los modelos creados a partir de las variables posicionales que se han añadido. Como se ha hecho anteriormente, se mostrarán los resultados para los distintos conjuntos de datos.

Conjunto base

Se trata del conjunto de datos sin normalizar, estandarizar y sin emplear $one - hot\ encoding$. Los resultados se muestran en la figura 5.8.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000659
MLP	0.892092	0.642857	0.117221	0.198286	0.789889	1.060118
Decision Tree	0.830313	0.285714	0.327062	0.304993	0.611012	0.373167
Random Forest	0.895717	0.674699	0.162084	0.261377	0.793964	5.606557
XGBoost	0.887809	0.516892	0.221418	0.310030	0.792538	0.458901
KNN	0.877924	0.402344	0.149059	0.217529	0.686820	1.518617
Logistic Regression	0.890774	0.679487	0.076700	0.137841	0.798236	0.391454

Figura 5.8: Métricas del conjunto base con variables posicionales

Conjunto con one-hot encoding

Se trata del conjunto de datos empleando $one - hot\ encoding$. Los resultados se muestran en la figura 5.9.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.001243
MLP	0.890115	0.547244	0.201158	0.294180	0.802823	5.806082
Decision Tree	0.831796	0.287918	0.324168	0.304969	0.610587	0.608945
Random Forest	0.895387	0.677215	0.154848	0.252061	0.796740	4.730657
XGBoost	0.886491	0.503546	0.205499	0.291881	0.792764	2.693190
KNN	0.878583	0.416058	0.164978	0.236269	0.694278	1.928111
Logistic Regression	0.891433	0.681818	0.086831	0.154044	0.792278	1.165573

Figura 5.9: Métricas del conjunto con one-hot encoding

Conjunto estandarizado

Se trata del conjunto de datos estandarizado. Los resultados se muestran en la figura 5.10.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.001139
MLP	0.893575	0.777778	0.091172	0.163212	0.809341	2.263647
Decision Tree	0.830478	0.275266	0.299566	0.286902	0.599123	0.324401
Random Forest	0.896705	0.686047	0.170767	0.273465	0.801451	4.255280
XGBoost	0.887315	0.513619	0.191027	0.278481	0.789163	0.466810
KNN	0.880395	0.435897	0.172214	0.246888	0.694496	1.341126
Logistic Regression	0.893904	0.671533	0.133140	0.222222	0.808210	0.146776

Figura 5.10: Métricas del conjunto estandarizado

Conjunto normalizado

Se trata del conjunto de datos normalizado. Los resultados se muestran en la figura 5.11.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000718
MLP	0.893410	0.661765	0.130246	0.217654	0.804773	3.228000
Decision Tree	0.830313	0.285714	0.327062	0.304993	0.611012	0.369117
Random Forest	0.895881	0.678788	0.162084	0.261682	0.793732	4.787725
XGBoost	0.887809	0.516892	0.221418	0.310030	0.792538	3.635147
KNN	0.879901	0.427481	0.162084	0.235047	0.676436	1.334217
Logistic Regression	0.888962	0.560284	0.114327	0.189904	0.784699	0.148829

Figura 5.11: Métricas del conjunto normalizado

Conjunto estandarizado con one-hot encoding

Se trata del conjunto de datos estandarizado y que emplea one-hot encoding. Los resultados se muestran en la figura 5.12.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.000635
MLP	0.873970	0.418860	0.276411	0.333043	0.750882	41.538049
Decision Tree	0.831466	0.280423	0.306802	0.293020	0.602834	0.407996
Random Forest	0.897199	0.708075	0.164978	0.267606	0.795466	3.988656
XGBoost	0.888303	0.523636	0.208394	0.298137	0.793697	0.726195
KNN	0.879077	0.428094	0.185239	0.258586	0.702987	3.181314
Logistic Regression	0.896046	0.689873	0.157742	0.256773	0.815071	0.965580

Figura 5.12: Métricas del conjunto estandarizado con one-hot encoding

Conjunto normalizado con one-hot encoding

Se trata del conjunto de datos normalizado empleando one-hot encoding. Los resultados se muestran en la figura 5.13.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE	ROC AUC	Tiempo
Statsbomb	0.896870	0.701863	0.163531	0.265258	0.826100	0.001225
MLP	0.881878	0.466495	0.261939	0.335496	0.772147	27.031874
Decision Tree	0.828336	0.274131	0.308249	0.290191	0.601698	0.419594
Random Forest	0.896540	0.695652	0.162084	0.262911	0.801004	4.879044
XGBoost	0.886491	0.503546	0.205499	0.291881	0.792764	0.678845
KNN	0.877100	0.403509	0.166425	0.235656	0.676830	1.914392
Logistic Regression	0.894563	0.660377	0.151954	0.247059	0.810529	0.279972

Figura 5.13: Métricas del conjunto normalizado con one-hot encoding

Se puede apreciar que con la incorporación de las variables posicionales ha habido un aumento general del rendimiento, y varios modelos se han acercado al rendimiento del modelo de referencia. En concreto, las mejores técnicas que se han destacado anteriormente han mejorado su rendimiento, pero cabe destacar el aumento de rendimiento de la regresión logística, que ha pasado de un rendimiento de en torno a 0,75 a 0,815 en la métrica $ROC - AUC$. De hecho, ha sido el modelo que mejor rendimiento ha obtenido.

En cuanto al impacto de los conjuntos de datos, se puede apreciar que depende de la técnica. Por ejemplo, el perceptrón multicapa ha visto reducido su rendimiento con los conjuntos que emplean $one-hotencoding$, mientras que a la regresión logística le ha sucedido lo contrario. Por tanto, en este caso lo más significativo ha sido aplicar o no aplicar esta transformación, ya que entre estandarizar y normalizar apenas se han podido ver diferencias.

La configuración que mayor rendimiento ha obtenido en función del $ROC-AUC$ ha sido el conjunto $one-hot\ encoding$ estandarizado. Por ello, se ha elegido dicho conjunto para ajustar los hiperparámetros. Los resultados de los modelos ajustados se pueden apreciar en la figura 5.14.

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE:	ROC AUC	Tiempo
MLP	0.882208	0.460784	0.204052	0.282849	0.754746	28.498548
Decision Tree	0.854860	0.333333	0.274964	0.301348	0.660540	1.264873
Random Forest	0.897529	0.711656	0.167873	0.271663	0.800892	10.646369
XGBoost	0.895881	0.657754	0.178003	0.280182	0.818482	22.790965
KNN	0.879077	0.428094	0.185239	0.258586	0.702987	1.372871
Logistic Regression	0.896705	0.716216	0.153401	0.252682	0.815570	0.354784

Figura 5.14: Métricas de los modelos ajustados con variables posicionales

Se puede apreciar que tras el ajuste de hiperparámetros ha habido un ligero aumento en el rendimiento de los modelos. Los modelos que peor rendimiento presentaban antes del ajuste han sido los más mejorados, mientras que aquellos que tenían un rendimiento mejor apenas han mejorado. Cabe destacar que otra vez más, el modelo que mejor rendimiento ha obtenido tras el ajuste de hiperparámetros ha sido el $XGBoost$, que ha obtenido un 0,818 en comparación al 0,793 de antes del ajuste. Dado que el modelo de referencia obtiene un 0,826 en esta métrica, se ha conseguido un modelo con un rendimiento muy similar. Dado que la cantidad de datos y recursos de los que dispone *statsbomb* para hacer los modelos, se puede considerar que se ha obtenido un rendimiento más que aceptable.

5.1.3. Reducción de variables

Habiendo conseguido el mejor modelo de clasificación, a continuación se tratará de mejorar su rendimiento con la eliminación de variables o se tratará de mantener el rendimiento similar reduciendo variables.

SelectKBest

El mejor rendimiento se obtiene excluyendo la variable $play\_pattern\_id\_8$(ver Fig. 5.15).

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE:	ROC AUC	Tiempo
XGBoost	0.895387	0.634615	0.191027	0.29366	0.820683	26.884627

Figura 5.15: Métricas del mejor modelo según $SelectKBest$

SelectFromModel(Random Forest)

El mejor rendimiento se obtiene excluyendo las variables *under_pressure*, *shot_first_time*, *aerial_won*, *shot_one_on_one*, *shot_follows_dribble*, *play_pattern_id_1*, *play_pattern_id_2*, *play_pattern_id_3*, *play_pattern_id_4*, *play_pattern_id_5*, *play_pattern_id_6*, *play_pattern_id_7*, *play_pattern_id_8*, *play_pattern_id_9*, *position_id_1.0*, *position_id_2.0*, *position_id_3.0*, *position_id_4.0*, *position_id_5.0*, *position_id_6.0*, *position_id_7.0*, *position_id_8.0*, *position_id_9.0*, *position_id_10.0*, *position_id_11.0*, *position_id_12.0*, *position_id_13.0*, *position_id_14.0*, *position_id_15.0*, *position_id_16.0*, *position_id_17.0*, *position_id_18.0*, *position_id_19.0*, *position_id_20.0*, *position_id_21.0*, *position_id_22.0*, *position_id_23.0*, *position_id_24.0*, *position_id_25.0*, *sub_type_id_61.0*, *sub_type_id_62.0*, *sub_type_id_87.0*, *sub_type_id_88.0*, *body_part_id_37.0*, *body_part_id_38.0*, *body_part_id_40.0*, *body_part_id_70.0*, *technique_id_89.0*, *technique_id_90.0*, *technique_id_91.0*, *technique_id_92.0*, *technique_id_93.0*, *technique_id_94.0*, *technique_id_95.0* (ver Fig.5.16).

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE:	ROC AUC	Tiempo
XGBoost	0.89374	0.64557	0.147612	0.240283	0.809032	14.18185

Figura 5.16: Métricas del mejor modelo según $SelectFromModel(RandomForest)$

SelectFromModel(Lasso)

El mejor rendimiento se obtiene excluyendo la variable *position_id*_19,0(ver Fig. 5.17).

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE:	ROC AUC	Tiempo
XGBoost	0.894563	0.629442	0.17945	0.279279	0.819353	22.522295

Figura 5.17: Métricas del mejor modelo según $SelectFromModel(Lasso)$

RFE

El mejor rendimiento se obtiene excluyendo la variable *position_id*_11,0 (ver Fig. 5.36).

MODELO	ACCURACY	PRECISION	RECALL	F1-SCORE:	ROC AUC	Tiempo
XGBoost	0.896376	0.659794	0.185239	0.289266	0.820898	20.576569

Figura 5.18: Métricas del mejor modelo según RFE

5.1.3.1. Conclusión

Se puede apreciar que las dos técnicas que mejor rendimiento han proporcionado han sido $SelectKBest$ y RFE. En suma, la mejor configuración de variables para el modelo de clasificación es excluyendo las variables *position_id* y se obtiene un

rendimiento de 0,820 en $ROC - AUC$. Si se eliminan más variables, el rendimiento empieza a decaer considerablemente, por lo que en este caso no se puede eliminar más que una variable. De esta forma, el rendimiento del mejor modelo obtenido se queda muy cerca del modelo de referencia.

5.2. Regresión

En primer lugar, se mostrarán los resultados del enfoque de regresión. Para ello, se emplearán las siguientes métricas.

- MSE
- $RMSE$
- MAE
- $MedianAE$
- R^2
- $ExplainedVariance$
- $AUC - ROC$

5.2.1. Sin variables nuevas

A continuación se mostrarán las métricas de los modelos creados a partir de las variables que ya se encontraban en la base de datos de la que se ha obtenido la información de los tiros. Como se ha introducido anteriormente, se mostrarán los resultados para los distintos conjuntos de datos.

Conjunto base

Evaluación del conjunto de datos sin normalizar, estandarizar y sin emplear *one – hot encoding*. (ver Fig. 5.19).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.009676	0.098369	0.061213	0.037602	0.459321	0.459360	0.745765	0.006754
Árbol de Decisión	0.007737	0.087962	0.044815	0.013987	0.567671	0.568273	0.810404	0.153554
Random Forest	0.004133	0.064291	0.033995	0.012359	0.769048	0.769261	0.837289	0.827141
XGBoost	0.003646	0.060385	0.033228	0.014260	0.796256	0.796260	0.849097	0.221028
MLP	0.015991	0.126457	0.104217	0.092700	0.106464	0.363242	0.739613	1.848396
KNN	0.008439	0.091863	0.049947	0.018324	0.528477	0.528479	0.761680	0.089074

Figura 5.19: Métricas del conjunto base con variables base

Conjunto con one-hot encoding

Evaluación del conjunto de datos empleando *one − hot encoding*. (ver Fig. 5.20).

Modelo	MSE	RMSE	MAE	MedianAE	R^2	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.008910	0.094395	0.058975	0.035779	0.502118	0.502258	0.758772	0.042251
Árbol de Decisión	0.007682	0.087648	0.044174	0.013456	0.570754	0.571298	0.801179	0.225058
Random Forest	0.004053	0.063663	0.033229	0.011556	0.773534	0.773582	0.837645	1.277224
XGBoost	0.003608	0.060067	0.033105	0.014628	0.798399	0.798400	0.851666	0.403150
MLP	0.007564	0.086972	0.059418	0.041137	0.577344	0.583516	0.789454	12.055287
KNN	0.007748	0.088023	0.048598	0.019657	0.567071	0.567078	0.769142	0.013796

Figura 5.20: Métricas del conjunto con one-hot encoding

Conjunto estandarizado

Evaluación del conjunto de datos estandarizado. (ver Fig. 5.21).

Modelo	MSE	RMSE	MAE	MedianAE	R^2	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.009676	0.098369	0.061213	0.037602	0.459321	0.459360	0.745765	0.006916
Árbol de Decisión	0.007695	0.087723	0.044651	0.013946	0.570018	0.570622	0.810649	0.156410
Random Forest	0.004111	0.064116	0.033970	0.012436	0.770303	0.770507	0.837359	0.809286
XGBoost	0.003646	0.060385	0.033228	0.014260	0.796256	0.796260	0.849097	0.228347
MLP	0.081980	0.286322	0.273632	0.282659	-3.580731	0.436298	0.742489	1.802132
KNN	0.008853	0.094091	0.054659	0.026372	0.505325	0.505360	0.815359	0.046005

Figura 5.21: Métricas del conjunto estandarizado

Conjunto normalizado

Evaluación del conjunto de datos normalizado. (ver Fig. 5.22).

Modelo	MSE	RMSE	MAE	MedianAE	R^2	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.009676	0.098369	0.061213	0.037602	0.459321	0.459360	0.745765	0.006856
Árbol de Decisión	0.007656	0.087500	0.044582	0.013946	0.572202	0.572741	0.810919	0.154802
Random Forest	0.004114	0.064144	0.033977	0.012419	0.770103	0.770312	0.837884	0.832385
XGBoost	0.003646	0.060385	0.033228	0.014260	0.796256	0.796260	0.849097	0.788943
MLP	0.010756	0.103709	0.063717	0.037085	0.399021	0.416273	0.747554	2.817106
KNN	0.012098	0.109991	0.065518	0.034590	0.324014	0.325333	0.837582	0.052636

Figura 5.22: Métricas del conjunto normalizado

Conjunto estandarizado con one-hot encoding

Evaluación del conjunto de datos estandarizado y empleando one-hot encoding. (ver Fig. 5.23).

Modelo	MSE	RMSE	MAE	MedianAE	R^2	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.008910	0.094395	0.058975	0.035779	0.502118	0.502258	0.758772	0.042577
Árbol de Decisión	0.007609	0.087229	0.043901	0.013416	0.574845	0.575312	0.801329	0.223070
Random Forest	0.004052	0.063656	0.033258	0.011646	0.773583	0.773628	0.837526	1.258828
XGBoost	0.003608	0.060067	0.033105	0.014628	0.798399	0.798400	0.851666	0.376127
MLP	0.008274	0.090964	0.055758	0.030852	0.537657	0.538339	0.766372	1.622785
KNN	0.007494	0.086566	0.049213	0.022032	0.581282	0.581339	0.838387	0.011623

Figura 5.23: Métricas del conjunto estandarizado con one-hot encoding

Conjunto normalizado con one-hot encoding

Se trata del conjunto de datos normalizado empleando one-hot encoding. (ver Fig. 5.24).

Modelo	MSE	RMSE	MAE	MedianAE	R^2	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.008910	0.094395	0.058975	0.035779	0.502118	0.502258	0.758772	0.041174
Árbol de Decisión	0.007635	0.087381	0.044018	0.013422	0.573362	0.573876	0.801344	0.228151
Random Forest	0.004049	0.063629	0.033250	0.011631	0.773774	0.773821	0.838049	1.235720
XGBoost	0.003608	0.060067	0.033105	0.014628	0.798399	0.798400	0.851666	0.398658
MLP	0.008689	0.093216	0.059292	0.038419	0.514483	0.515265	0.776756	6.193163
KNN	0.011244	0.106037	0.063299	0.032952	0.371737	0.373132	0.833619	0.021798

Figura 5.24: Métricas del conjunto normalizado con one-hot encoding

Se puede apreciar que no hay diferencias significativas en función del preprocesamiento del conjunto de datos, ya que todos los resultados se han mantenido en un rango muy reducido. En cambio, sí que se puede apreciar una diferencia sustancial entre los distintos modelos. Atendiendo a todas las métricas, en general se puede apreciar que el mejor modelo es el $XGBoost$ seguido del $RandomForest$. Ambos han obtenido un valor cercano a 0,8 en R^2, mientras que el resto se ha quedado cerca del 0,5.

La configuración que mayor rendimiento ha obtenido en función del R^2 han sido los conjuntos $one-hot$, $one-hot\ estandarizado$ y $one-hot\ normalizado$. Por ello, se ha elegido el conjunto $one-hot$ para ajustar los hiperparámetros. (ver Fig. 5.25).

Modelo	MSE	RMSE	MAE	MedianAE	R^2	Explained Variance	ROC AUC	Tiempo
MLP	0.004633	0.068064	0.037730	0.016742	0.741141	0.741924	0.807169	299.623119
KNN	0.006054	0.077810	0.042531	0.017329	0.661703	0.661712	0.797773	115.739189
RandomForest	0.004556	0.067500	0.038332	0.018575	0.745416	0.745434	0.858904	243.624646
XGBoost	0.003390	0.058222	0.031333	0.013148	0.810592	0.810601	0.859642	5.970187
DecisionTree	0.007197	0.084833	0.046151	0.018879	0.597883	0.604142	0.794374	339.294213
LinearRegression	0.008910	0.094395	0.058975	0.035779	0.502118	0.502258	0.758772	0.286705

Figura 5.25: Métricas de los modelos ajustados con variables base

Se puede apreciar que tras el ajuste de hiperparámetros ha habido un ligero aumento en el rendimiento de los modelos. En concreto, el mejor modelo ha obtenido un valor de 0,810 en la métrica R^2, y se trata, nuevamente, de $XGBoost$. Otra vez más, se puede intuir que esta técnica va a ser la que mejor rendimiento vaya a obtener, como ha sucedido en el enfoque de clasificación.

5.2.2. Con variables nuevas

A continuación se mostrarán las métricas de los modelos creados a partir de las variables posicionales que se han añadido. Como se ha hecho anteriormente, se mostrarán los resultados para los distintos conjuntos de datos.

Conjunto base

Evaluación del conjunto de datos sin normalizar, estandarizar y sin emplear $one-hot\ encoding$. (ver Fig. 5.26).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	5.191304e-03	0.072051	0.043054	0.024351	7.099299e-01	7.099362e-01	0.801090	0.011220
Árbol de Decisión	3.966014e-03	0.062976	0.031229	0.009767	7.783944e-01	7.784054e-01	0.805831	0.539223
Random Forest	2.184252e-03	0.046736	0.023620	0.008025	8.779524e-01	8.779524e-01	0.819846	2.998027
XGBoost	1.941209e-03	0.044059	0.022224	0.008147	8.915327e-01	8.915922e-01	0.831997	0.474875
MLP	2.822908e+11	531310.488941	8791.896048	0.477436	-1.577333e+13	-1.576901e+13	0.576983	0.579879
KNN	5.967210e-03	0.077248	0.039237	0.014163	6.665752e-01	6.670221e-01	0.788762	0.003726

Figura 5.26: Métricas del conjunto base con variables posicionales

Conjunto con one-hot encoding

Evaluación del conjunto de datos empleando $one - hot\ encoding$. (ver Fig. 5.27).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	4.262032e-03	0.065284	0.040004	0.024192	7.618540e-01	7.619306e-01	0.806654	0.054516
Árbol de Decisión	4.219733e-03	0.064959	0.031694	0.009718	7.642175e-01	7.642997e-01	0.800495	0.460801
Random Forest	2.171951e-03	0.046604	0.023519	0.007771	8.786397e-01	8.786414e-01	0.820743	2.864904
XGBoost	1.989253e-03	0.044601	0.022313	0.008383	8.888482e-01	8.888742e-01	0.832969	0.659886
MLP	4.337522e+10	208267.188022	3448.938311	2.934253	-2.423641e+12	-2.422976e+12	0.359711	1.683202
KNN	5.418227e-03	0.073609	0.038124	0.015994	6.972503e-01	6.973614e-01	0.787988	0.013684

Figura 5.27: Métricas del conjunto con one-hot encoding

Conjunto estandarizado

Evaluación del conjunto de datos estandarizado.(ver Fig. 5.28).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.005191	0.072051	0.043054	0.024351	0.709930	0.709936	0.801090	0.013915
Árbol de Decisión	0.003887	0.062350	0.030805	0.009699	0.782782	0.782826	0.803386	0.504027
Random Forest	0.002142	0.046284	0.023390	0.007793	0.880300	0.880300	0.820368	3.061660
XGBoost	0.001974	0.044430	0.022438	0.008330	0.889700	0.889745	0.831047	0.539495
MLP	0.006127	0.078277	0.047098	0.029378	0.657628	0.665251	0.805704	1.577118
KNN	0.004927	0.070193	0.037775	0.014691	0.724699	0.724699	0.799832	0.003792

Figura 5.28: Métricas del conjunto estandarizado

Conjunto normalizado

Evaluación del conjunto de datos normalizado. (ver Fig.5.29).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.005191	0.072051	0.043054	0.024351	0.709930	0.709936	0.801090	0.010858
Árbol de Decisión	0.003981	0.063092	0.031261	0.009770	0.777577	0.777586	0.805782	0.379368
Random Forest	0.002187	0.046769	0.023627	0.008032	0.877779	0.877779	0.819904	2.384660
XGBoost	0.001941	0.044059	0.022224	0.008147	0.891533	0.891592	0.831997	0.485089
MLP	0.016671	0.129116	0.109555	0.104315	0.068495	0.693714	0.803889	1.977441
KNN	0.006389	0.079928	0.043811	0.017247	0.643034	0.643095	0.796923	0.004072

Figura 5.29: Métricas del conjunto normalizado

Conjunto estandarizado con one-hot encoding

Evaluación del conjunto de datos estandarizado y empleando one-hot encoding.(ver Fig.5.30).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.004262	0.065284	0.040004	0.024192	0.761854	0.761931	0.806654	0.056387
Árbol de Decisión	0.004067	0.063769	0.031232	0.009691	0.772777	0.773142	0.806486	0.435315
Random Forest	0.002164	0.046521	0.023436	0.007772	0.879074	0.879079	0.819720	3.042981
XGBoost	0.001988	0.044586	0.022335	0.008173	0.888922	0.888993	0.828563	1.999793
MLP	0.006274	0.079211	0.042200	0.026049	0.649416	0.653162	0.793476	4.130556
KNN	0.003941	0.062775	0.033522	0.012964	0.779812	0.779936	0.814863	0.013612

Figura 5.30: Métricas del conjunto estandarizado con one-hot encoding

Conjunto normalizado con one-hot encoding

Evaluación del conjunto de datos normalizado empleando one-hot encoding.(ver Fig. 5.31).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	0.004262	0.065284	0.040004	0.024192	0.761854	0.761931	0.806654	0.232329
Árbol de Decisión	0.004077	0.063848	0.031261	0.009779	0.772217	0.772315	0.799496	0.661794
Random Forest	0.002170	0.046587	0.023471	0.007711	0.878727	0.878728	0.820868	2.848520
XGBoost	0.001990	0.044610	0.022316	0.008383	0.888804	0.888831	0.832959	0.666784
MLP	0.004584	0.067705	0.045275	0.030614	0.743862	0.762124	0.801676	2.293480
KNN	0.005604	0.074860	0.041003	0.016511	0.686870	0.686872	0.808632	0.012357

Figura 5.31: Métricas del conjunto normalizado con one-hot encoding

Adicionalmente a las conclusiones que se han sacado en la sección anterior respecto de las diferencias de los conjuntos de datos y de los modelos, en cuanto a la diferencia entre incluir o no las variables posicionales, a diferencia de lo sucedido en el enfoque de clasificación, sí que se ha podido apreciar una mejora notoria. Lo cierto es que en la mayoría de los casos se puede apreciar una mejora no menor al 10 % en la métrica R^2 e incluso más del 20 % en algún modelo.

La configuración que mayor rendimiento ha obtenido en función del R^2 ha sido el conjunto base y el estandarizado. Por ello, se ha elegido el conjunto base para ajustar los hiperparámetros. En concreto, el mejor modelo se ha creado con *XGBoost* y ha obtenido un valor de 0,891 en la métrica R^2. Los resultados de los modelos ajustados se pueden apreciar en la figura 5.32.

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
Regresión Lineal	5.191304e-03	0.072051	0.043054	0.024351	7.099299e-01	7.099362e-01	0.801090	0.011220
Árbol de Decisión	3.966014e-03	0.062976	0.031229	0.009767	7.783944e-01	7.784054e-01	0.805831	0.539223
Random Forest	2.184252e-03	0.046736	0.023620	0.008025	8.779524e-01	8.779524e-01	0.819846	2.998027
XGBoost	1.941209e-03	0.044059	0.022224	0.008147	8.915327e-01	8.915922e-01	0.831997	0.474875
MLP	2.822908e+11	531310.488941	8791.896048	0.477436	-1.577333e+13	-1.576901e+13	0.576983	0.579879
KNN	5.967210e-03	0.077248	0.039237	0.014163	6.665752e-01	6.670221e-01	0.788762	0.003726

Figura 5.32: Métricas de los modelos ajustados con variables posicionales

Se puede apreciar que tras el ajuste de hiperparámetros ha habido un ligero aumento en el rendimiento de los modelos. Los modelos que peor rendimiento presentaban antes del ajuste han sido los más mejorados, mientras que aquellos que tenían un rendimiento mejor apenas han mejorado.

5.2.3. Reducción de variables

Habiendo conseguido el mejor modelo de regresión, a continuación se tratará de mejorar su rendimiento con la eliminación de variables o se tratará de mantener el

rendimiento similar reduciendo variables.

SelectKBest

El mejor rendimiento se obtiene excluyendo la variable *shot_follows_dribble* (5.33).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
XGBoost	0.001629	0.040363	0.02037	0.007272	0.908966	0.909016	0.832009	4.968785

Figura 5.33: Métricas del mejor modelo según *SelectKBest*

SelectFromModel(Random Forest)

El mejor rendimiento se obtiene excluyendo las variables *play_pattern_id*, *position_id*, *body_part_id*, *sub_type_id*, *under_pressure*, *technique_id*, *shot_first_time*, *aerial_won*, *shot_one_on_one*, *shot_open_goal*, *shot_follows_dribble* (ver Fig. 5.34).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
XGBoost	0.005091	0.071349	0.038418	0.01502	0.715552	0.715571	0.860646	1.34642

Figura 5.34: Métricas del mejor modelo según *SelectFromModel*(*RandomForest*)

SelectFromModel(Lasso)

El mejor rendimiento se obtiene excluyendo las variables $cuantos_cerca$ y *angulo_desviado_portero* (ver Fig. 5.35).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
XGBoost	0.002067	0.045468	0.022518	0.007448	0.884486	0.884563	0.835915	7.117692

Figura 5.35: Métricas del mejor modelo según *SelectFromModel*(*Lasso*)

*RFE

El mejor rendimiento se obtiene excluyendo la variable *angulo_desviado_portero* (ver Fig. 5.36).

Modelo	MSE	RMSE	MAE	MedianAE	R²	Explained Variance	ROC AUC	Tiempo
XGBoost	0.001708	0.04133	0.02077	0.007529	0.904555	0.904587	0.830611	4.614712

Figura 5.36: Métricas del mejor modelo según *RFE*

5.2.3.1. Conclusión

Se puede apreciar que, al igual que ha sucedido en el enfoque de clasificación, las técnicas de reducción de variables que mejor rendimiento han proporcionado han sido *SelectKBest* y *RFE*.

En suma, la mejor configuración de variables para el modelo de regresión es excluyendo las variables *shot_follows_dribble* y se obtiene un rendimiento de 0,908966 en R^2. Si se eliminan más variables, el rendimiento empieza a decaer considerablemente, por lo que en este caso no compensa eliminar más que una variable.

5.3. Conclusión

En suma, la incorporación de las variables posicionales que se han creado desde cero ha tenido un impacto significativo en el rendimiento de los modelos. Por otro lado, la diferencia de rendimiento entre los distintos conjuntos de datos no ha sido tan significativa, y ha dependido de cada modelo. No obstante, entre las distintas técnicas sí ha habido diferencias significativas, aunque la mejor técnica ha resultado ser *XGBoost*. La reducción de variables no ha resultado muy fructífera, puesto que el aumento de rendimiento ha sido casi inexistente, y cuando se han eliminado más variables, el rendimiento ha decaído considerablemente.

Capítulo 6

Conclusiones y Trabajo Futuro

En este trabajo se ha abordado el problema de crear un modelo que calcule la probabilidad de que un tiro termine en gol (gol esperado). Para ello se ha planteado un enfoque experimental basado en la comparativa de distintas alternativas. Se han probado distintas configuraciones de datasets (añadiendo nuevas variables y haciendo distintos preprocesamientos), diferentes modelos e hiperparámetros, y se ha realizado un análisis de reducción de variables. Todo ello se ha llevado a cabo desde dos enfoques: un planteamiento con un modelo de referencia de *statsbomb* que busca crear un modelo lo más parecido posible al mismo (enfoque de regresión). El otro se ha basado en crear un modelo desde cero sin ningún modelo de referencia (enfoque de clasificación).

6.1. Conclusiones

Tras la realización del trabajo se pueden extraer diversas conclusiones. Por un lado, en cuanto al preprocesamiento de los datos, se puede apreciar que en este caso particular apenas hay diferencias de rendimiento entre las distintas modificaciones que se le pueden hacer al conjunto de datos. Simplemente han servido para tratar de refinar el rendimiento. El impacto que han tenido las distintas formas de preprocesamiento ha variado en función del modelo empleado, pero sin ser determinantes en el rendimiento.

Por otro lado, las variables posicionales que se han añadido han tenido un impacto variado. Estas variables capturan patrones más complejos del posicionamiento de los jugadores en el campo en el momento del disparo. En el enfoque de clasificación apenas han mejorado el rendimiento. No obstante, en el enfoque de regresión sí que ha habido una mejora significativa del rendimiento de los modelos.

Cabe destacar que entre los modelos sí que ha habido diferencias más sustanciales. En general, los modelos basados en árboles han tenido un desempeño general superior al resto de modelos. A este conjunto de modelos se le suman las redes neuronales, que también han obtenido un rendimiento equiparable.

Para terminar, la reducción de variables no ha tenido un impacto determinante en los modelos, puesto que, exceptuando unas pocas variables, al eliminar el resto,

el rendimiento se ha visto degradado.

6.2. Trabajo futuro

Como líneas futuras de investigación, se podría ampliar el trabajo de las siguientes formas. Por un lado, se podría llevar a cabo un enfoque más personalizado por el lanzador, tal y como se ha presentado en los trabajos que han servido de referencia (Madrero Pardo, 2020), (Hewitt y Karakuş, 2023). Asimismo, se podrían haber tenido en cuenta factores psicológicos, incluidos también en estos trabajos (Mead et al., 2023). En suma, se podrían ampliar los enfoques con los que se aborda el problema y tratar de hacerlo de las máximas formas posibles, incluyendo lo mejor de cada uno.

También se podría extender este análisis a otras de las principales ligas de fútbol y analizar si existen diferencias entre ellas. Asimismo, se podría extender el análisis al fútbol femenino y examinar si existen o no patrones diferentes entre el fútbol masculino y el femenino.

Además de esto, se podría tratar de obtener datos más completos. Esto podría llevar a que modelos que antes no obtenían un buen rendimiento pasaran a tenerlo, como es el caso de las redes neuronales. De esta forma, se podría probar el rendimiento de diversas redes neuronales para ver si se mejora el rendimiento. Por otro lado, el incremento de los datos daría pie a probar el rendimiento real del modelo en casos prácticos. Por ejemplo, si se dispusiera de todos los tiros de una temporada completa, se podría tratar de predecir quién debería haber ganado dicha liga, o qué jugador debería haber sido el *pichichi*, es decir, el máximo goleador, entre otros.

Otro punto a considerar sería la combinación de la variable de goles esperados con otras relacionadas, como las asistencias esperadas, lo cual podría llevar a hacer un análisis más profundo de las jugadas. Esto podría ser más útil para los equipos de fútbol, ya que obtendrían un enfoque más completo del ataque.

Para finalizar, se podría mejorar la interpretabilidad y explicabilidad de los modelos creados mediante diferentes técnicas de inteligencia artificial explicativa. Esto ayudaría a transferir los resultados obtenidos al mundo real y ayudaría a sacar conclusiones más certeras.

Introduction

Soccer is a sport that was invented in England during the 19th century (Goldblatt, 2008). It quickly spread throughout Europe and later across the world, becoming one of the most played and followed sports globally. In its origins, the sport was very offensive, and players focused almost exclusively on attacking. As time went by, the sophistication of the game also evolved. Coaches began creating more balanced formations and started to detail more specific positions (Wilson, 2013).

It was in the mid-20th century when plays began to be recorded with the aim of finding game patterns that optimized the chances of winning matches (Anderson y Sally, 2013). Initially, an erroneous but pioneering conclusion in sports analysis was drawn—that the game should be more direct.

By the end of the last century, match recordings were used to extract the best possible conclusions about which types of plays were most effective. Likewise, the first annotations of the most significant match events (passes, shots, fouls, etc.) began to be created.

With the arrival of the new millennium, and thanks to Opta Sports (Barnett, 2014), detailed statistics of each match started to be recorded, marking a before and after in the way soccer was understood. Until then, all tactical revolutions in soccer resulted from the insightful intuitions and deep understanding of the game that great coaches had in past decades. However, from this point on, all teams began incorporating all these data to shape their style of play.

Finally, in the 2010s, the emergence of more advanced statistical models such as expected goals (xG) and expected assists (xA) opened the door to the current way of analyzing soccer (Fernández et al., 2019).

Expected Goals (xG)

With the arrival of large volumes of data and the development of data analysis techniques, one of the major revolutions was the appearance of the concept of expected goals (xG) (Magazine, 2022). Although this concept cannot be attributed to any single author, the first expected goals models were developed in the early 21st century by both academic researchers (Ensum et al., 2005) and companies specialized in sports data analysis (Knutson, 2018).

Until then, the way to analyze soccer was certainly result-oriented, or although

not always, there were no accessible tools to avoid this type of analysis.

Taking reductionism to the extreme, one can analyze who played better merely based on the result. However, it is clear that this approach carries some problems. It might happen that a team loses a match despite having played better and deserved to win more than the opponent, or vice versa. Or a team might have played much better than the rival but won only by one goal.

Being slightly less reductionist, one could analyze who deserved to win based on the number of shots taken. This is somewhat more realistic, since a team that has taken many more shots can be assumed to have had more chances to win. Refining this analysis further, one could also look at shots on target, as it is obvious that some shots barely cause danger, and the previous approach would count those.

Nevertheless, there is still a great imprecision when measuring who should have won considering only shots on target. This is precisely where the concept of expected goals plays a role. Although this concept is still a simplification, it is a much clearer approach to what happened during a match.

Expected goals assign a probability of scoring to each shot, thus solving the problem mentioned above about the difference in shot danger. That is, a shot from midfield should not be counted the same as a penalty or a shot at an empty net, since the probability that each shot ends up in the goal—and therefore the danger each shot represents—is very different. Using just shots or shots on target ignores this difference in danger, leading to a poorer and less representative analysis.

Thus, summing all the goal probabilities of the shots taken by a team during a match yields what is known as expected goals. It is evident that this approach is more accurate than the previous ones, which is why it is currently the most representative statistic in soccer.

It should be reiterated that this approach still leads to some inaccuracies in certain cases. For example, a team might have reached very close positions to the opponent's goal, but due to imprecisions in the last pass or interceptions before the shot, the expected goals statistic might not fully reflect reality. Nonetheless, it is a much more faithful perspective of what happened in a match. That is why, alongside expected goals, other concepts such as expected assists have emerged to better reflect the reality of the game.

In sum, the importance of this concept lies in enabling more realistic match analysis. This allows coaches to make more detailed tactical adjustments, as they have a more precise understanding of reality. Otherwise, if the more reductionist approach were taken, a coach might think no adjustments are needed just because the team won, potentially leading them to believe the victory was due to their tactical proposal when the reality is closer to having won despite it.

Today, club technical teams, sports journalists, and analysts use this metric to gain deeper insights into matches. However, the main beneficiaries are fans, due to the increased level of play in matches thanks to the incorporation of these analytical tools.

Motivation

Due to the growing importance of data analysis in many fields, such as sports and especially soccer, it is interesting to delve into this aspect, which has been one of the most transformative in the sport in recent years. In soccer, the concept of expected goals is one of the most important in this type of analysis.

As a sports enthusiast, especially in the tactical aspect, I believe developing this work can contribute to me both professionally and personally.

Professionally, this work will provide me knowledge about data analysis and serve as an introduction to this field. I will familiarize myself with some of the tools used and enhance some of my technical skills. Additionally, this work could have a direct impact on the field if new knowledge about the topic is obtained.

On a personal level, I might start to watch matches with new eyes. For example, I might begin to enjoy plays that, although not resulting in goals, generated great danger. This would allow me to become a more curious and detail-oriented spectator. Moreover, I could spread a less result-focused view of soccer among those around me, who could benefit from it.

Objectives

The main objective of this work is to create a model that assigns a probability of scoring to each shot. For this, the machine learning task of expected goals for a match will be subdivided into smaller objectives serving as subgoals to complete the work:

- Collect shot data.
- Preprocess the dataset.
- Train different models from a classification approach.
- Train different models from a regression approach.
- Tune the best models.
- Study the effect of positional variables, different data preprocessing methods, and various models.

The work plan (see Fig. 6.1) was as follows: first, the dataset serving as the basis for the model was selected. Considerations included the variables present and the number of rows. If the dataset had very few variables or the variables were insignificant, it would be difficult to create an accurate model. Similarly, if the number of shots in the dataset was very low, drawing general conclusions would be complicated.

Next, the dataset was preprocessed for the most efficient use possible. That is, the dataset is likely not ready to use initially, so several modifications were made, including removing rows with unknown values, discarding insignificant variables,

creating new representative variables, and standardizing and/or normalizing the data, among others.

After this, the dataset was ready to train a model. Two approaches were followed. On one hand, a model to predict goal probability was created without any prior reference—that is, no preexisting goal probability was assumed, and the model tried to assign a probability to each shot from scratch.

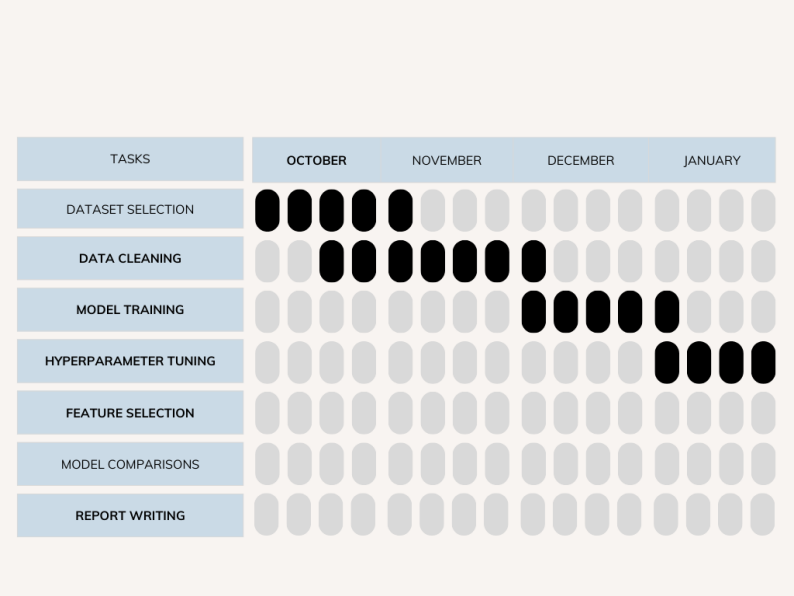
On the other hand, the probability of scoring was estimated based on existing goal probability values from a reference model. In both cases, metrics were used to assess model performance.

Therefore, the first approach corresponds to building a model from scratch, while the second attempts to replicate the operation of an existing model.

Then, the best models were tuned to maximize performance, primarily by checking if any insignificant variables remained.

Conclusions were drawn from the results and presented to capture the insights gained through the described process.

Finally, this report was written after completing the previous steps.



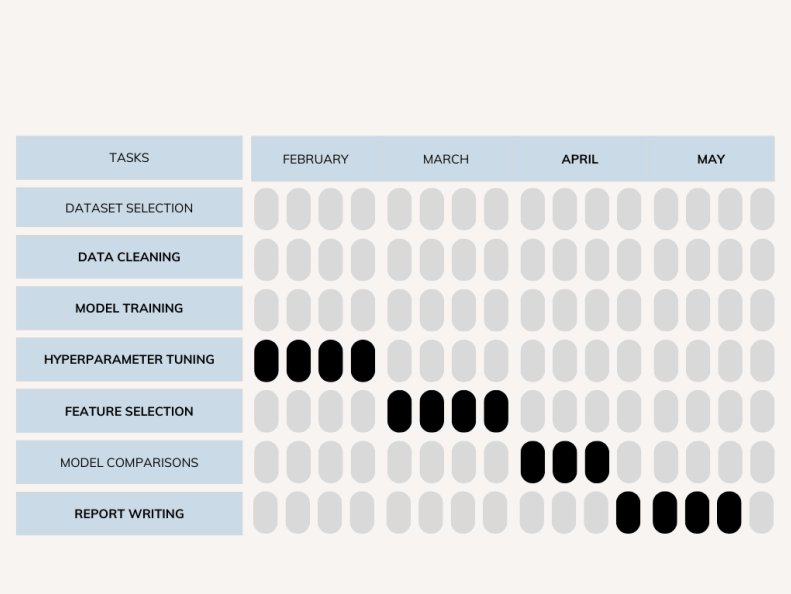


Figure 6.1: Timeline diagram of the work organization

Document Structure

The following chapters will address the following topics. First, the next chapter will cover the basic technical concepts necessary to understand the work development, the techniques and tools used, and some previous works of a similar nature, as well as the main differences of this work compared to them (see Chapter 2).

Next, the infrastructure chapter will detail the specific technical tools used to develop the work (see Chapter 3).

Afterwards, the entire process of creating the expected goals models will be shown—from dataset creation and processing to the different approaches used to build the models (see Chapter 4).

Before concluding, the results obtained from the generated models will be presented, along with a comparison between them (see Chapter 5).

Finally, the last chapter will summarize the main conclusions drawn from carrying out this work, as well as possible changes or improvements that could be made in future work extensions (see Chapter 6.2).

Conclusions and Future Work

This work has addressed the problem of creating a model that calculates the probability that a shot will result in a goal (expected goal). For this, an experimental approach based on comparing different alternatives was proposed. Various dataset configurations were tested (adding new variables and performing different preprocessing steps), different models and hyperparameters were evaluated, and a variable reduction analysis was carried out. All of this was done from two perspectives: one approach with a reference model from *statsbomb* that aims to create a model as close as possible to it (regression approach), and the other based on creating a model from scratch without any reference model (classification approach).

Conclusions

From the work carried out, several conclusions can be drawn. On the one hand, regarding data preprocessing, it can be observed that in this particular case there are hardly any performance differences between the various modifications applied to the dataset. They have simply served to try to refine performance. The impact of the different preprocessing methods varied depending on the model used but was not decisive in the overall performance.

On the other hand, the positional variables that were added had a varied impact. These variables capture more complex patterns of player positioning on the field at the moment of the shot. In the classification approach, they barely improved performance. However, in the regression approach, there was a significant improvement in model performance.

It is worth noting that there were more substantial differences among the models. In general, tree-based models performed better overall than the other models. Neural networks are added to this group, also achieving comparable performance.

Finally, variable reduction did not have a decisive impact on the models since, except for a few variables, removing the rest degraded the performance.

Future Work

As future lines of research, the work could be expanded in the following ways. On the one hand, a more personalized approach by the shooter could be carried out, as presented in the referenced works (Madrero Pardo, 2020), (Hewitt y Karakuş, 2023). Additionally, psychological factors could be taken into account, also included in these studies (Mead et al., 2023). In summary, the approaches to the problem could be broadened and combined to include the best aspects of each.

This analysis could also be extended to other major football leagues to investigate whether differences exist among them. Furthermore, the analysis could be extended to women's football to examine whether different patterns exist between men's and women's football.

Besides this, more comprehensive data could be sought. This could enable models that previously did not perform well to improve, such as neural networks. In this way, the performance of various neural networks could be tested to see if there is an improvement. On the other hand, the increase in data would allow testing the actual performance of the model in practical cases. For example, if all shots from a complete season were available, it could be attempted to predict who should have won that league or which player should have been the *pichichi* (top scorer), among others.

Another point to consider would be combining the expected goals variable with related ones, such as expected assists, which could lead to a deeper analysis of plays. This might be more useful for football teams, as it would provide a more complete view of the attack.

Finally, the interpretability and explainability of the created models could be improved using different explainable artificial intelligence techniques. This would help transfer the results obtained to the real world and assist in drawing more accurate conclusions.

Bibliografía

Alpaydin, E. *Machine learning*. MIT press, 2021.

Anderson, C. y Sally, D. *The Numbers Game: Why Everything You Know About Soccer is Wrong*. Penguin Books, 2013.

Arnold, C., Biedebach, L., Küpfer, A. y Neunhoeffer, M. The role of hyperparameters in machine learning models and how to tune them. *Political Science Research and Methods*, vol. 12(4), páginas 841–848, 2024.

Barnett, M. Data professional: Simon banoub. *Marketing Week*, 2014. Https://www.marketingweek.com/data-professional-simon-banoub/. Accedido el 3 de mayo de 2025.

Burzykowski, T., Geubbelmans, M., Rousseau, A.-J. y Valkenborg, D. Validation of machine learning algorithms. *American Journal of Orthodontics and Dentofacial Orthopedics*, vol. 164(2), páginas 295–297, 2023.

Chen, T. y Guestrin, C. Xgboost: A scalable tree boosting system. 2016. Https://xgboost.readthedocs.io/.

De Ville, B. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5(6), páginas 448–455, 2013.

Ensum, J., Pollard, R. y Taylor, S. Applications of logistic regression to shots at goal in association football. En *World congress on science and football, SCIENCE AND FOOTBALL -CONGRESS-*, 5, páginas 211–218. Routledge,;, London:, 2005. ISBN 0415333377.

Fernández, J., Bornn, L. y Cervone, D. Decomposing the immeasurable sport: A deep learning expected possession value framework for soccer. En *13th edition of MIT Sloan Sports Analytics Conference*. MIT Sloan, 2019.

Goldblatt, D. *The Ball is Round: A Global History of Soccer*. Penguin Books, 2008.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J.

ET AL. Array programming with numpy. *Nature*, vol. 585(7825), páginas 357–362, 2020.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., HASTIE, T., TIBSHIRANI, R. y FRIEDMAN, J. Overview of supervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, páginas 9–41, 2009a. Springer.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., HASTIE, T., TIBSHIRANI, R. y FRIEDMAN, J. Unsupervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, páginas 485–585, 2009b. Springer.

HEWITT, J. H. y KARAKUŞ, O. A machine learning approach for player and position adjusted expected goals in football (soccer). *Franklin Open*, vol. 4, página 100034, 2023.

KNUTSON, T. La evolución de los goles esperados en statsbomb. https://statsbomb.com/es/articulos/futbol/mejorando-los-goles-esperados-xg/, 2018. Accedido el 21 de mayo de 2025.

MADRERO PARDO, P. *Creating a model for expected goals in football using qualitative player information*. Proyecto Fin de Carrera, Universitat Politècnica de Catalunya, 2020.

MAGAZINE, B. D. El dato como complemento en el fútbol. 2022. Https://bigdatamagazine.es/el-dato-como-complemento-en-el-futbol/.

MCKINNEY, W. Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, páginas 51–56, 2010.

MEAD, J., O'HARE, A. y MCMENEMY, P. Expected goals in football: Improving model performance and demonstrating value. *Plos one*, vol. 18(4), página e0282295, 2023.

NAIDU, G., ZUVA, T. y SIBANDA, E. M. A review of evaluation metrics in machine learning algorithms. En *Computer science on-line conference*, páginas 15–25. Springer, 2023.

NICK, T. G. y CAMPBELL, K. M. Logistic regression. *Topics in biostatistics*, páginas 273–301, 2007.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M. y ÉDOUARD DUCHESNAY. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, vol. 12, páginas 2825–2830, 2011.

RAXIMOV, N., KUVANDIKOV, J. y DILMUROD, K. The importance of loss function in artificial intelligence. En *2022 International Conference on Information Science and Communications Technologies (ICISCT)*, páginas 1–3. IEEE, 2022.

RESEARCH, G. Google colaboratory. 2025. Https://colab.research.google.com/.

Rigatti, S. J. Random forest. *Journal of Insurance Medicine*, vol. 47(1), páginas 31–39, 2017.

Rowlinson, A. y Durgapal, A. mplsoccer: A python library for plotting soccer/football charts. 2021. Https://mplsoccer.readthedocs.io/. Accedido el 20 de mayo de 2025.

Singh, J. y Banerjee, R. A study on single and multi-layer perceptron neural network. En *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, páginas 35–40. IEEE, 2019.

StatsBomb. statsbombpy: A python client for the statsbomb api. 2020. Https://github.com/statsbomb/statsbombpy. Accedido el 20 de mayo de 2025.

StatsBomb. Conjunto de datos. 2025.

Tian, Y. y Zhang, Y. A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, vol. 80, páginas 146–166, 2022.

Varano, G. understatpy: Python wrapper for understat data. 2020. Https://understat.readthedocs.io/. Accedido el 20 de mayo de 2025.

Wiering, M. A. y Van Otterlo, M. Reinforcement learning. *Adaptation, Learning, and Optimization*, vol. 12(3), página 729, 2012.

Wilson, J. *Inverting the Pyramid: The History of Soccer Tactics*. Nation Books, 2013.