



TRABAJO FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA

# Modelos predictivos aplicados al análisis del salario de futbolistas profesionales

---

**Autor**

Javier Moreno Morón

**Directores**

Óscar Cordon García

Pablo Mesejo Santiago



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

Granada, julio de 2024



# Modelos predictivos aplicados al análisis del salario de futbolistas profesionales

Javier Moreno Morón

**Palabras clave:** Fútbol, estimación del salario de futbolistas profesionales, estimación de influencia de estadísticas, Aprendizaje Automático, modelos predictivos, regresión, selección de características.

## Resumen

En las últimas décadas el fútbol se ha convertido en uno de los deportes más populares a nivel mundial y uno de los que más dinero mueve. Un porcentaje importante del mismo es gracias a los salarios que cobran los futbolistas. El problema, es que actualmente los métodos para estimar sueldos son muy subjetivos. Y en muchas ocasiones se ve influenciado por la capacidad negociadora tanto del club como del agente del futbolista. Se plantea entonces la necesidad de dotar de automatización a la estimación de los salarios de los futbolistas.

Este TFG aborda tanto el desarrollo de un sistema capaz de predecir el salario estimado que debería cobrar un futbolista, como la determinación de las características que más influyen en el mismo. Nos planteamos alcanzar estos objetivos enfocando el problema dentro del ámbito del Aprendizaje Automático, mediante el entrenamiento de varios modelos y la selección del que mejor se ajuste a los datos.

La principal peculiaridad de esta investigación frente al estado del arte actual es que en lugar de escoger un conjunto de estadísticas generales reducido, como hacen otros trabajos, se realiza una selección inicial indiscriminada de un amplio conjunto de características. Posteriormente, se reduce el número de las mismas mediante técnicas de selección de características, consiguiendo así que se tengan en cuenta estadísticas que a priori podemos pensar que no son influyentes. Gracias a esta selección indiscriminada, podemos obtener predicciones más precisas y conclusiones más significativas.

Los resultados finales obtenidos en el experimento son bastante competitivos en comparación con el estado del arte actual, consiguiendo un coeficiente de determinación de 0.7. Aunque estos resultados no dejan de ser similares a los obtenidos en otros trabajos, la explicabilidad de la que dotamos al modelo nos permite extraer conclusiones muy interesantes sobre qué características influyen más a la hora de estimar el salario de un futbolista.



# Predictive models applied to the analysis of professional footballers wages

Javier Moreno Morón

**Keywords:** Football, professional footballers wages estimation, stats influence estimation, Machine Learning, predictive models, regression, feature selection.

## Abstract

In recent decades, football has become one of the most popular sports worldwide and one of the most lucrative. A significant percentage of this revenue is due to the salaries that football players earn. The problem is that current methods for estimating salaries are very subjective. Often, these estimates are influenced by the negotiating abilities of both the club and the player's agent. Therefore, there is a need to automate the estimation of football players' salaries.

This final degree project addresses both the development of a system capable of predicting the estimated salary a football player should earn and the determination of the characteristics that most influence it. We aim to achieve these objectives by framing the problem within the scope of Machine Learning, through the training of various models and the selection of the one that best fits the data.

The main peculiarity of this research compared to the current state of the art is that, instead of selecting a limited set of general statistics, as other works do, an initial indiscriminate selection of a wide range of characteristics is performed. Subsequently, the number of these characteristics is reduced using feature selection techniques, thus considering statistics that we might initially think are not influential. Thanks to this indiscriminate selection, we can obtain more accurate predictions and more significant conclusions.

The final results obtained in the experiment are quite competitive compared to the current state of the art, achieving a determination coefficient of 0.7. Although these results are similar to those obtained in other works, the explainability provided by our model allows us to draw very interesting conclusions about which characteristics are more influential when estimating a football player's salary.



---

Yo, **Javier Moreno Morón**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77769129B, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Javier Moreno Morón

Granada a 20 de junio de 2024.





---

D. **Pablo Mesejo Santiago**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

D. **Óscar Cordon García**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado ***Modelos predictivos aplicados al análisis del salario de futbolistas profesionales***, ha sido realizado bajo su supervisión por **Javier Moreno Morón**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 20 de junio de 2024.

**Los directores:**

**Pablo Mesejo Santiago**

**Óscar Cordon García**



# Agradecimientos

Quiero agradecer a Pablo y Óscar por darme la oportunidad de realizar este proyecto junto a ellos y por toda la ayuda brindada a lo largo del mismo. Agradecimientos también para mi familia, pues gracias a ellos estoy hoy donde estoy.



# Índice general

<b>1. Introducción</b>	<b>9</b>
1.1. Definición del Problema . . . . .	9
1.2. Motivación . . . . .	11
1.3. Objetivos . . . . .	12
<b>2. Planificación</b>	<b>15</b>
2.1. Planificación del proyecto . . . . .	15
2.2. <i>Framework</i> y entorno de ejecución . . . . .	19
<b>3. Fundamentos teóricos</b>	<b>21</b>
3.1. Aprendizaje Automático . . . . .	21
3.1.1. Tipologías de modelos considerados . . . . .	22
3.1.2. Métricas de error empleadas . . . . .	27
3.2. Selección de características . . . . .	28
3.2.1. Información Mutua para Selección de Características .	29
3.2.2. Coeficiente $r$ de <i>Pearson</i> y <i>F-value</i> . . . . .	29
<b>4. Estado del arte</b>	<b>31</b>
4.1. Estimación de salarios deportivos . . . . .	31
4.1.1. Métodos ajenos a la Inteligencia Artificial . . . . .	31
4.1.2. Métodos relacionados con la Inteligencia Artificial . .	33
4.2. Explicabilidad de las predicciones . . . . .	36
<b>5. Materiales y métodos</b>	<b>39</b>
5.1. Conjunto de datos . . . . .	39
5.1.1. Extracción de los datos . . . . .	39
5.1.2. Preprocesamiento de los datos . . . . .	41
5.1.3. Visualización y análisis preliminar del conjunto de datos	44
5.2. Selección de características . . . . .	50
5.2.1. Heurísticas propuestas . . . . .	51
5.3. Protocolo de validación . . . . .	53
5.4. Aplicación <i>web</i> . . . . .	54
5.4.1. Diseño . . . . .	54
5.4.2. Desarrollo . . . . .	55

<b>6. Experimentos y análisis de resultados</b>	<b>61</b>
6.1. Experimentos iniciales . . . . .	61
6.1.1. k-NN . . . . .	64
6.1.2. Regresión lineal . . . . .	65
6.1.3. <i>Random Forest</i> . . . . .	66
6.1.4. <i>Gradient Boosting</i> . . . . .	68
6.1.5. Perceptrón Multicapa . . . . .	68
6.1.6. Conclusiones iniciales . . . . .	70
6.2. Pruebas sin porteros . . . . .	74
6.3. Pruebas considerando una única posición . . . . .	78
6.4. Experimentos de selección de características . . . . .	81
6.4.1. Información Mutua de Selección de Características . .	81
6.4.2. Coeficiente r de <i>Pearson</i> y <i>F-value</i> . . . . .	81
6.4.3. Eliminación Recursiva de Características . . . . .	82
6.4.4. Selector de Características Las Vegas (filtro) . . . . .	82
6.4.5. Selector de Características Las Vegas (envoltorio) . . .	83
6.4.6. Conclusiones en la selección de características . . . . .	83
6.5. Experimentos finales . . . . .	84
6.5.1. Explicabilidad de las predicciones . . . . .	89
<b>7. Conclusiones y trabajos futuros</b>	<b>95</b>
<b>Bibliografía</b>	<b>97</b>
<b>A. Características del dataset</b>	<b>107</b>
<b>B. Resumen de las variables del problema</b>	<b>111</b>

# Índice de figuras

1.1. Esquema del procedimiento llevado a cabo para estimar el salario de un futbolista dadas sus estadísticas, así como explicar dicha predicción. . . . .	12
2.1. Modelo de desarrollo en cascada. . . . .	15
2.2. Modelo de desarrollo en cascada con alimentación. . . . .	16
2.3. Diagrama de <i>Gantt</i> con la cronografía de trabajo inicial. . . .	18
2.4. Diagrama de <i>Gantt</i> con la cronografía de trabajo final. Nótese que los cambios realizados se muestran con otra fuente de escritura. . . . .	18
3.1. Función resultante de entrenar un modelo k-NN de regresión	23
3.2. Esquema de un árbol de regresión . . . . .	24
3.3. Algoritmo de <i>Gradient Boosting</i> . . . . .	26
3.4. Estructura de una unidad del MLP. . . . .	26
3.5. Estructura de un MLP. . . . .	27
4.1. Documentos publicados anualmente relacionados con estimación del salario deportivo. Búsqueda realizada en <i>Scopus</i> con la <i>query</i> : TITLE-ABS-KEY ( ( ( wage OR salary OR ( market AND value ) ) AND ( estimation OR prediction ) AND ( sport OR football OR soccer ) ) ). Fecha: 02/05/2024. . . . .	32
4.2. Documentos publicados anualmente relacionados con estimación del salario deportivo y el uso de IA. Búsqueda realizada en <i>Scopus</i> con la <i>query</i> : TITLE-ABS-KEY ( ( ( ( deep AND learning ) OR ( machine AND learning ) OR ( soft AND computing ) OR ( artificial AND intelligence ) OR ( data AND mining ) ) AND ( wage OR salary OR ( market AND value ) ) AND ( estimation OR prediction ) AND ( sport OR football OR soccer ) ) ). Fecha: 02/05/2024. . . . .	33

4.3. Documentos publicados anualmente relacionados con la explicabilidad de las predicciones. Búsqueda realizada en <i>Scopus</i> con la <i>query</i> : TITLE-ABS-KEY ( ( explainable AND ( ( deep AND learning ) OR ( machine AND learning ) OR ( soft AND computing ) OR ( artificial AND intelligence ) OR ( data AND mining ) ) ) ). Rango de la búsqueda: Hasta 2023. Fecha: 05/05/2024. . . . .	36
5.1. Ejemplo de entradas disponibles en la tabla de estadísticas generales para 'Jugador' == 'João Cancelo', tras el primer paso del preprocesamiento. . . . .	42
5.2. Ejemplo de entradas disponibles en la tabla de estadísticas generales para 'Jugador' == 'João Cancelo', tras el segundo paso del preprocesamiento. . . . .	42
5.3. Ejemplo de entradas disponibles en <i>dataset</i> para 'Jugador' == 'Aaron Connolly', tras el tercer paso del preprocesamiento. . . . .	43
5.4. Histograma con la frecuencia de cada posición en el <i>dataset</i> . . . . .	45
5.5. Distribución de los salarios de los futbolistas en 50 <i>bins</i> . . . . .	46
5.6. Diagrama de caja con los salarios de los futbolistas. . . . .	47
5.7. Distribución de los salarios de los futbolistas tras escalarlos logarítmicamente. . . . .	47
5.8. Distribución de los jugadores en base a la posición del futbolista. . . . .	48
5.9. Distribución de los porteros. . . . .	49
5.10. Distribución de los porteros en base a los minutos jugados. . . . .	49
5.11. Distribución de los jugadores que han jugado menos minutos en base a la posición del futbolista. . . . .	50
5.12. Esquema de <i>cross validation</i> con $k = 5$ . . . . .	54
5.13. Diagrama de casos de uso de la aplicación <i>web</i> . . . . .	55
5.14. Diagrama de flujo de la aplicación <i>web</i> . . . . .	56
5.15. Estructura principal de la aplicación <i>web</i> . . . . .	58
5.16. Ejemplo de leyenda de ayuda de la aplicación <i>web</i> . . . . .	58
5.17. Resultado tras ejecutar el modelo que se imprime en la aplicación <i>web</i> . . . . .	59
6.1. Varianza explicada en función del número de componentes principales con el que nos quedamos. . . . .	63
6.2. Entrenamiento progresivo para el modelo <i>Random Forest</i> . . . . .	67
6.3. Entrenamiento progresivo para el modelo <i>Gradient Boosting</i> . . . . .	69
6.4. Entrenamiento progresivo para el modelo de Perceptrón Multicapa. . . . .	71
6.5. Resultados de los modelos iniciales para el conjunto de entrenamiento. . . . .	72
6.6. Resultados de los modelos iniciales para el conjunto de validación. . . . .	73



6.7. Distribución de los jugadores de campo en base a la posición del futbolista. . . . .	74
6.8. Resultados de los modelos entrenados sin porteros para el conjunto de entrenamiento. . . . .	76
6.9. Resultados de los modelos entrenados sin porteros para el conjunto de validación. . . . .	77
6.10. Resultados de los modelos entrenados solo con centrocampistas para el conjunto de entrenamiento. . . . .	79
6.11. Resultados de los modelos entrenados solo con centrocampistas para el conjunto de validación. . . . .	80
6.12. Comparativa de las técnicas de selección de características con mejores resultados. . . . .	85
6.13. Resultados para el conjunto reducido de entrenamiento de los mejores modelos. . . . .	87
6.14. Resultados para el conjunto reducido de validación de los mejores modelos. . . . .	88
6.15. Dispersión de las predicciones del conjunto de test. . . . .	89
6.16. Gráfico de pastel de las características con más de un 2 % de importancia. . . . .	92
6.17. Gráfico de pastel de las características con menos de un 2 % de importancia. . . . .	93



# Índice de cuadros

2.1. Desglose del coste total del proyecto. . . . .	19
5.1. Distribución en cuartiles del salario. . . . .	46
6.1. Conjunto de hiperparámetros refinados en k-NN. . . . .	65
6.2. Conjunto de hiperparámetros refinados en regresión lineal. . .	66
6.3. Conjunto de hiperparámetros refinados en <i>Random Forest</i> . . .	67
6.4. Conjunto de hiperparámetros refinados en <i>Gradient Boosting</i> . .	69
6.5. Conjunto de hiperparámetros refinados en el Perceptrón Mul- ticapa. . . . .	70
6.6. Resultados obtenidos para todos los modelos iniciales. . . . .	72
6.7. Resultados obtenidos para los modelos entrenados con el <i>da-</i> <i>taset</i> sin porteros. . . . .	75
6.8. Resultados obtenidos para los modelos entrenados con el <i>da-</i> <i>taset</i> de solo centrocampistas. . . . .	79
6.9. Resultados obtenidos por MIFS para distintos números de características ( $k$ ) para el conjunto de validación. . . . .	81
6.10. Resultados obtenidos por <i>Pearson's r</i> para distintos números de características ( $k$ ) para el conjunto de validación. . . . .	82
6.11. Resultados obtenidos por <i>F value</i> para distintos números de características ( $k$ ) para el conjunto de validación. . . . .	82
6.12. Resultados obtenidos por RFE para distintos números de ca- racterísticas ( $k$ ) para el conjunto de validación. . . . .	82
6.13. Resultados obtenidos por LVF para las distintas soluciones obtenidas para el conjunto de validación. . . . .	83
6.14. Resultados obtenidos por LVW para las distintas soluciones obtenidas para el conjunto de validación. . . . .	83
6.15. Mejores resultados de las técnicas de selección de característi- cas para el conjunto de validación. . . . .	84
6.16. <i>Ranking</i> de importancia de las características en el modelo .	91
B.1. Resumen de las variables categóricas . . . . .	111
B.2. Resumen de las variables continuas. Parte 1 . . . . .	112
B.3. Resumen de las variables continuas. Parte 2 . . . . .	113

B.4. Resumen de las variables continuas. Parte 3 . . . . .	114
B.5. Resumen de las variables continuas. Parte 4 . . . . .	115

# Capítulo 1

## Introducción

### 1.1. Definición del Problema

El fútbol es un mercado que mueve miles de millones. Lo que para muchos es entretenimiento o pasión, para otros es un negocio. Y no un negocio menor: solo en España, en la temporada 2016/2017, se superaron los 15.688 millones de euros generados por la industria de fútbol profesional [31]. Estos elevadísimos ingresos son responsables de que los salarios sean igualmente altos. Además, todos los agentes implicados quieren obtener el máximo beneficio en la negociación de dichos sueldos.

El problema viene entonces en cómo estimar el salario de un futbolista. Es muy importante tener un valor aproximado calculado de forma objetiva, ya que existen tres actores principales [12] influenciados por el salario de un único futbolista. Es vital que este sueldo esté ajustado a la realidad para evitar problemas entre las distintas partes, las cuales son:

- **Futbolista:** Es el beneficiario del salario. Es en base a sus estadísticas y juego sobre lo que se calcula su salario. Su principal problema es que no es él mismo el que negocia el sueldo, sino que depende de un representante, lo cual puede dar lugar a importantes problemas y conflictos de intereses.
- **Representante del futbolista:** Es el encargado de negociar con el club el salario del futbolista. El problema es que en muchas ocasiones, en vez de defender los intereses de su representado, buscará su propio beneficio [34]. Aparte de que en otras ocasiones, sobre todo con futbolistas jóvenes y de países en vías de desarrollo, los agentes pueden usar su posición de poder para quedarse con hasta el 50 % del salario del jugador [25].
- **Club:** El último actor es el equipo en el que juega o va a jugar el

futbolista, el cual delega bien en el entrenador, bien en el director deportivo las negociaciones. El club busca minimizar el gasto en salarios, incluso pagando menos de lo que el futbolista 'merecería cobrar'<sup>1</sup>.

Por ello, es de vital importancia tener un método fiable para estimar el salario de un futbolista. De este modo, el propio futbolista podría tener más autonomía a la hora de negociar y no ser tan dependiente de su agente[41]. Igualmente, permitiría tener más argumentos para despedirle en caso de que éste negocie de forma ineficaz. El propio representante también se vería beneficiado al disponer de una herramienta de Inteligencia de Negocio, capaz de calcular el salario aproximado que corresponde a su representado, así como de explicar el porqué de dicha estimación. El agente tendría más poder de negociación en ocasiones en las que el club ofrezca un salario inferior al que corresponde. Por último, a los clubes también les interesaría disponer de una herramienta como esta para el caso contrario, poder tener argumentos para rebatir al agente cuando este solicite un salario demasiado elevado para su jugador.

Por último, existe un cuarto actor que entra en juego y al que este estudio le puede resultar de gran ayuda. Estamos hablando de la UEFA<sup>2</sup>, la FIFA<sup>3</sup> y en general todas las instituciones encargadas de las regulaciones financieras en las distintas ligas y competiciones deportivas. Un ejemplo es la propia UEFA, que en la temporada 2011/2012 lanzó una serie de regulaciones en pro del juego limpio financiero. Estas medidas fueron propuestas con la intención de que los clubes no se endeuden y compitan en igualdad de condiciones económicamente hablando [26].

Incumplir estas normas financieras acarrea sanciones. Es por ello que muchos equipos cometen irregularidades como traspasos de jugadores entre equipos inflados de precio [86] o acuerdos de salarios inferiores a los reales con jugadores pagando el restante por vías alternativas [60]. Esto se hace para evitar tener pérdidas en sus cuentas y ajustarse al Juego Limpio Financiero (FFP<sup>4</sup>).

En resumen, con un estimador fiable y explicativo del salario se podrían facilitar mucho las negociaciones entre club, jugador y representante. Del mismo modo, sería una herramienta de gran ayuda para las organizaciones mencionadas a la hora de detectar posibles salarios irregulares e investigarlos.

---

<sup>1</sup>Refiriéndonos a un salario estimado como podría ser el calculado en este estudio.

<sup>2</sup>*Union of European Football Associations.*

<sup>3</sup>*Fédération Internationale de Football Association.*

<sup>4</sup>*Financial Fair Play*, es decir, las normas de juego limpio financiero a las que nos referimos anteriormente.

## 1.2. Motivación

Actualmente la estimación del sueldo de un futbolista es un procedimiento totalmente subjetivo, como ya se mencionó en el apartado anterior. Dotar de automatización y explicabilidad dicho proceso será de gran utilidad tanto para los reguladores económicos del mundo del fútbol, como para los tres actores que forman parte de una negociación. En el caso de los primeros, porque podrán identificar más fácilmente los salarios irregulares e investigar dichas irregularidades más veloz y eficazmente. En el caso de los segundos, por un lado los jugadores cobrarán los sueldos que merecen, ni más, ni menos. Por último, clubes y representantes dispondrán de herramientas para poder hacer mejor su trabajo.

El auge del *Big Data*<sup>5</sup> y la Inteligencia Artificial (IA) [78] en los últimos tiempos ha permitido su uso en numerosas aplicaciones, incluido el propio fútbol. En este contexto, ya se aplican técnicas de IA para calcular los goles esperados [79] que debería haber marcado un equipo o un jugador y, en general, para ayudar a los entrenadores a desarrollar más eficientemente sus tácticas y jugadores [5, 56]. Pero no se queda solo ahí, aspectos subjetivos del fútbol, como las decisiones arbitrales, que siempre han generado polémica, también se ayudan de la IA para mejorar y ser más eficaces [75]. Algunos ejemplos son la inclusión del fuera de juego automático o la tecnología *Goal-line*.

Sin embargo, a la hora de negociar los salarios de los futbolistas se siguen usando métodos tradicionales de forma general. Y digo por lo general, porque ya se están empezando a ver los primeros casos de uso del *Big Data* y la IA para calcular el salario estimado que un jugador merece cobrar. El caso de Kevin De Bruyne, que tras despedir a su agente por varios delitos de fraude, negoció él mismo su renovación aportando datos y estadísticas que justificaban una subida de sueldo [41], se ha convertido en un caso paradigmático. Con ello se abren las puertas al uso del *Big Data* a la hora de negociar salarios y solo es cuestión de tiempo que se empiece a popularizar su uso.

Por todo ello, considero de gran interés indagar en este problema. No obstante, el hecho de que el uso de estas técnicas esté ganando fama hace que ya existan algunos trabajos sobre el tema [100, 9, 55, 63, 101]. Desgraciadamente estas propuestas tienen algunas limitaciones como el uso de estadísticas de videojuego como el *Football Manager* o el FIFA<sup>6</sup>, en lugar de obtener características reales de los futbolistas, o la realización de una

---

<sup>5</sup>El *Big Data* se refiere al manejo y análisis de grandes cantidades de datos.

<sup>6</sup>No confundir con la FIFA, la organización. En este caso nos referimos a una saga de videojuegos de fútbol publicados durante más de 20 años por *Electronic Arts* bajo el sello de *EA Sports*. [87]

selección subjetiva de estadísticas a usar, reduciendo así la dimensionalidad sin ningún criterio claro.

Viendo estas deficiencias, he considerado interesante subsanarlas en mi trabajo de fin de grado. Por un lado, haciendo una gran selección de datos de distintas *webs*. Por otro lado, utilizando técnicas de selección de características, en lugar de elegir manualmente con qué estadísticas quedarme. Por último, también como aportación interesante, he considerado que no solo se estime el salario que debe cobrar el jugador, sino que también se explique por qué se llega a dicha conclusión, es decir, dotar al modelo de explicabilidad. Esto permite al usuario entender, confiar y manejar de forma efectiva estas predicciones [8]. Como hemos visto anteriormente, esta cualidad puede resultar de gran ayuda para todos los actores que forman parte en la negociación de un contrato, pues ellos mismos podrán dar las explicaciones de por qué el futbolista debe cobrar tal salario.

### 1.3. Objetivos

El objetivo principal de este trabajo es, por un lado, obtener un estimador fiable capaz de predecir de forma robusta y precisa qué salario debe cobrar un futbolista en la temporada  $x$ , dadas sus estadísticas de la temporada  $x - 1$ . Por otro, se busca obtener una explicación de por qué se ha estimado dicho salario, es decir, se busca dotar de explicabilidad al modelo entrenado, de forma que este sea más confiable e interpretable. Ambos propósitos, así como el proceso hasta conseguirlos, se muestran de forma esquemática en la figura 1.1.

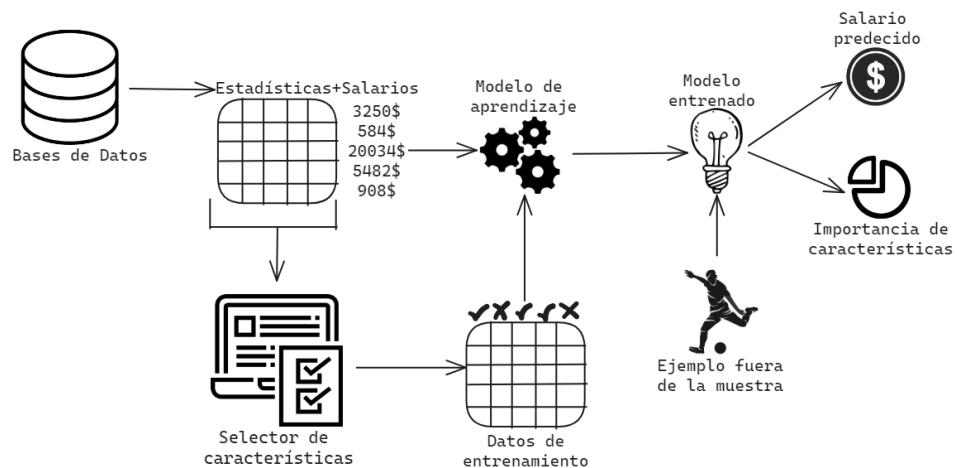


Figura 1.1: Esquema del procedimiento llevado a cabo para estimar el salario de un futbolista dadas sus estadísticas, así como explicar dicha predicción.



Este doble objetivo principal se desglosa en los siguientes objetivos específicos:

- Realizar un análisis pausado del estado del arte actual, prestando especial atención a las distintas técnicas utilizadas y su desempeño.
- Plantear y diseñar cómo va a ser el conjunto de datos con el que entrenar, y estudiar cómo obtenerlo.
- Realizar una selección de las características más importantes.
- Seleccionar un conjunto de hipótesis, teniendo especialmente en cuenta que necesitamos modelos de aprendizaje automático que sean capaces de aportar explicabilidad a sus predicciones.
- Implementar el conjunto de hipótesis y seleccionar el mejor modelo empleando un protocolo de validación experimental riguroso.
- Comparar los resultados obtenidos por nuestro modelo con los conseguidos en los otros estudios analizados.
- Analizar el estimador propuesto, para conseguir obtener la explicación de las predicciones que éste realiza.



## Capítulo 2

# Planificación

### 2.1. Planificación del proyecto

Debido a la naturaleza del trabajo, donde esencialmente tenemos que hacer una extracción y selección de características por un lado, y seleccionar y entrenar un modelo de Aprendizaje Automático para realizar las predicciones por otro lado. Nos deja con un proyecto donde a priori la planificación del desarrollo *software* no será muy compleja. Por lo que decido plantear el desarrollo con un estilo de vida en cascada (figura 2.1), en el cual no se puede iniciar una etapa del proyecto hasta haber acabado la anterior, y una vez finalizada una etapa no se puede volver a la misma.

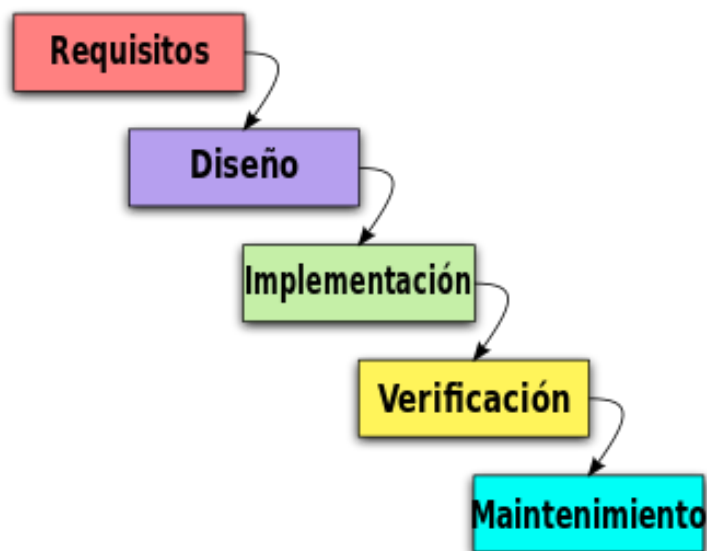


Figura 2.1: Modelo de desarrollo en cascada.

Aunque en principio esta fuera una buena opción, cuando llegué a la parte de experimentación vi que era necesario realizar cambios en la implementación, es decir retroceder en las etapas de planificación, lo cual no se corresponde con el desarrollo en cascada. Por esta razón, tomé la decisión de modificar el modelo trabajo por uno de cascada con realimentación (figura 2.2). Su principal diferencia respecto al modelo de cascada básico es, que en este caso sí se puede retroceder en las etapas del proyecto para hacer modificaciones. Decido pasarme a este modelo de trabajo por su similitud al método de planificación en cascada que era el modelo que venía utilizando desde el inicio del proyecto. En las etapas finales, aprovechando este cambio de metodología y disponiendo aún de tiempo para mejorar el proyecto, decido crear una pequeña aplicación *web*.

Por otro lado, acordé con los tutores tener reuniones mensuales, en las cuales comentaba los avances realizados, recibía *feedback* por su parte, y acordábamos los próximos pasos a desarrollar. Una metodología similar al método de trabajo *Sprint*, donde la tarea principal se divide en entregables más pequeños y posteriormente en las reuniones, se va informando acerca del trabajo realizado. Aunque sin llegar a ser una fiel aplicación del mismo.

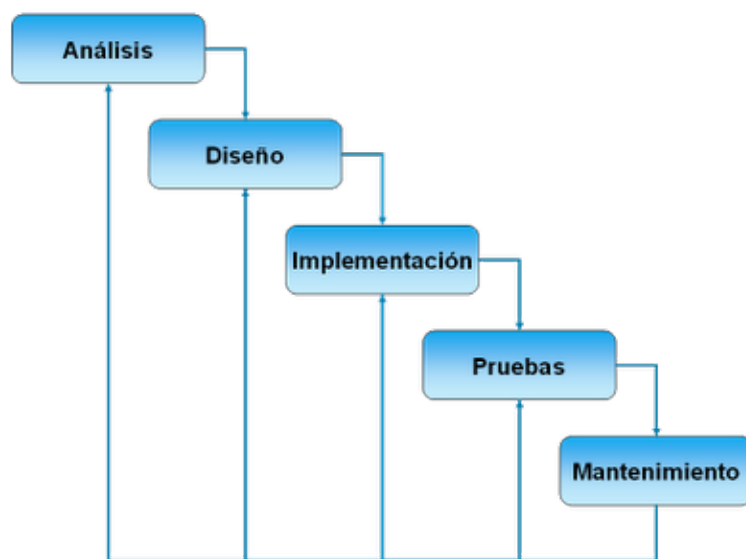


Figura 2.2: Modelo de desarrollo en cascada con alimentación.

Una vez definido como se desarrollará el trabajo, tocaba organizar los tiempos a invertir en cada etapa del desarrollo. Al ser el TFG una asignatura equivalente a 12 créditos ECTS, lo cual corresponde a 300 horas (25h/crédito). Mi idea fue repartir esas 300 horas entre todas las etapas del proyecto, como se puede observar en la figura 2.3. Sumando todas las horas de la cronografía inicial, hacen 270 horas de trabajo, menos de las 300 correspon-

dientes. Es por ello, que decido implementar esta *app web* que menciono anteriormente. Teniendo en cuenta esta nueva tarea en la planificación, la cronografía final queda como se muestra en la figura 2.4, con unas 315 horas de trabajo estimadas. Entrando más en detalle, las partes del proyecto y el tiempo dedicado han sido las siguientes:

- **Análisis:** Esta parte la dedico inicialmente a comentar la idea con los tutores, así como a explorar las posibles dificultades que voy a poder encontrarme. Tras tener claro el proyecto, paso a una fase de análisis del estado del arte existente en la materia, prestando especial atención a las distintas técnicas usadas y como hacer que mi trabajo resulte innovador. Dedico todo octubre y mitad de noviembre para esta etapa, es decir, 45 horas. No obstante, ya había tenido alguna reunión preliminar antes de estas fechas, que siendo tenidas en cuenta elevarían el tiempo empleado en esta fase a unas 50 horas.
- **Diseño:** Aquí, me dedico por un lado a investigar técnicas de extracción de datos, así como a explorar de qué páginas *webs* obtenerlos. Por otro lado, a buscar métodos de selección de características. Por último, selecciono posibles modelos y métricas de Aprendizaje Automático a utilizar. Con toda esta información, paso a diseñar el *software* necesario para realizar las distintas tareas. Decido invertir en esta parte lo que queda de noviembre y diciembre, otras 45 horas. A este tiempo, hay que sumarle las 15 horas usadas para diseñar la aplicación *web*.
- **Implementación:** En este apartado, desarrollo todo el *software* diseñado en la anterior etapa. Esta es la sección por la cual cambié el modelo de trabajo a cascada con alimentación, ya que cuando pasé a la etapa de pruebas me vi obligado a retroceder y hacer nuevos cambios en la implementación. Inicialmente reservo el mes de enero para esta fase. No obstante, debido a los contratiempos ya comentados hago una modificación y guardo también las primeras quincenas de marzo y junio para dedicar a esta etapa, en total 60 horas.
- **Pruebas:** Por último, se comprueba que todas las implementaciones funcionan correctamente y se pasa a la fase de experimentación. Como ya comento en la etapa anterior, en esta parte aparecieron fallos y nuevas ideas de implementaciones que probar, por lo que se tuvo que retroceder a la fase de implementación. Tras finalizar los experimentos, en esta etapa se hace también el análisis de los resultados. Como punto final, se comprueba que la *web* funcione correctamente. Para esta parte, se estimó usar inicialmente el resto de meses, desde febrero hasta mayo. Al final dedico todo ese tiempo, solo que en lugar de la primera quincena de marzo, que como he comentado antes dedico a implementación, uso la última quincena de junio. Por lo que

en resumen, el tiempo invertido en esta etapa termina siendo de unas 120 horas.

Adicionalmente a estas fases, realizo una última etapa de revisión, en la cual termino la memoria del TFG a la par que la repaso y corrijo errores que esta pudiera tener.

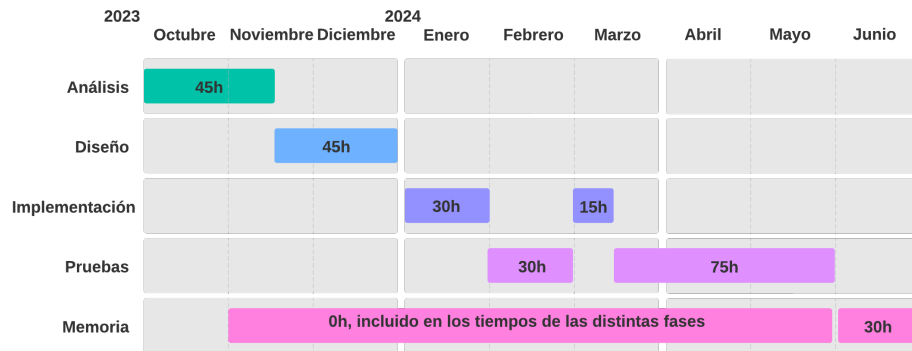


Figura 2.3: Diagrama de *Gantt* con la cronografía de trabajo inicial.

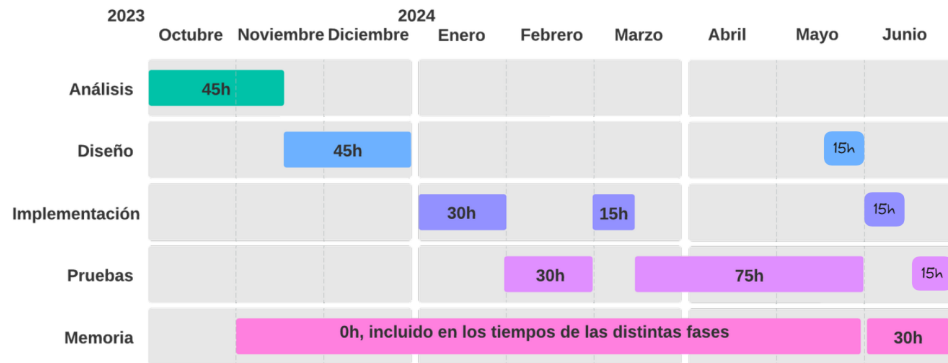


Figura 2.4: Diagrama de *Gantt* con la cronografía de trabajo final. Nótese que los cambios realizados se muestran con otra fuente de escritura.

Teniendo todo esto en cuenta y sabiendo que los honorarios de un investigador senior en España son de 30€/h [42], el gasto económico en personal para el proyecto es de unos 9450€. A esta cantidad habría que sumarle por otro lado, el coste del *hardware* necesario para la investigación, el cual se puede ver de forma más detallada en la tabla 2.1, y que cuyo precio total final sería de unos 646€. Otra opción podría haber sido pagar un alquiler de una unidad de cómputo en la nube, como puede ser *Google Colab Pro*. Pero finalmente nos decantamos por tener una unidad física, para tener la

mayor disponibilidad posible. Por último, no es necesaria ninguna inversión en *software*, ya que todas las librerías usadas en el proyecto son gratuitas.

Cuadro 2.1: Desglose del coste total del proyecto.

Concepto	Coste
Salarios	9450€
i5-10400F 2.90 GHz	104.99€[70]
RAM 16GB DDR4 3200MHz	42.99€[69]
Almacenamiento SSD 500GB	39.26€[71]
Nvidia GeForce RTX 4060 VRAM 8GB	309€[68]
Más hardware (Placa base, refrigeración, cableado...)	150€
Total	10096€

## 2.2. *Framework* y entorno de ejecución

Como lenguaje de programación para realizar el proyecto decido usar *Python*. Esto es debido a que ya existen multitud de librerías implementadas para usar en *Machine Learning*. Otros lenguajes como *R* o *MatLab* también cumplen esta características, pero me decanto por *Python* por ser el más utilizado a lo largo de la carrera. Más concretamente realizo los experimentos en cuadernos interactivos de *Jupyter*, ya que me permiten hacer varias pruebas de un experimento sin tener que ejecutar todo el *script* en cada una.

Para el control de versiones utilizo *GitHub*. En el siguiente repositorio: <https://github.com/JMMelcrack/code>, se puede acceder a todos los *Notebooks* donde realizo los experimentos, así como a los *scripts* para obtener y preprocesar los datos. En cuanto a la metodología de trabajo seguida, al tener el proyecto una carga de desarrollo *software* baja opto por no utilizar ramas y trabajar siempre en la rama *main*. No obstante, en un punto del proyecto donde encuentro una mejora para el preprocesado de los datos, he de actualizar todos los ficheros. Para esa situación si encuentro útil trabajar en una rama aparte sin modificar el trabajo de *main*. Una vez obtenidos los resultados finales con el preprocesamiento modificado, uno ambas ramas. Para ese y otros problemas que iban surgiendo, voy generando *issues* en el propio repositorio de *GitHub*. Las *issues*, son recordatorios de la existencia de un error o una funcionalidad a crear/mejorar, estas son de gran utilidad para llevar un registro de tareas pendientes por hacer en el proyecto.

Como entorno de edición utilizo *Visual Studio Code* por todas las funcionalidades que ofrece, como soporte para la depuración, control integrado de *Git*, resaltado de sintaxis, finalización inteligente de código o refactorización de código. Por último las principales librerías que uso son:

- ***NumPy***: Para realizar todo tipo de operaciones lógicas a matrices y vectores, como por ejemplo transformar la variable a predecir [90].
- ***Pandas***: Para almacenar todo el *dataset* y poder realizar rápidas modificaciones y búsquedas en el mismo [65].
- ***Sklearn***: Para poder realizar los entrenamientos, ya que esta librería cuenta con implementaciones de la mayoría de modelos de *Machine Learning*. También la uso por ejemplo, para visualización de datos o selección de características [91].
- ***Matplotlib* y *Seaborn***: Para realizar gráficas, tablas y visualizar conjuntos de datos [92, 96].
- ***BeautifulSoup***: Para obtener los datos de los códigos HTML de las distintas *webs* [76].
- ***Streamlit***: Para poder crear y desplegar una pequeña web en la cual mostrar el funcionamiento de mi modelo final. [40].



## Capítulo 3

# Fundamentos teóricos

En este capítulo nos centraremos en explicar los fundamentos teóricos en los cuales nos hemos basado para realizar la investigación.

### 3.1. Aprendizaje Automático

El aprendizaje automático o *Machine Learning* (ML) [2, 7, 14, 64] es una rama de la inteligencia artificial (IA) que se centra en el desarrollo de sistemas que pueden aprender y mejorar automáticamente a partir de la experiencia sin ser programados explícitamente. En esencia, se trata de enseñar a las máquinas a aprender patrones a partir de datos para realizar tareas específicas de manera más eficiente y precisa con el tiempo.

Dentro del *Machine Learning* existen diversos tipos de aprendizaje. Por ejemplo, está el aprendizaje supervisado, en el cual los datos de entrada que se proporcionan al modelo para el entrenamiento están debidamente etiquetados con la salida deseada. También está el aprendizaje no supervisado, donde el aprendizaje se lleva a cabo sobre un conjunto de ejemplos formado tan solo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos, por lo que en este caso el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas. Por último, encontramos el aprendizaje por refuerzo, en el cual el sistema aprende a través de la interacción con el entorno y la retroalimentación recibida sobre su desempeño en las tareas que realiza, utilizando un enfoque de ensayo y error.

Nuestro proyecto se construye sobre el aprendizaje supervisado, ya que nuestros datos de entrada sí están etiquetados. Dentro del aprendizaje supervisado destacan principalmente dos tipos de problemas. Los problemas de clasificación, donde el objetivo es asignar una etiqueta o clase a la entrada. Y los problemas de regresión, donde se busca predecir un valor numérico

continuo en función de una o más variables de entrada. Este último es el tipo de problema de nuestro proyecto. Ya que nosotros buscamos estimar el salario que cobra un futbolista, es decir predecir un valor numérico continuo.

### 3.1.1. Tipologías de modelos considerados

En la actualidad existen una gran diversidad de modelos que permiten realizar predicciones. Dependiendo de las características del problema, se seleccionan unos u otros, en lo que se conoce como el conjunto de hipótesis del problema.

#### Regresión lineal

La Regresión Lineal (RL) [36], es un enfoque para modelar la relación entre una variable dependiente escalar  $y$  y una o más variables explicativas denotadas por  $X$ .

Dado un *dataset*  $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^n$  de  $n$  unidades estadísticas. Un modelo de regresión lineal asume que la relación entre la variable dependiente  $y_i$  y el p-vector de variables explicativas  $x_i$  es lineal. Esta relación está modelada por un término de perturbación,  $\epsilon_i$ , una variable aleatoria no observada que añade ruido a la relación lineal entre la variable dependiente y las variables explicativas. Por tanto el modelo tiene la siguiente forma:

$$y_i = \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \epsilon_i = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i, \quad i = 1, \dots, n, \quad (3.1)$$

donde  $T$  hace referencia a la traspuesta, por lo que  $x_i^T \boldsymbol{\beta}$  es el producto interno entre los vectores  $x_i$  y  $\boldsymbol{\beta}$ .

#### *K-Nearest Neighbor*

*K-Nearest Neighbor* (k-NN) [10] es uno de los algoritmos de *Machine Learning* más simples, a pesar de esto es una técnica ampliamente utilizada que se aplica con éxito en una gran cantidad de problemas.

En k-NN [46] buscamos predecir una etiqueta  $y' \in R^d$  para nuevos ejemplos  $x' \in R^q$  basándose en un conjunto de  $N$  observaciones. El objetivo es aprender una función  $f : R^q \rightarrow R^d$  conocida como función de regresión. Para un ejemplo desconocido  $x'$ , k-NN calcula la media de los valores de la función de sus  $k$  vecinos más cercanos:

$$f_{k-NN}(x') = \frac{1}{k} \sum_{i \in N_k(x')} y_i \quad (3.2)$$

con  $N_k(x')$  conteniendo los índices de los  $k$  vecinos más cercanos de  $x'$ .

La idea de promediar en k-NN se basa en la suposición de la localidad de las funciones en el espacio de datos y etiquetas. En vecindarios locales de  $x_i$ , se espera que  $x'$  tenga etiquetas continuas similares  $f(x')$  como  $y_i$ . Por lo tanto, para un  $x'$  desconocido, la etiqueta debe ser similar a las etiquetas de los vecinos más cercanos, lo que se modela mediante el promedio de la etiqueta de los  $k$  ejemplos más cercanos. El resultado final de este modelado es una función  $f(x)$  como la de la imagen 3.1.

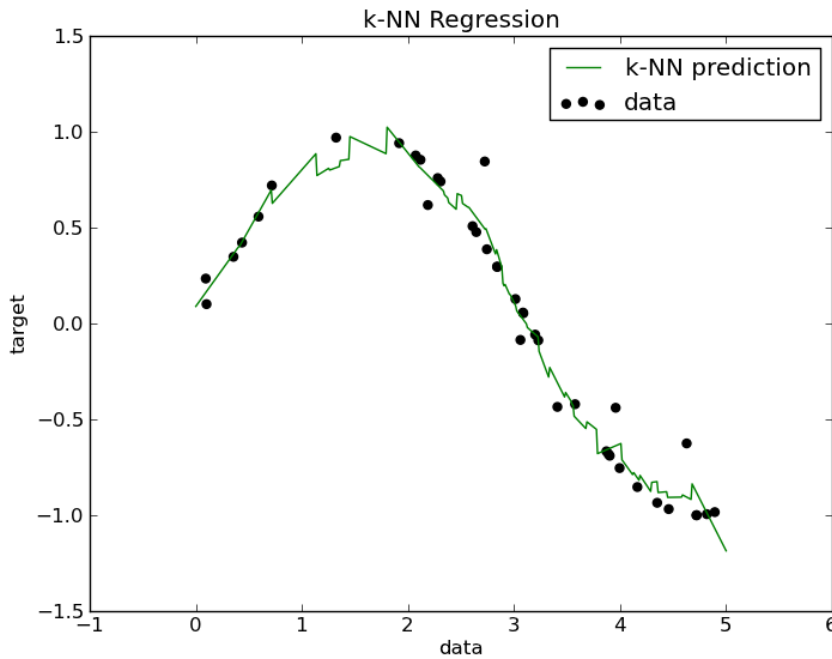


Figura 3.1: Función resultante de entrenar un modelo k-NN de regresión

### *Random Forest*

Como su nombre sugiere, un *Random Forest* (RF) [21, 17] es un *ensemble* [93] basado en árbol [23]. En la siguiente figura 3.2 se puede observar la estructura de un árbol en un problema de regresión, donde cada árbol depende de un conjunto de variables aleatorias. Más formalizado, para un vector aleatorio  $p$ -dimensional  $X = (X_1, \dots, X_p)^T$ , que representa la entrada real y una variable aleatoria  $Y$ , que representa las etiquetas reales, asumimos una distribución conjunta desconocida  $P_{XY}(X, Y)$ . El objetivo es determinar una función  $f(X)$  para predecir  $Y$ . Esta función se determina por una función de pérdida  $L(Y, f(X))$  y se define para minimizar el valor

de la pérdida.

$$E_{XY}(L(Y, f(X))) \quad (3.3)$$

donde los subíndices denotan la esperanza con respecto a la distribución conjunta de  $X$  e  $Y$ .

Intuitivamente,  $L(Y, f(X))$  es una medida que indica cómo de cerca está  $f(X)$  a  $Y$  y que penaliza valores de  $f(X)$  que están lejos de  $Y$ . Elecciones típicas de  $L$  son por ejemplo el error cuadrático medio, del cual hablaré en el apartado de métricas de error.

Los *ensembles* construyen  $f$  en términos de una colección de *base learners*,  $h_1(x), \dots, h_J(x)$ , y estos se combinan para obtener la función de predicción  $f(x)$ . En regresión, estos *base learners* se promedian:

$$f(x) = \frac{1}{J} \sum_{j=1}^J h_j(x) \quad (3.4)$$

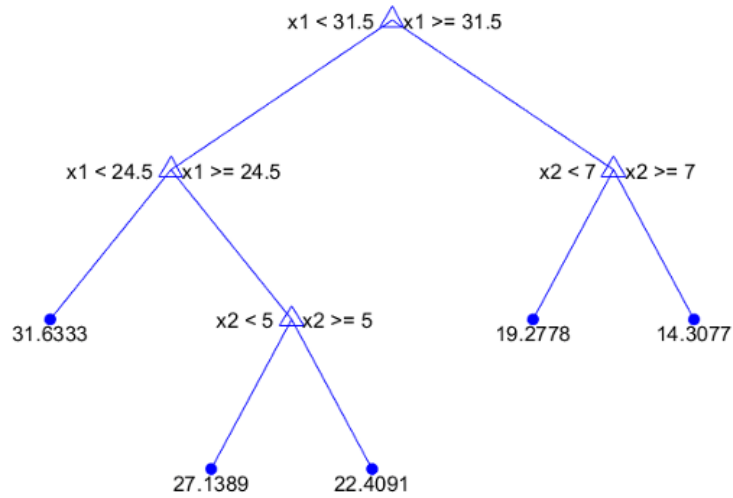


Figura 3.2: Esquema de un árbol de regresión

### ***Gradient Boosting***

*Gradient Boosting* (GB) [28, 29, 62], al igual que *Random Forest* es un *ensemble*, donde se crean múltiples modelos débiles y se combinan para

obtener un mejor rendimiento en su conjunto, el cual funciona de la siguiente manera.

El primer paso es hacer una función de predicción base  $F_0$  para la variable  $y$ , como podría ser la media de  $y$ . A continuación *Gradient Boosting* va iterando para mejorar los resultados de dicha predicción. Esta mejora se consigue enfocándose en los residuos,  $r_i$ , que son errores en la predicción de la iteración anterior.

Buscamos minimizar esos residuos, para así mejorar las predicciones. Conseguimos este objetivo construyendo uno de estos modelos débiles, con  $x$  como características y  $r_i$  como etiqueta a predecir. La razón detrás de esto es que si conseguimos encontrar algún tipo de patrón entre  $x$  y  $r_i$  construyendo este modelo, podremos reducir el residuo usándolo. El modelo que usaremos en nuestro caso es, árboles simples con pocos niveles de profundidad llamados *stump*.

La predicción  $y_i$  obtenida se añade a nuestra predicción inicial  $F_0$  para reducir el residuo. Pero no solo se añade esta  $y_i$ , ya que esto provocaría que el modelo se sobreajustara a los datos de entrenamiento. En su lugar,  $y_i$  se escala con el *learning rate*,  $v$ , y se añade a  $F_i$ . Por lo que una iteración quedaría así:

$$F_1 = F_0 + v * y_1 \quad (3.5)$$

Y seguimos iterando hasta que el modelo deje de mejorar. Este proceso algorítmico que he explicado se puede ver de manera más formal en la figura 3.3.

## Perceptrón Multicapa

Un Perceptrón Multicapa [6] (MLP) es el tipo de red neuronal más conocido y utilizado, el cual está formado por unidades (neuronas), como la que se puede ver en la figura 3.4. Cada una de estas unidades forma una suma ponderada de sus entradas, a la cual se agrega un término constante. Esta suma luego se pasa a través de una función, que a menudo se llama función de activación. Podemos ver en 3.4 cómo se hace la suma ponderada de sus *inputs*, a la que se le añade un término independiente. Y el resultado se pasa por una función de activación. Esta, suele ser una *sigmoide*, cuyas expresiones más comunes son las siguientes:

$$S(s) = \frac{1}{1 + e^{-s}} = \frac{1 + \tanh(s/2)}{2} \quad (3.6)$$

$$S(s) = \tanh(s) \quad (3.7)$$

### **Gradient Boosting Algorithm**

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \gamma)$$

2. for  $m = 1$  to  $M$ :

2-1. Compute residuals  $r_{im} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$  for  $i = 1, \dots, n$

2-2. Train regression tree with features  $x$  against  $r$  and create terminal node regions  $R_{jm}$  for  $j = 1, \dots, J_m$

2-3. Compute  $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$  for  $j = 1, \dots, J_m$

2-4. Update the model:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

Figura 3.3: Algoritmo de *Gradient Boosting*

$$S(s) = \arctan(s) \quad (3.8)$$

Por lo general las unidades están interconectadas mediante *feedforward*, es decir, con interconexiones que no forman ningún bucle, como se muestra en la Figura 3.5. Para algunos tipos de aplicaciones, también se utilizan redes recurrentes (es decir, no *feedforward*), en las cuales algunas de las interconexiones forman bucles.

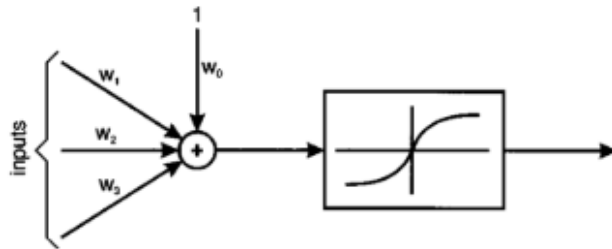


Figura 3.4: Estructura de una unidad del MLP.

El entrenamiento de estas redes normalmente se realiza de manera supervisada. Este entrenamiento se basa en la minimización de alguna medida

de error entre las salidas de la red y las salidas deseadas, lo que implica una propagación hacia atrás a través de la red que se está entrenando. Por esta razón, el algoritmo de entrenamiento normalmente se denomina *backpropagation* [77].

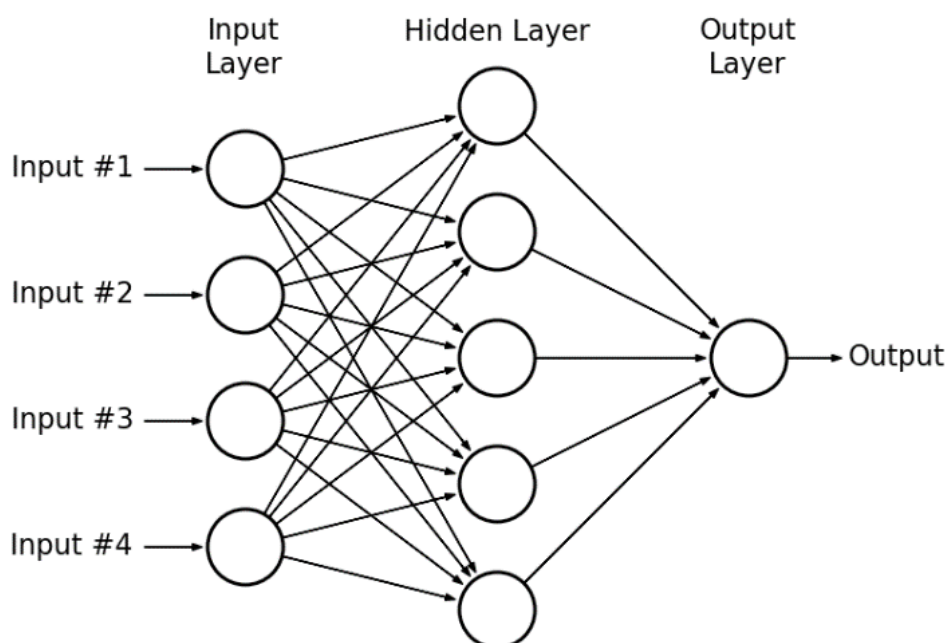


Figura 3.5: Estructura de un MLP.

### 3.1.2. Métricas de error empleadas

A la hora de comparar los modelos de nuestro conjunto de hipótesis entre sí, así como para saber como de bueno es un modelo, necesitamos algún tipo de medida que nos dé dicha información. Es por ello, que debemos seleccionar algunas métricas de error para obtener información del desempeño de los modelos a la hora realizar las predicciones. Las métricas elegidas son:

- **Coefficiente de determinación o  $R^2$  score [81]:** Es la proporción de varianza (de  $y$ ) que ha sido explicada por las variables independientes en el modelo. Esta métrica proporciona una indicación de la bondad del ajuste, y por tanto una medida de qué tan probable es que se predigan muestras no vistas por el modelo, a través de la proporción de varianza explicada.

Dado que la varianza es dependiente del conjunto de datos,  $R^2$  puede no ser significativamente comparable entre diferentes conjuntos de da-

tos. La puntuación óptima posible es 1.0 y puede ser negativa (porque el modelo puede ser arbitrariamente peor). Un modelo constante que siempre predice el valor esperado (promedio) de  $y$ , sin tener en cuenta las características de entrada, obtendría una puntuación de 0.0.

Si  $y'_i$  es el valor predicho de la  $i$ -ésima muestra e  $y_i$  es la correspondiente etiqueta real para un total de  $n$  muestras,  $R^2$  se define como:

$$R^2(y, y') = 1 - \frac{\sum_{i=1}^n (y_i - y'_i)^2}{\sum_{i=1}^n (y_i - Y')^2} \quad (3.9)$$

donde  $Y' = \frac{1}{n} \sum_{i=1}^n y_i$

- **Error absoluto medio (MAE) [81]:** Una métrica correspondiente al valor esperado de la pérdida de error absoluto o pérdida *l1-norm*.

Si  $y'_i$  es el valor predicho de la  $i$ -ésima muestra e  $y_i$  es la correspondiente etiqueta real, entonces MAE estimado sobre  $n$  muestras se define como:

$$MAE(y, y') = \frac{1}{n} \sum_{i=0}^{n-1} |y_i - y'_i| \quad (3.10)$$

- **Error cuadrático medio (MSE) [81]:** Una métrica correspondiente al valor esperado de la pérdida (cuadrática) media.

Si  $y'_i$  es el valor predicho de la  $i$ -ésima muestra e  $y_i$  es la correspondiente etiqueta real, entonces MSE estimado sobre  $n$  muestras se define como:

$$MSE(y, y') = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - y'_i)^2 \quad (3.11)$$

## 3.2. Selección de características

Un paso previo al diseño de un modelo de aprendizaje, es la selección del subconjunto de variables (características) relevantes para el entrenamiento del conjunto de objetos disponible.

Existen tres enfoques generales para la selección de características [47, 72]. Primero, el enfoque de filtro, que aprovecha las características generales de los datos de entrenamiento independientemente del algoritmo de ML [15]. Segundo, el enfoque de envoltura (*wrapper*), este explora la relación entre la relevancia y la selección óptima de subconjuntos de características y busca un subconjunto de características óptimo adaptado al modelo de *Machine Learning* especificado [45]. Y tercero, el enfoque integrado, el cual



se realiza con un algoritmo de aprendizaje específico que realiza la selección de características en el proceso de entrenamiento.

En nuestro caso utilizaremos métodos de tipo filtro en su mayoría y métodos de tipo envolvente. La idea es seleccionar las mejores características basadas en pruebas estadísticas univariadas, utilizando distintos algoritmos para calcularlas.

### 3.2.1. Información Mutua para Selección de Características

El algoritmo MIFS [11, 72], por sus siglas en inglés, es un método de filtro que selecciona a los atributos relevantes para la predicción con la menor correlación con otras características. El algoritmo calcula el valor  $I(C, f_i)$ , siendo  $C$  la variable de salida y  $f_i$  el atributo  $i$ -ésimo para cada atributo. Tras esto, se selecciona al de mayor información mutua como el primer elemento del subconjunto de atributos seleccionados  $G$ . Luego para el próximo  $f_i$  se calcula la información mutua de  $(f_i, g_i)$  y se seleccionan los  $m$  atributos  $f_i$  que maximizan el criterio MIFS. Este criterio incluye el término  $I(C, f_i)$  para garantizar la relevancia del atributo, pero introduce una penalidad  $\beta \sum_{g_j} I(f_i, g_j)$  para forzar que exista baja correlación.

Esta condición MIFS, se define entonces como:

$$MIFS = \operatorname{argmax}_{f_i} (I(C, f_i) - \beta \sum_{g_j} I(f_i, g_j)) \quad (3.12)$$

donde  $\beta$  es un parámetro de peso que es configurable, y es quien regula la importancia relativa de la información mutua entre el atributo candidato y los ya seleccionados respecto a la información mutua con la clase. Si toma valor 0, la expresión resultará en el cálculo de la relevancia individual. Si toma un valor grande, esta denotará mayor énfasis en la reducción de la correlación entre los atributos.

### 3.2.2. Coeficiente $r$ de *Pearson* y *F-value*

El coeficiente de correlación de *Pearson* [83, 97] es un algoritmo de filtro que, dados  $n$  pares de datos  $(x_i, y_i)$ , se define como:

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - x')(y_i - y')}{\sqrt{\sum_{i=1}^n (x_i - x')^2} \sqrt{\sum_{i=1}^n (y_i - y')^2}} \quad (3.13)$$

donde  $x'$  e  $y'$  representan la media muestral definida por  $x' = \frac{1}{n} \sum_{i=1}^n x_i$  (ídem para  $y'$ ).

Una vez tenemos representada la correlación de cada características, seleccionamos las  $k$  más importantes. Por otro lado el *F-value* [82] se obtiene transformando los resultados del coeficiente  $r$  de *Pearson* en *F score* y luego en *p-value*.

## Capítulo 4

# Estado del arte

### 4.1. Estimación de salarios deportivos

El fútbol no siempre ha movido las grandes cantidades de dinero que mueve ahora. En 1961, los sueldos llegaban como máximo a unas 20 libras semanales, que ajustado a la inflación, supone un salario de 350 libras semanales o 18000 anuales. En la década de los 90, es cuando los sueldos se empiezan a disparar, existiendo ya un salario medio de 77083 libras anuales. Estos siguen creciendo hasta superar los 3 millones anuales en 2019 [73]. Es por ello que hasta inicios de este siglo, no se tiene aún interés en estimar estos sueldos, como se puede observar en la figura 4.1, donde desde 1976 se han publicado 156 documentos relacionados con la estimación del salario de futbolistas. Pero no es hasta inicios de los 2000 que el número de publicaciones por año empieza a crecer. Vemos, eso sí, que cada año que pasa, este campo ganando más atención de los investigadores y aparecen más publicaciones acerca del mismo.

Por otro lado, en cuanto a estudios utilizando técnicas de *Machine Learning* o similares, solo contamos con 31 estudios publicados, como se puede observar en la figura 4.2. Y todos ellos muy recientes, los primeros trabajos datan del año 2014. Al igual que en el caso anterior, la tendencia es ascendente. Como comentaba en la introducción, a partir de 2021 se pueden empezar a ver los primeros usos prácticos de estudios sobre la estimación del salario [41], aunque aún queda mucho por investigar.

#### 4.1.1. Métodos ajenos a la Inteligencia Artificial

Uno de los primeros estudios, [85], de 1994, donde para realizar predicciones de salario en la *Major League Baseball* (MLB)<sup>1</sup> se utiliza un modelo

---

<sup>1</sup>Primera división de *baseball* de Estados Unidos.

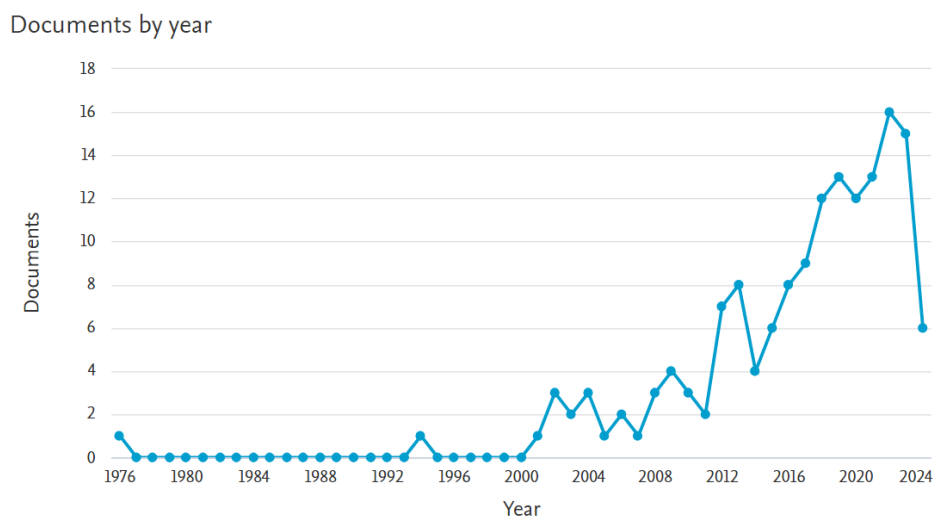


Figura 4.1: Documentos publicados anualmente relacionados con estimación del salario deportivo. Búsqueda realizada en *Scopus* con la *query*: TITLE-ABS-KEY ( ( ( wage OR salary OR ( market AND value ) ) AND ( estimation OR prediction ) AND ( sport OR football OR soccer ) ) ). Fecha: 02/05/2024.

muy básico que usa la media relativa de cada equipo. Este estimador ya consigue mejorar significativamente las predicciones de reporteros y expertos de la época, estimaciones por otra parte totalmente subjetivas. Dando a entender que existe cierta relación entre el salario de un jugador y el gasto en salarios del equipo.

En [16] se estudia la relación entre las ganancias de los equipos y los salarios de los futbolistas. Llegando a la conclusión de que los beneficios crecían a mayor ritmo que los salarios en el periodo de 2001 a 2009, suponiendo esto que el salario no tiene por qué estar ligado a los beneficios obtenidos por el club.

Por otro lado un estudio sobre la dispersión de salarios y como afecta a las actuaciones de un equipo, [18], utilizando datos sobre la liga italiana de fútbol. Este trabajo obtiene unos resultados que muestran que la consideración de que el equipo consiste únicamente en los miembros que contribuyen directamente a la alta dispersión salarial, tiene un impacto perjudicial en el rendimiento del equipo. Ampliar la definición del equipo hace que este efecto desaparezca o incluso cambie de dirección.

Vemos al final, que todos estos estudios buscan relacionar algún tipo de información sobre el futbolista, o su equipo, con el salario, al igual que haría un modelo de *Machine Learning*. El problema es que al usar métodos

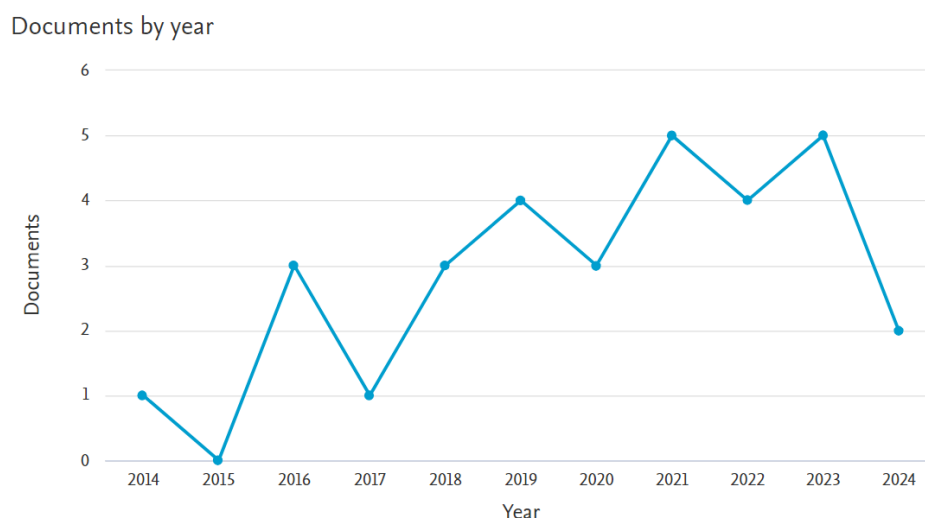


Figura 4.2: Documentos publicados anualmente relacionados con estimación del salario deportivo y el uso de IA. Búsqueda realizada en *Scopus* con la *query*: TITLE-ABS-KEY ( ( ( ( deep AND learning ) OR ( machine AND learning ) OR ( soft AND computing ) OR ( artificial AND intelligence ) OR ( data AND mining ) ) AND ( wage OR salary OR ( market AND value ) ) AND ( estimation OR prediction ) AND ( sport OR football OR soccer ) ) ). Fecha: 02/05/2024.

más rudimentarios, los investigadores no son capaces de manejar grandes cantidades de datos y estadísticas, que un regresor de ML si es capaz.

#### 4.1.2. Métodos relacionados con la Inteligencia Artificial

En 2016 empiezan a aparecer los primeros *papers* que ya utilizan un gran número de datos y estadísticas para estimar el salario [88]. Pero no es hasta el año siguiente donde aparece el primer trabajo prometedor, [100].

En este estudio, se propone un método objetivo y cuantitativo para determinar los salarios de los futbolistas basándose en sus estadísticas, las cuales son obtenidas de distintos videojuegos de fútbol. El método se basa en la aplicación de algoritmos de reconocimiento de patrones a datos de rendimiento (por ejemplo, goles), comportamiento (por ejemplo, agresión) y habilidades (por ejemplo, aceleración) de los futbolistas. Pero este no tiene en cuenta aspectos que no están directamente relacionados con el juego, como la popularidad del jugador entre los aficionados, las ventas de *merchandise* previstas, etc. Los resultados experimentales, utilizando datos de 6082 jugadores, muestran que la correlación de *Pearson* entre el salario predicho

y el salario real de los jugadores es de aproximadamente  $0.77(p < .001)$ , para el algoritmo de *Nearest Neighbor*.

Este estudio obtiene grandes avances en la comprensión y estimación del salario. Aunque el uso de habilidades y comportamientos como características supone la introducción de subjetividad en las estimaciones del modelo. Ya que dichos datos son creados por los desarrolladores del videojuego del que se han obtenido las estadísticas, FIFA en este caso.

Otro estudio similar es [55], donde en lugar de utilizar estadísticas de videojuegos, se usan estadísticas reales de distintas *webs*. Este artículo utiliza también distintos medidores para calcular los logros conseguidos por los jugadores en la temporada, así como su rendimiento en las distintas competiciones. Consiguiendo un coeficiente de determinación ( $R^2$ ) final de 0.606. El principal problema de este estudio, es que solo utiliza 24 características, es decir, obvia muchas otras que no considera relevantes.

El paper [84] si que tiene cuidado de realizar una selección objetiva de características, seleccionando inicialmente un gran número de estadísticas distintas. Y posteriormente reduciendo dicho número, tras observar las relaciones entre las características y la variable a predecir, que en este estudio en vez de ser el salario del futbolista, es su valor de mercado. Nuevamente, eso sí, se utilizan estadísticas del videojuego FIFA como algunas de las características, pero no todas como en estudios anteriores [100]. Otra novedad es el uso de parámetros que miden la popularidad del futbolista, como el número de visitas en *Google* o de *clicks* en *Wikipedia*. En cuanto a los resultados, se obtiene un coeficiente de determinación de 0.85 para *Random Forest*, suponiendo esto un gran hito en la estimación del valor de mercado de un futbolista.

Otro estudio de estimación de valor de mercado es [39]. Este es uno de los primeros estudios que utiliza ya estadísticas más refinadas como los  $xG^2$  o las  $xA^3$ , las cuales ya de por sí se obtienen mediante técnicas de IA [79]. Así mismo, también es uno de los primeros estudios que utiliza redes neuronales para la resolución del problema, un Perceptrón Multicapa en este caso. La principal pega de este estudio es que solo utiliza datos de centrocampistas de la liga turca, quedándose con un *dataset* final de solo 236 jugadores. Los resultados son muy buenos, 0.999 de coeficiente de determinación usando 9, 10 u 11 capas ocultas. Sin embargo, tengo mis dudas en cuanto a la capacidad de generalización del modelo para datos de otras ligas o posiciones.

El presentado en [13] es otro estudio interesante, cuyo método propuesto tiene dos fases. En la primera etapa el conjunto de datos se agrupa utilizando un método de agrupamiento automático llamado *APSO-clustering*. Este

---

<sup>2</sup>Goles esperados.

<sup>3</sup>Asistencias esperadas.

método de *clustering* divide el *dataset* en 4 grupos que indican la posición de los jugadores: porteros, centrocampistas, defensas y delanteros. En la segunda fase se utiliza un modelo de regresión híbrido, que es una combinación de la heurística *particle swarm optimization* (PSO) y el modelo de ML *support vector regression* (SVR). En este método híbrido, PSO se utiliza para la selección de características y el ajuste de parámetros de SVR. Los resultados obtenidos muestran que el método propuesto puede estimar el valor de los jugadores con un coeficiente de determinación de 0.74.

Investigaciones ya más recientes como [67], donde se busca estimar salarios, pero de la NBA<sup>4</sup>, utiliza estadísticas del videojuego *NBA 2K* y obtiene los salarios de una web similar a la que se usa en este artículo. Para los salarios, al ser valores tan elevados, los escala logarítmicamente. Posteriormente entrena, obteniendo un MSE de 0.361 usando *Random Forest*. Este es ya un trabajo muy similar a los anteriores, en cuanto a técnicas utilizadas, solo que aplicado a otro deporte.

Volviendo al fútbol, otro estudio reciente, [9], separa el problema en tres pasos. Un primer paso, donde determina que factores afectan al valor de mercado de los jugadores y los organiza en variables dependientes e independientes. En este paso simplemente se seleccionan las variables más frecuentes en otros estudios, que de nuevo se obtienen del videojuego FIFA, finalmente se queda solo con 7 características. En el siguiente paso, para las variables influyentes se hace un análisis preliminar para comprobar la calidad de las mismas. Y posteriormente otro más exhaustivo, donde se verifica la elección lógica de las características. Estos análisis son, primero el coeficiente de correlación de *Pearson* y luego un análisis de varianza (ANOVA). Por último se usan estos datos para entrenar modelos de ML. El estudio consigue, con *Random Forest*, un coeficiente de determinación de 0.95, este buen desempeño se debe en gran medida a que el problema cuenta con solo 7 características, por 17980 muestras.

Vemos que por lo general no hay muchos estudios sobre la estimación del salario de futbolistas. Existen, eso sí, trabajos similares como estimadores del valor de mercado de futbolistas, o estimadores del salario en otros deportes. Como son investigaciones similares, tendré todos los trabajos en cuenta. Aunque en especial, mi TFG se basa en el siguiente artículo, [13], por el hecho de utilizar primero un método heurístico para seleccionar las características y el *fine tuning* del modelo y posteriormente entrenar el estimador. Por otro lado, para la parte de los datos me baso sobretudo en [55], intentando extraer y obtener información que muestre el rendimiento del jugador la pasada temporada.

---

<sup>4</sup>La *National Basketball Association* o NBA por sus siglas en inglés es la liga de baloncesto de mayor renombre de Estados Unidos.

## 4.2. Explicabilidad de las predicciones

Aparte de obtener predicciones, es importante saber por qué el estimador llega a las conclusiones que llega [8]. Por ello también es interesante realizar un análisis del estado del arte en este campo, para así ver que técnicas podemos usar en este *paper* y dotar de explicabilidad a nuestras predicciones. Este es un campo mucho más popular, con 11787 documentos y donde cada año salen más investigaciones relacionadas, como podemos observar en la figura 4.3.

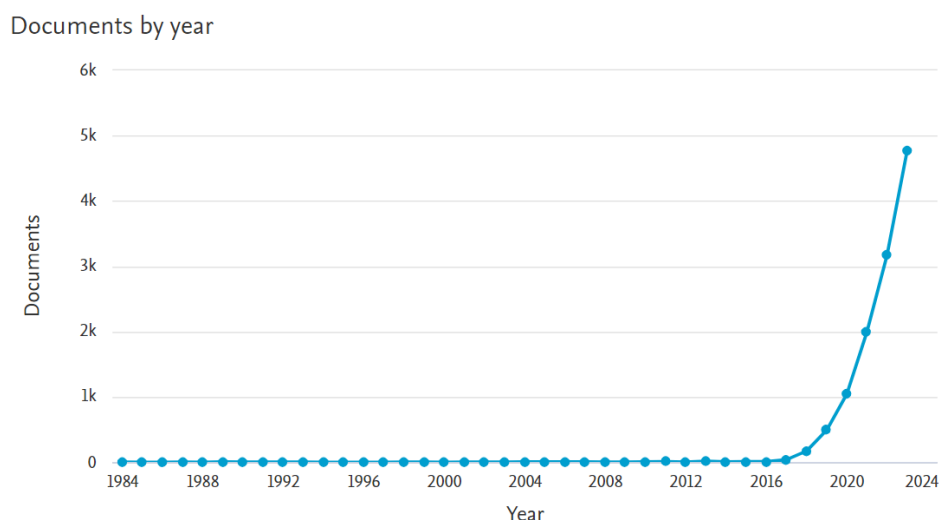


Figura 4.3: Documentos publicados anualmente relacionados con la explicabilidad de las predicciones. Búsqueda realizada en *Scopus* con la *query*: TITLE-ABS-KEY ( ( explainable AND ( ( deep AND learning ) OR ( machine AND learning ) OR ( soft AND computing ) OR ( artificial AND intelligence ) OR ( data AND mining ) ) ) ). Rango de la búsqueda: Hasta 2023. Fecha: 05/05/2024.

Los conceptos de interpretabilidad y explicabilidad son difíciles de definir rigurosamente. Sin embargo, se han realizado múltiples investigaciones para tratar de definirlos, siendo los trabajos más emblemáticos los de [57, 24].

El trabajo [33] constituye un intento de definir los conceptos clave en torno a la interpretabilidad en el aprendizaje automático. Los autores, centrándose principalmente en el aprendizaje profundo, también propusieron una taxonomía mediante la cual los métodos de interpretabilidad para redes neuronales podrían clasificarse en tres categorías diferentes. La primera abarca métodos que emulan el procesamiento de datos para crear percepciones sobre las conexiones entre las entradas y salidas del modelo. La segunda categoría



contiene enfoques que intentan explicar la representación de datos dentro de una red. La última categoría consiste en redes transparentes que se explican a sí mismas.

En [4] se realiza una extensa revisión de la literatura, recopilando y analizando 381 trabajos científicos diferentes entre 2004 y 2018. Los autores organizaron todo el trabajo científico en el campo de la IA explicativa a lo largo de cuatro ejes principales y subrayaron la necesidad de introducir más formalismo en el campo de la XAI<sup>5</sup> y de una mayor interacción entre humanos y máquinas. Después de destacar la tendencia de la comunidad a explorar la explicabilidad solo en términos de modelado, propusieron abrazar la explicabilidad en otros aspectos del aprendizaje automático. Finalmente, sugirieron una dirección de investigación potencial que sería hacia la composición de métodos de explicabilidad existentes.

Otra investigación que intentó categorizar los métodos de explicabilidad existentes fue [37]. En primer lugar, los autores identificaron cuatro categorías para cada método basadas en el tipo de problema para el que fueron creados. Una categoría para explicar modelos de caja negra, una para inspeccionarlos, una para explicar sus resultados y una para crear modelos de caja negra transparentes. Posteriormente, propusieron una taxonomía que tiene en cuenta el tipo de modelo de explicación subyacente (explicador), el tipo de datos utilizados como entrada, el problema que el método encuentra, así como el modelo de caja negra que fue 'abierto'. Como en trabajos discutidos anteriormente, se destacó una vez más la falta de formalismo y la necesidad de una definición de métricas para evaluar el rendimiento de los métodos de interpretabilidad. Mientras que también se planteó la incapacidad de la mayoría de los métodos de explicabilidad de caja negra para interpretar modelos que toman decisiones basadas en características desconocidas o latentes. Por último, se identificó la falta de técnicas de interpretabilidad en el campo de los sistemas de recomendación y se propuso un enfoque según el cual los modelos podrían aprender directamente de las explicaciones.

Un estudio más reciente, [8], desarrolla una taxonomía alternativa específicamente para los métodos de interpretabilidad de aprendizaje profundo. Bajo esta taxonomía se propusieron cuatro categorías: una para proporcionar explicaciones sobre el procesamiento de redes profundas, otra en relación con la explicación de la representación de redes profundas, una preocupada por la explicación de sistemas de producción y una que abarca híbridos de métodos transparentes y de caja negra. Por último, los autores profundizaron en el concepto de Inteligencia Artificial Responsable, una metodología que introduce una serie de criterios para implementar la IA en las organizaciones.

---

<sup>5</sup>Inteligencia Artificial Explicable por sus siglas en inglés.



## Capítulo 5

# Materiales y métodos

### 5.1. Conjunto de datos

Nuestro *dataset* está formado por 10786 ejemplos de futbolistas. Este cuenta con 71 características distintas para la temporada  $x$ , más la variable a predecir, es decir, el salario semanal en euros del jugador para la temporada  $x+1$ . Dicho conjunto de datos no existía previamente en ninguna base de datos y ha sido creado por mí mismo, a través de información de tres *webs* distintas. Estas han sido, *fbref.com* para obtener las estadísticas generales, *www.capology.com* para obtener los salarios, y *www.transfermarkt.es* para obtener valores de mercado de equipos y ligas. Los datos pertenecen a jugadores de las 5 grandes ligas, que son las ligas de Inglaterra, España, Italia, Alemania y Francia. El *dataset* contiene datos de las temporadas 2017-2018 a la 2022-2023, ambas inclusives.

#### 5.1.1. Extracción de los datos

Al estar los datos dispersos en distintas tablas y *webs*, es necesario extraerlos primero, para poder construir el *dataset* inicial. Para ello utilizo una técnica llamada *web scraping* [44], que hace referencia al proceso de automatizar la extracción de datos de páginas *webs*, utilizando algún *software*.

Nuestro *web scraper* se divide en dos partes. La primera, en la cual se obtiene el código HTML de la página. Y la segunda, en donde se detecta cual es la tabla con los datos y se transforma en un *dataframe*<sup>1</sup>.

Para el primer paso, utilizamos este *script*:

---

<sup>1</sup>Nombre con el que se le conocen a los objetos de la librería de *Python*, *Pandas*. Que sirven para representar tablas.

```

1 url_start = 'https://fbref.com/es/comps/Big5/{-}/{-}/jugadores
  /Estadisticas-{--}-Las-5-grandes-ligas-europeas'
2 dir_names = ['stats', 'keepers', 'misc', 'passing', 'possession',
  ', 'shooting']
3
4 driver = webdriver.Chrome()
5
6 for dir in dir_names:
7     for year in years:
8         url = url_start.format(year, year+1, dir, year, year+1)
9
10        driver.get(url)
11        driver.execute_script("window.scrollTo(1, 10000)")
12        time.sleep(2)
13
14        html_code = driver.page_source
15
16        with open(("html/"+dir+"/{-}/{-}.html").format(year, year
17            +1), "w+", encoding='utf-8') as f:
18            f.write(html_code)

```

Listing 5.1: *Script* para extraer códigos fuente de páginas *web*

Lo que hace este código es, dada una plantilla de URL (en el ejemplo 5.1 se usa la de la web *fbref*), se crea la URL real para la temporada *x*, se accede a dicha URL con un emulador (el objeto *driver*), se carga toda la página haciendo *scroll* y esperando un poco, y por último se obtiene el código fuente de la *web*. Finalmente guardamos ese fuente en un archivo HTML, dentro del directorio con el mismo nombre. Hacemos esto porque necesitaremos estos archivos a continuación para extraer los datos como tal.

En el segundo paso, de nuevo usamos un *script* para extraer los datos de los ficheros obtenidos en el primer paso. El *script* utilizado es el siguiente:

```

1 id = ['stats_standard', 'stats_keeper', 'stats_misc', '
  stats_passing', 'stats_possession', 'stats_shooting']
2
3 for i in range(len(id)):
4     dfs = []
5
6     for year in years:
7         with open(("html/"+dir_names[i]+"/{-}/{-}.html").format(
8             year, year+1), encoding='utf-8') as f:
9             page = f.read()
10
11            soup = BeautifulSoup(page, "html.parser")
12
13            for header_row in soup.find_all('tr', class_="
14                over_header"):
15                header_row.decompose()

```

```
15     for header_row in soup.find_all('tr', class_="thead"):  
16         header_row.decompose()  
17  
18         stats_table = soup.find(id=id[i])  
19         stats_table = pd.read_html(str(stats_table))[0]  
20         stats_table["Temporada"] = year  
21  
22         dfs.append(stats_table)  
23  
24     stats = pd.concat(dfs)  
25     stats.to_csv("csv/"+id[i]+".csv", index=False)
```

Listing 5.2: *Script* para extraer una tabla de un archivo HTML

Para comprender como funciona este código, hay que hacer una breve explicación sobre aspectos básicos de HTML. Los elementos de una página web, como pueden ser una imagen, una tabla, o un párrafo, están organizados en clases. Por lo que dentro del código fuente, al encapsular un elemento, se le asigna al menos una clase. Por otro lado, elementos especiales, como las tablas, pueden tener un identificador único, para referenciales.

Ahora sí, la tarea que realiza el *script* 5.2 es, primero leer un *HTML*. Una vez se tiene el fuente, elimina los elementos de las clases *over\_header* y *thead*, las cuales encapsulan subcabeceras de tablas, que no nos interesan extraer. Por último, usando el identificador de la tabla que queremos extraer, buscamos y guardamos en un fichero dicha tabla transformada en un *dataframe*. Es importante mencionar, que a cada tabla se le añade una columna extra, con el año que le corresponde. Hago esto para luego no tener problemas a la hora de unir las distintas tablas. Finalmente unimos todas las tablas de un mismo tipo en una sola.

Este proceso se puede unificar en un único *script*. No obstante, debido al elevado coste de tiempo que consume extraer los códigos fuente de las distintas *webs*, es muy recomendable ir guardando los ficheros HTML según se van obteniendo. Esto, para dado el caso que se pierda el *dataset*, solo haya que extraer los datos de los fuentes, en lugar de, tener de nuevo que volver a *scrapear* las distintas *webs*. Es por ello que decido dividir la tarea en dos. Tras ejecutar estos dos *scripts*, ya podemos pasar al siguiente paso, preprocesar los datos.

### 5.1.2. Preprocesamiento de los datos

Con los datos extraídos en bruto, aún es necesario realizarles una profunda limpieza. En un principio tenemos las distintas tablas con estadísticas por separado. Hacemos una primera reducción de características, donde eliminamos la nacionalidad del jugador, ya que no aportan ninguna información, así como otras variables que son el resultado de realizar una operación con

otras variables, que tampoco aportan más información extra de la que dan sus características padre. Por ejemplo la estadística 'goles por partido' no es más que dividir goles/partidos. El nombre del jugador, en cambio, no lo eliminamos, ya que nos servirá de clave primaria para diferenciar las distintas entradas de la tabla.

El siguiente paso es eliminar los jugadores duplicados. Estos son jugadores que en una misma temporada han jugado en varios equipos, y por tanto tienen varias entradas para la misma tabla. Para detectar cuales son estos jugadores, utilizamos tres características como clave primaria: Jugador, Edad y Temporada. Estas y el resto de estadísticas vienen explicadas en el apéndice A. Al juntar las entradas, para algunas características nos quedamos con la suma de las distintas entradas. Para otras que no pueden o tienen sentido ser sumadas, como la posición o el salario, nos quedamos con la característica de la primera entrada. Podemos observar como se modifica la tabla de estadísticas generales, teniendo en la figura 5.1 varias entradas para un mismo año, y colapsándose dichas entradas repetidas en la figura 5.2, tras realizar este paso.

Jugador	Posc	Edad	PJ	Mín	Temporada
João Cancelo	DF	23.0	26	1851	2017
João Cancelo	CC	23.0	1	90	2017
João Cancelo	DF	24.0	25	1968	2018
João Cancelo	DF	25.0	17	1208	2019
João Cancelo	DF,CC	26.0	28	2299	2020
João Cancelo	DF	27.0	36	3227	2021
João Cancelo	DF	28.0	17	1274	2022
João Cancelo	DF,DL	28.0	15	1012	2022

Figura 5.1: Ejemplo de entradas disponibles en la tabla de estadísticas generales para 'Jugador' == 'João Cancelo', tras el primer paso del preprocesamiento.

Jugador	Posc	Edad	PJ	Mín	Temporada
João Cancelo	DF	23.0	27	1941	2017
João Cancelo	DF	24.0	25	1968	2018
João Cancelo	DF	25.0	17	1208	2019
João Cancelo	DF,CC	26.0	28	2299	2020
João Cancelo	DF	27.0	36	3227	2021
João Cancelo	DF	28.0	32	2286	2022

Figura 5.2: Ejemplo de entradas disponibles en la tabla de estadísticas generales para 'Jugador' == 'João Cancelo', tras el segundo paso del preprocesamiento.

Luego unimos las tablas que contienen información sobre distintas es-

tadísticas en una verticalmente. Esto lo conseguimos uniendo las entradas con la misma clave primaria, que tras realizar el segundo paso son únicas, por lo que aseguramos que no haya conflictos al juntar las características en una tabla. Para la tabla que contiene el salario, al ser estos de la temporada siguiente, hay que restar uno a la edad para que la clave primaria: Jugador, Edad y Temporada, cuadre con el resto de tablas. De nuevo, por la peculiaridad de que los salarios no son de la misma temporada que el resto de características, hay algunos jugadores que en la temporada  $x$  jugaban en las cinco grandes ligas, y en la temporada  $x + 1$  ya no, y viceversa. Por lo que existen casos de entradas con columnas sin valores, simplemente eliminamos estas entradas del *dataset*. Por ejemplo, al unir las claves primarias del jugador 'Aaron Connolly', podemos ver en la figura 5.3 como en las temporadas 2021 y 2018 hay datos faltantes. El de la temporada 2021 se debe a que el sueldo es de la temporada 2022, temporada en la que Aaron Connolly juega en el Venecia, fuera de las cinco grandes ligas. En el caso de la temporada 2018, el salario es de la temporada 2019 donde Connolly juega en la *Premier League*, la primera liga de Inglaterra. Sin embargo las estadísticas, que son de 2018, son de cuando jugaba en categorías inferiores aún.

Jugador	Edad	Temporada	Equipo_x	Salario semanal
Aaron Connolly	19	2019	Brighton	£ 23,077 (€ 27,521, \$28,044)
Aaron Connolly	20	2020	Brighton	£ 23,077 (€ 27,521, \$28,044)
Aaron Connolly	21	2021	Brighton	NaN
Aaron Connolly	18	2018	NaN	£ 7,500 (€ 8,944, \$9,114)

Figura 5.3: Ejemplo de entradas disponibles en *dataset* para 'Jugador' == 'Aaron Connolly', tras el tercer paso del preprocesamiento.

El siguiente paso, es eliminar caracteres no deseados de las variables, por ejemplo como se puede ver en la figura 5.3, el salario viene dado en libras, euros y dólares. Lo que hacemos es eliminar todo excepto la cifra en euros. Otras columnas como 'Comp\_x' y 'Comp\_y' también reciben este tratamiento. Posteriormente añadimos la información del equipo al *dataset*, esto es cambiar el nombre del equipo por su posición en liga, dependiendo del nombre del equipo y de la temporada.

A continuación modificamos los salarios para ajustarlos a la tasa de inflación actual. Este es un punto importante para que el modelo aprenda correctamente, de otra forma el modelo estimaría salarios por debajo del valor real, ya que en años anteriores la inflación era menor. Resumiendo, lo que buscamos es que los predictores estimen el salario que debería cobrar un futbolista en 2023. Más precisamente, calculamos la inflación ajustada al 1 de junio de 2023, ya que el periodo de fichajes se abre esa fecha. Para calcular las tasas de inflación utilizamos la calculadora de [32], donde seleccionamos

utilizar el índice *EUCPI2005*<sup>2</sup> a la hora de realizar las conversiones de salarios. Utilizamos este índice ya que a excepción de la *Premier League*, el resto de ligas pertenecen a la Unión Europea y utilizan el euro como moneda de curso legal.

Tras esto, para las ligas añadimos información del beneficio obtenido en la temporada correspondiente. Lo hacemos de forma similar a como hemos sustituido los nombres de equipos por sus posiciones en liga. Estos beneficios los obtenemos de [89], que a su vez consigue la información del siguiente artículo: [22].

Luego sustituimos los nombres de los equipos por los valores de mercado en la temporada correspondiente de los mismos. Por último cambiamos algunos nombres de columnas, comprobamos que no existan valores nulos y guardamos la tabla resultante en un fichero CSV<sup>3</sup>.

Una vez tenemos el *dataset* preprocesado. Eliminamos las variables 'Jugador' y 'Temporada', que no van a aportar ninguna información a los modelos de ML y solo las necesitábamos para unificar las distintas tablas de estadísticas. Posteriormente, codificamos las variables categóricas. En nuestro *dataset* la única variable categórica es 'Posicion', que puede tener hasta 10 valores distintos, ya que algunos jugadores pueden jugar en varias posiciones. Decido crear cuatro nuevas variables booleanas, una por posición única. Y para cada jugador indicar si juega en dichas posiciones. Así para los jugadores que juegan en varias posiciones no tengo que codificar cada par de posiciones, evitando que la dimensionalidad del problema aumente mucho más.

### 5.1.3. Visualización y análisis preliminar del conjunto de datos

Ya con el *dataset* procesado, podemos analizar el mismo. Lo primero que hago es separar el conjunto de datos en entrenamiento y test. Como tenemos un *dataset* de tamaño medio, selecciono el 15 % de la muestra para test, para así tener una buena cantidad de datos para entrenar nuestros modelos y bastantes datos para luego comprobar como funcionaría el modelo elegido para ejemplos de fuera de la muestra [74]. Quedándonos así con 9168 instancias en el conjunto de entrenamiento y 1618 para test. A partir de este momento, cada vez que nos refiramos al *dataset* en realidad nos referiremos al conjunto de entrenamiento. Es muy importante no usar el conjunto de test para nada, para evitar que los resultados del estudio estén sesgados. Ya que si tomamos alguna decisión en base a test, posteriormente cuando obtengamos las predicciones de este conjunto, no sabremos realmente como

<sup>2</sup>*European Union Consumer Price Index.*

<sup>3</sup>Valores Separados por Comas por sus siglas en inglés.



funcionaría el modelo elegido para ejemplos de fuera de la muestra. Esto debido a que hemos utilizado test como parte de la muestra en al menos un paso [98, 3].

Tras esto, es interesante observar un resumen de las variables para ver si existe alguna anomalía. Para las variables continuas nuestro: media, mediana, desviación típica, máximo, mínimo y valores faltantes. Para las categóricas nuestro: número de ejemplos, rango de valores y valores faltantes. Este resumen se puede ver en el apéndice B. Las principales conclusiones que saco del mismo son, primero que todos los valores están dentro de rangos normales. Y segundo que para algunas variables existen intervalos muy grandes, por lo que será interesante normalizar las mismas. Esto para facilitar el proceso de aprendizaje a los modelos. En cuanto a las variables categóricas, podemos ver su distribución en la figura 5.4. Observamos que los porteros son el grupo minoritario, en cuanto al resto de posiciones, estas están más o menos balanceadas.

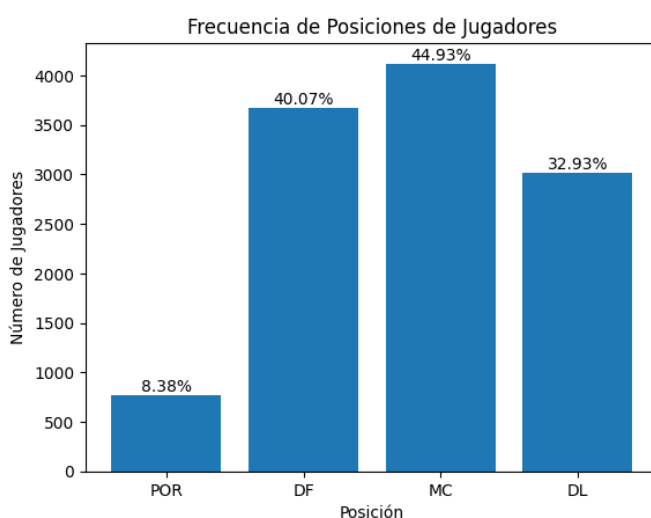


Figura 5.4: Histograma con la frecuencia de cada posición en el *dataset*.

Visualizar la variable a predecir también es un punto muy importante. En esta línea, en la figura 5.5 podemos ver la distribución de salarios. Se puede observar que la mayoría de sueldos no llegan a los 100000 euros semanales. Más exactamente, en la tabla 5.1 podemos observar que el 75 % de los futbolistas que menos cobran, ganan menos de 73426€, pero existen algunos pocos jugadores con salarios muy superiores a la media. En la figura 5.6 se pueden ver mejor esos futbolistas *outliers*. A la distribución de la figura 5.5 se la conoce como distribución potencial, ya que sigue la denominada "ley de la potencia".<sup>en</sup> la que un número significativo de individuos presentan los menores valores de la variable y un número muy limitado los valores más

altos, además de darse la circunstancia de que la media no es representativa de la distribución. Los *datasets* con este tipo de distribución dan muchos problemas a la hora de entrenar [20, 35]. Por lo que escalo logarítmicamente la variable para conseguir una distribución normal, las cuales dan mejores resultados en problemas de regresión [48]. En la figura 5.7 se puede ver la nueva distribución del salario tras ser este escalado.

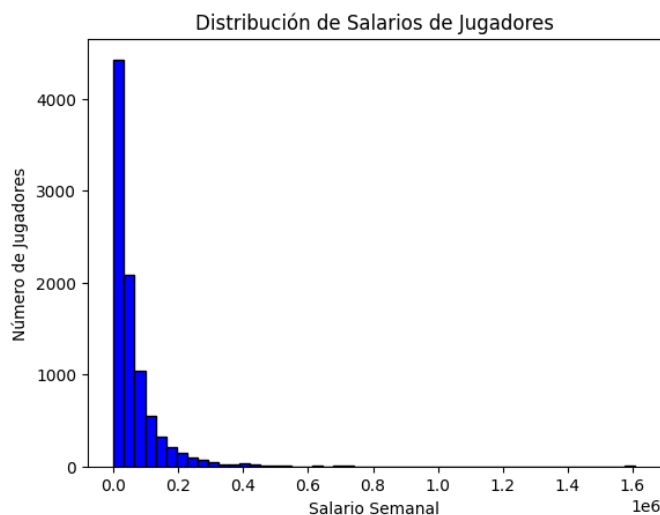


Figura 5.5: Distribución de los salarios de los futbolistas en 50 *bins*.

Cuadro 5.1: Distribución en cuartiles del salario.

Cuartil	Valor
Ejemplos	9168
Media	61919.33€
Desviación típica	91465.81€
Mínimo	296€
25 %	15003€
50 %	33957€
75 %	73426€
Máximo	1609609€

Así como es importante analizar la distribución de la variable a predecir, también lo es analizar el resto de características del problema. El problema es que al tener tantas características, la dimensión a la hora de visualizar la distribución es también muy elevada. Por lo que hay que utilizar alguna técnica para reducir esta dimensión. El método seleccionado es *t-distributed Stochastic Neighbor Embedding* (t-SNE) [94]. Esta técnica convierte las afinidades de los puntos de datos en probabilidades. Las afinidades en el espacio

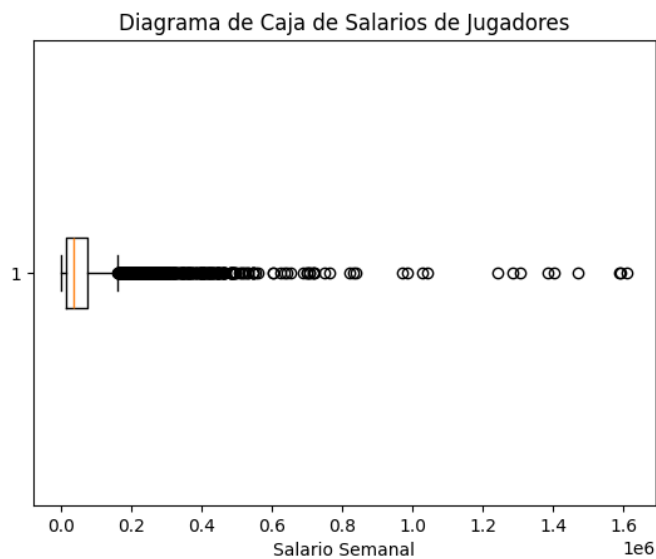


Figura 5.6: Diagrama de caja con los salarios de los futbolistas.

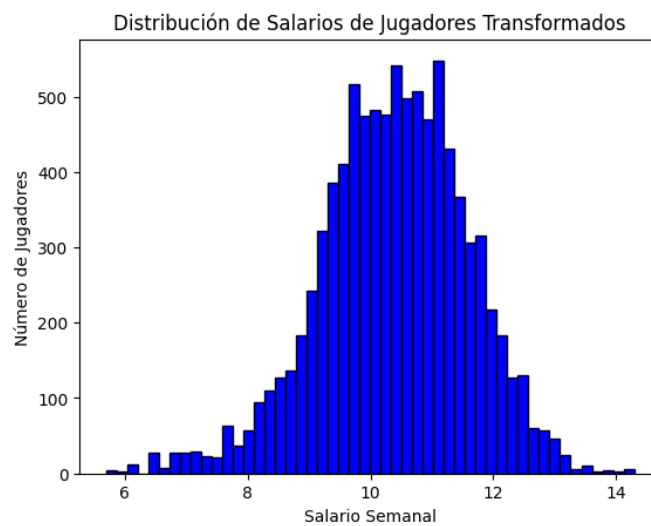


Figura 5.7: Distribución de los salarios de los futbolistas tras escalarlos logarítmicamente.

original están representadas por probabilidades conjuntas *gaussianas* y las afinidades en el espacio incrustado están representadas por distribuciones *t-Student*. En la figura 5.8 podemos observar como se distribuyen los jugadores dependiendo de su posición, se puede apreciar como la mayoría de futbolistas pertenecen a un único *cluster*. No obstante, en el caso de los porteros, se forma otro *cluster* independiente. En la figura 5.9 se puede apreciar mejor esta peculiaridad. Tras varias pruebas consigo descubrir que los porteros anómalos son porteros que han jugado más minutos, como se puede ver en la figura 5.10. Mi hipótesis es que los porteros con más minutos, al tener sus propias estadísticas, forman su *cluster* independiente. Por otro lado, los porteros con menos minutos también tienen sus propias estadísticas, como porteros que son, pero no las han aumentado demasiado al haber jugado pocos minutos. Por lo que estas tendrán valores cercanos a cero, haciendo a estos porteros indistinguibles de otros jugadores que han jugado pocos minutos, que también tendrán valores cercanos a cero en sus estadísticas. Esta hipótesis la podemos dar como válida observando la figura 5.11. En la cual se observa como los jugadores con menos minutos para cualquier posición se distribuyen por la zona izquierda del espacio, formando todos un único *cluster*.

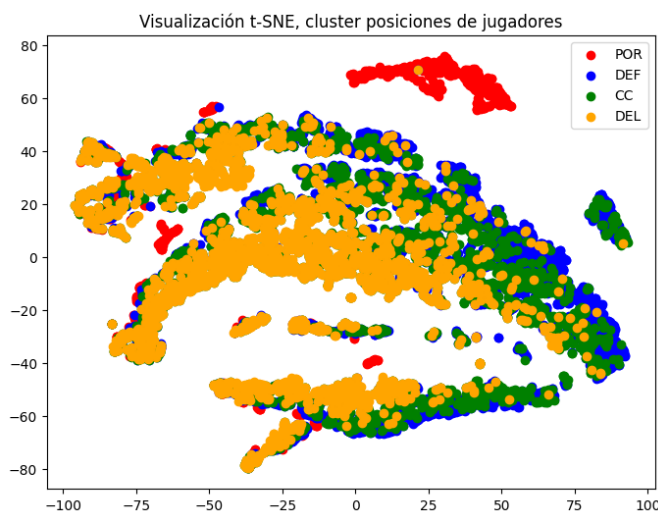


Figura 5.8: Distribución de los jugadores en base a la posición del futbolista.

Una vez visualizado el conjunto de datos, vamos a normalizar las variables continuas como comentábamos más arriba. La idea es dejar todas las variables en un rango entre 0 y 1, por lo que para cada valor  $x$  de una característica le restamos el valor mínimo de dicha característica y dividimos entre el máximo menos el mínimo, quedando la siguiente función:

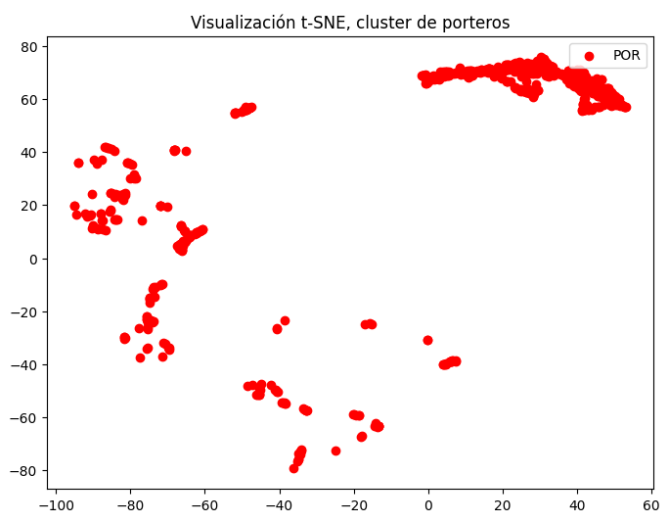


Figura 5.9: Distribución de los porteros.

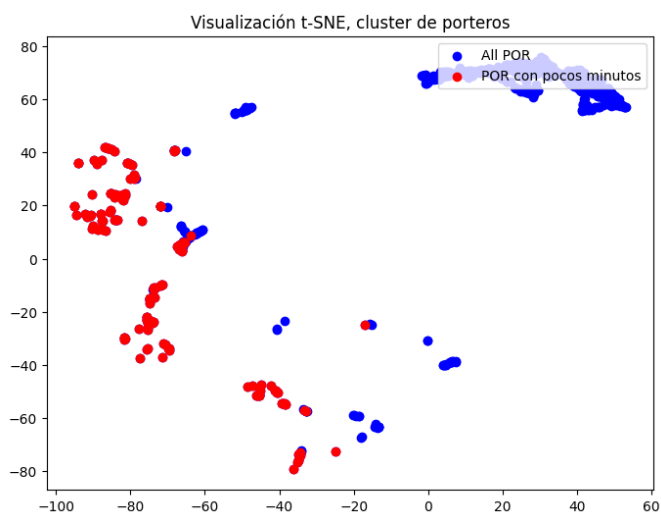


Figura 5.10: Distribución de los porteros en base a los minutos jugados.

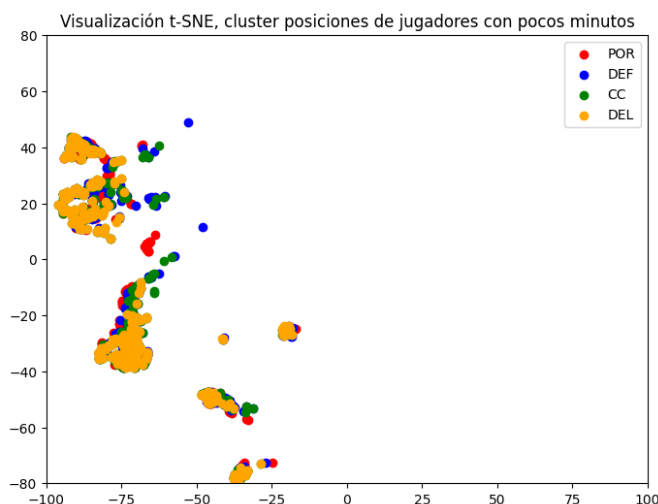


Figura 5.11: Distribución de los jugadores que han jugado menos minutos en base a la posición del futbolista.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (5.1)$$

Al normalizar utilizamos el conjunto de entrenamiento para guardar los valores máximo y mínimo de cada variable y luego normalizamos con esos valores tanto test como el propio conjunto de entrenamiento. La idea es que si realmente tenemos un *dataset* que represente correctamente a toda la población existente, al normalizar test con los valores de entrenamiento el rango en el que este quede sea cercano a  $[0 - 1]$ .

## 5.2. Selección de características

En un problema de *Machine Learning* es vital reducir la dimensionalidad del problema, siempre claro, intentando perder el mínimo de explicabilidad. Esto es tan importante ya que con una menor dimensionalidad los resultados suelen mejorar, al reducirse la complejidad del problema [95]. Existen diversas técnicas para conseguir este objetivo. En nuestro caso como queremos tener control de las características usaremos métodos de selección de características. No obstante existen otras técnicas, que por ejemplo implican transformaciones en las características, resultado de unir varias variables en una manteniendo así una mayor cantidad de explicabilidad.

### 5.2.1. Heurísticas propuestas

Algunos de los métodos utilizados consisten en calcular la correlación entre las características del problema, utilizando alguna métrica como las explicadas en los fundamentos teóricos. Otros métodos, sin embargo, utilizan procesos algorítmicos para seleccionar las  $k$  mejores características. En especial los métodos de tipo envolvente. Las técnicas utilizadas en nuestro estudio son las siguientes:

#### Eliminación Recursiva de Características

Este [19] es un método de tipo envolvente, ya que utiliza un modelo de ML para seleccionar las características. RFE, por sus siglas en inglés, elimina recursivamente características en cada paso y vuelve a realizar las predicciones al reentrenar el modelo basado en estas características restantes. En cada entrenamiento se evalúa la importancia de cada característica y se elimina la menos importante, siguiendo una heurística voraz (*greedy*). El proceso se puede resumir en el siguiente algoritmo:

---

**Algorithm 1:** Eliminación Recursiva de Características

---

**Data:**  $M, x, k$   
**Result:**  $x$   
**while**  $\text{len}(x) > k$  **do**  
     $M.\text{fit}(x)$   
     $\text{idx} = M.\text{CaracteristicaMenosImportante}()$   
     $x.\text{drop}(\text{idx})$   
**end**

---

Donde:

- $M$ : Es un modelo de *Machine Learning*.
- $x$ : Es el conjunto de características del problema.
- $k$ : Es el número de características final que se desea tener.

#### Método de Selección de Características Las Vegas, versión filtro

Los algoritmos Las Vegas toman decisiones probabilísticas para ayudar a guiarse más rápidamente hacia una solución correcta. Un tipo de algoritmos Las Vegas utiliza la aleatoriedad para dirigir su búsqueda de tal manera que se garantiza una solución correcta incluso si se toman decisiones erróneas [58].

El algoritmo LVF<sup>4</sup> genera un subconjunto aleatorio  $S$  de las  $N$  características totales en cada iteración. Si el número de características ( $C$ ) de  $S$  es menor que el mejor actual, es decir,  $C < C_{best}$ , se calcula para el *dataset* ( $D$ ), usando las características especificadas en  $S$ , un criterio de inconsistencia, que se explicará más adelante. Si su tasa de inconsistencia ( $i$ ) es menor o igual a la tasa actual ( $i_{best}$ ),  $C_{best}$ ,  $i_{best}$  y  $S_{best}$  se reemplazan por  $C$ ,  $i$  y  $S$  respectivamente. En caso de que  $C == C_{best}$ , se añade la solución actual a la lista de soluciones ( $L$ ). Cuando LVF llega a *MAX TRIES* iteraciones, se detiene. Todo este proceso algorítmico se puede ver de forma más esquemática en el algoritmo 2.

---

**Algorithm 2:** Las Vegas Filtro

---

**Data:** MAX TRIES,  $D$ ,  $N$   
**Result:**  $L$   
 $C_{best} = N$   
 $S_{best} = \text{Todas las características}$   
 $i_{best} = \text{InconCheck}(D, S_{best})$   
 $L = [S_{best}]$   
**for**  $i = 1$  **to** *MAX TRIES* **do**  
     $S = \text{randomSet}()$   
     $C = \text{numOffFeatures}(S)$   
    **if**  $C < C_{best}$  **then**  
         $i = \text{InconCheck}(D, S)$   
        **if**  $i \leq i_{best}$  **then**  
             $S_{best} = S$   
             $C_{best} = C$   
             $i_{best} = i$   
             $L = [S_{best}]$   
        **end**  
    **else**  
        **if**  $C == C_{best}$  **then**  
            **if**  $S$  *not in*  $L$  **then**  
                 $L.\text{append}(S)$   
            **end**  
        **end**  
    **end**  
**end**

---

El criterio de inconsistencia es la clave del éxito de LVF. La tasa de inconsistencia de un conjunto de datos se calcula de la siguiente manera: dos instancias se consideran inconsistentes si coinciden excepto por sus etiquetas de clase. Para todas las instancias coincidentes, sin considerar sus etiquetas

---

<sup>4</sup>La letra F hace referencia a que es la versión filtro.



de clase, el número de inconsistencias es el número de instancias menos el mayor número de instancias de etiquetas de clase. Por ejemplo, supongamos que hay  $n$  instancias coincidentes, entre ellas  $c_1$  instancias pertenecen a la etiqueta  $label_1$ ,  $c_2$  a la etiqueta  $label_2$ , y  $c_3$  a la etiqueta  $label_3$ , donde  $c_1 + c_2 + c_3 = n$ . Si  $c_3$  es el mayor de los tres, entonces el número de inconsistencias es  $n - c_3$ . La tasa de inconsistencia es la suma de todos los números de inconsistencias entre el número total de instancias. Como nuestro problema es de regresión, debemos discretizar la variable objetivo. Para ello creo 30 *clusters* utilizando k-medias, donde cada salario pertenece a una de las clases [27].

### Método de Selección de Características Las Vegas, versión envolvente

Esta versión en lugar de utilizar el criterio de inconsistencia como métrica, utiliza el coeficiente de determinación, obtenido fruto de entrenar un modelo de ML con el subconjunto de características obtenido. Por lo demás funciona igual que LVF, salvo que en vez de llamar a *InconCheck*, se llama a *entrenaSubset*. El cual, como he dicho, entrena el modelo que el usuario elija con el subconjunto seleccionado y devuelve su valor de  $R^2$ .

## 5.3. Protocolo de validación

Para evaluar los resultados obtenidos en los experimentos realizados, empleamos la conocida como validación cruzada. Este método divide el *dataset* en  $k$  partes iguales y realiza el mismo número de entrenamientos. En cada uno se utilizan  $k - 1$  partes como conjunto de entrenamiento y la  $k$ -ésima parte se usa para validar. En la figura 5.12 se puede ver un esquema de como funciona todo este proceso de validación. Con este método, en cada iteración del ciclo, se tienen datos de entrenamiento y validación distintos, lo que proporciona una evaluación diferente para todas las métricas. En cada iteración se obtiene un valor para cada métrica y al final del proceso se calcula la media de todos los valores obtenidos para estimar la calidad del modelo evaluado. Estando así los resultados obtenidos menos sesgados al conjunto de validación seleccionado, ya que estamos seleccionando  $k$  conjuntos.

Es necesario buscar un compromiso entre fiabilidad de las predicciones y tiempo de cómputo, ya que a mayor valor para  $k$  elijamos, más entrenamientos tendremos, y por tanto mayor tiempo de cómputo. Siguiendo esta idea, selecciono un valor para  $k$  de 5, es decir, como protocolo de validación selecciono *5-folds cross validation* [30]. Me quedo con este valor ya que tenemos un *dataset* de tamaño medio, y por ejemplo seleccionar *10-folds* elevaría demasiado los tiempos de entrenamiento.

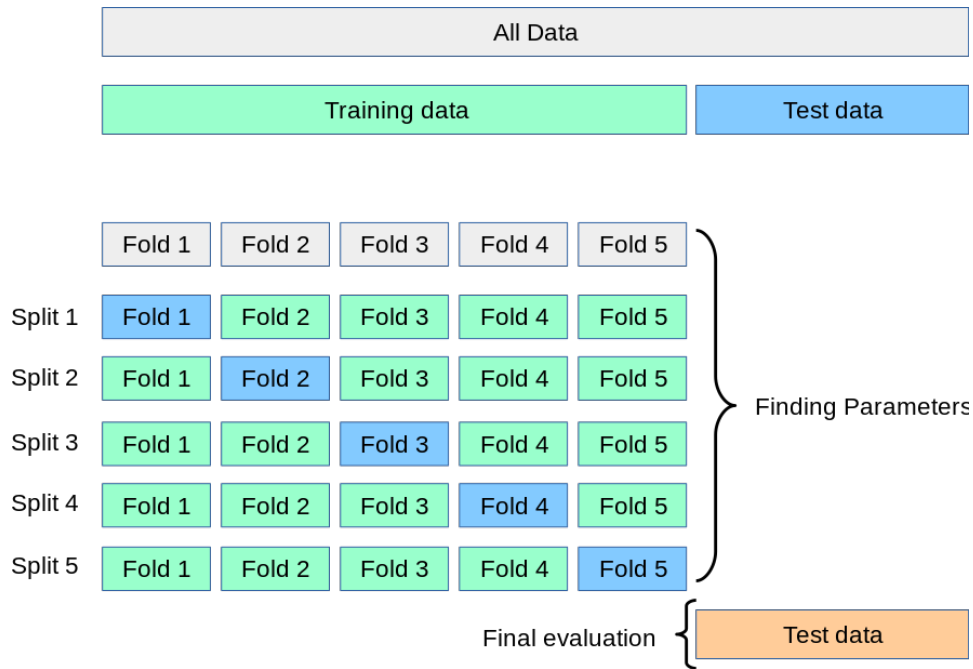


Figura 5.12: Esquema de *cross validation* con  $k = 5$ .

## 5.4. Aplicación *web*

En este apartado se explica todo el proceso de diseño y desarrollo de la aplicación *web* creada para este TFG, cuyo nombre es *FindTheSalary*. El objetivo principal de *FindTheSalary* desarrollada en este proyecto es permitir a cualquier usuario estimar el salario semanal que debería cobrar un futbolista a partir de sus estadísticas de juego. Utilizando para ello el mejor modelo del conjunto de hipótesis seleccionado en este estudio, así como las estadísticas más relevantes escogidas por el mejor selector de características. Además, se pretende que la *app* sea lo más intuitiva posible para el usuario, por lo que esta debe ser lo más simple posible.

### 5.4.1. Diseño

Para conseguir la simplicidad de la que hablamos, decido que la aplicación esté diseñada para un solo actor, el usuario. Para obtener la predicción, el usuario debe ser capaz de:

- Obtener los valores correspondientes a las características que la aplicación solicita.
- Introducir estos valores en la *app*.

- Ejecutar el modelo para obtener la estimación.

A partir de estos requisitos básicos puede obtenerse un diagrama de casos de uso que muestre la interacción del usuario con el sistema, como puede observarse en la figura 5.13. Por otro lado el diagrama de flujo se puede observar en la figura 5.14. La idea es que el usuario introduzca las estadísticas del futbolista que desee estimar el salario y si todo es correcto, se preprocesen dichos datos (por ejemplo, pasando las distancias introducidas en metros a yardas, que es la métrica usada al entrenar el modelo) y que el modelo devuelva la predicción obtenida. Para este momento, se han completado tanto el análisis de requisitos como el diseño. Dado que la aplicación es de baja complejidad, no se ha considerado necesario crear otros tipos de diagramas, como diagramas de clases o de secuencia. Por lo tanto, se procede a la fase de implementación.

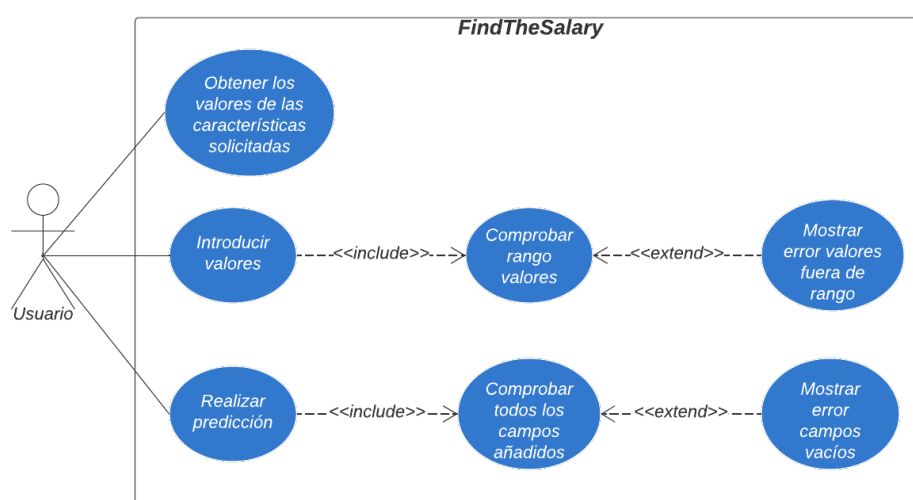


Figura 5.13: Diagrama de casos de uso de la aplicación *web*.

#### 5.4.2. Desarrollo

Para desarrollar la aplicación utilicé el *framework Streamlit*, por lo que no necesito centrarme en el desarrollo del *front-end*. Ya que utilizando la API de *Streamlit*, esta se encarga de hacer el *display* de las distintas estructuras de declare. De esta forma, solo me encargo de desarrollar el *back-end*. Para la estructura, llamo a un objeto *form* de la API, el cual servirá para almacenar las estadísticas del jugador que el usuario introduzca, en la figura 5.15 se puede ver como lo muestra *Streamlit* en la *app*. Todos los campos de este formulario cuentan con una leyenda de ayuda, que explica con exacti-

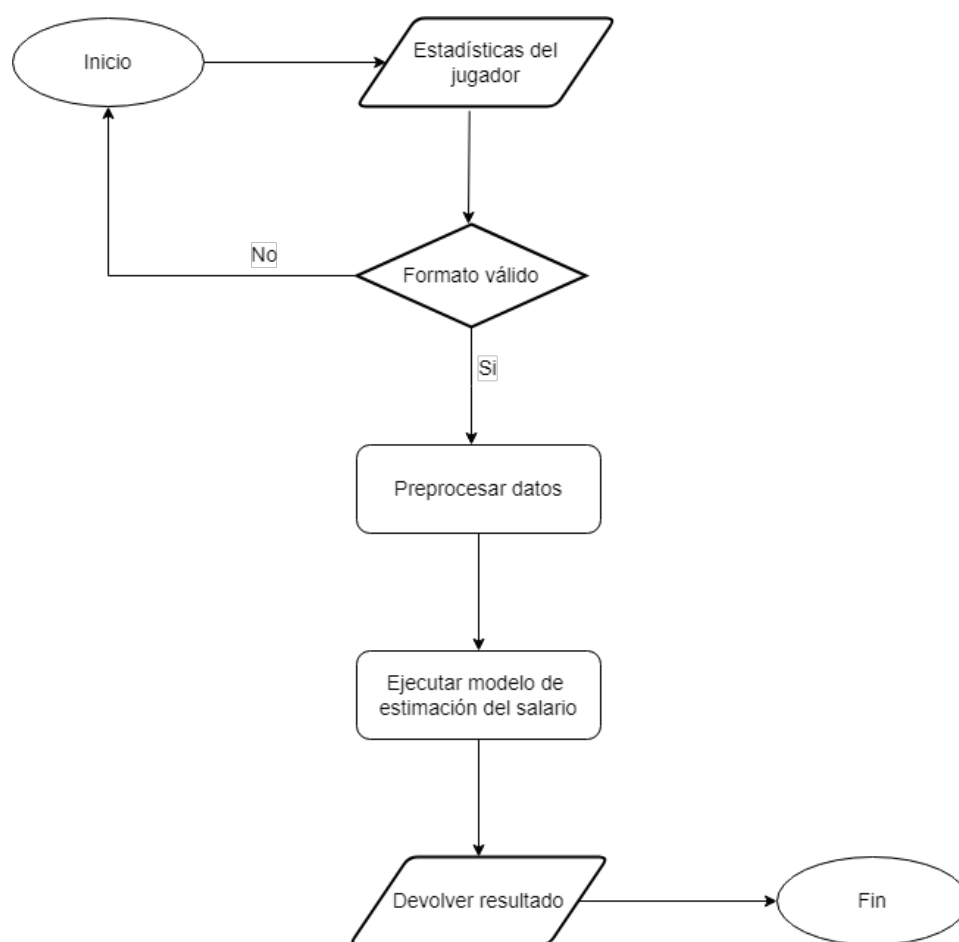


Figura 5.14: Diagrama de flujo de la aplicación *web*.

tud que introducir en dicho apartado, en la figura 5.16 se puede observar un ejemplo de dicho desplegable. Añado también un botón que debe presionar el usuario cuando haya introducido todas las características, para calcular la estimación. Cuando el botón se pulsa, se comprueba que todos los campos del *form* han sido rellenados correctamente. Posteriormente se pasan las estadísticas de metros a yardas, se normalizan todas las características utilizando el normalizador entrenado en este estudio y se ejecuta el mejor modelo desarrollado también en este proyecto para obtener el resultado final, en la figura 5.17 se puede observar como aparece impreso este resultado.

Para la parte de desplegar la aplicación no tengo que hacer nada, ya que *Streamlit* se encarga de desplegarla automáticamente. Existen dos opciones para realizar el *deploy*. Por un lado, puedes desplegar la *app* localmente solo ejecutando el *script* *webDeployment.py* que se encuentra en el mismo repositorio de *GitHub* que el resto del proyecto. Para ejecutarlo hay que introducir el comando: *streamlit run webDeployment.py*, una vez introducido el comando se mostrará en la consola el puerto en el que está siendo ejecutada la *app*. Por otro lado, puedes ejecutar la aplicación en la nube de *Streamlit* simplemente enlazando el repositorio de *GitHub* donde se encuentre el ejecutable con tu usuario de *Streamlit*. Este segundo método es el que uso y es por ello que puedo tener la aplicación corriendo en la siguiente dirección: <https://findthesalary.streamlit.app/>.

## FindTheSalary

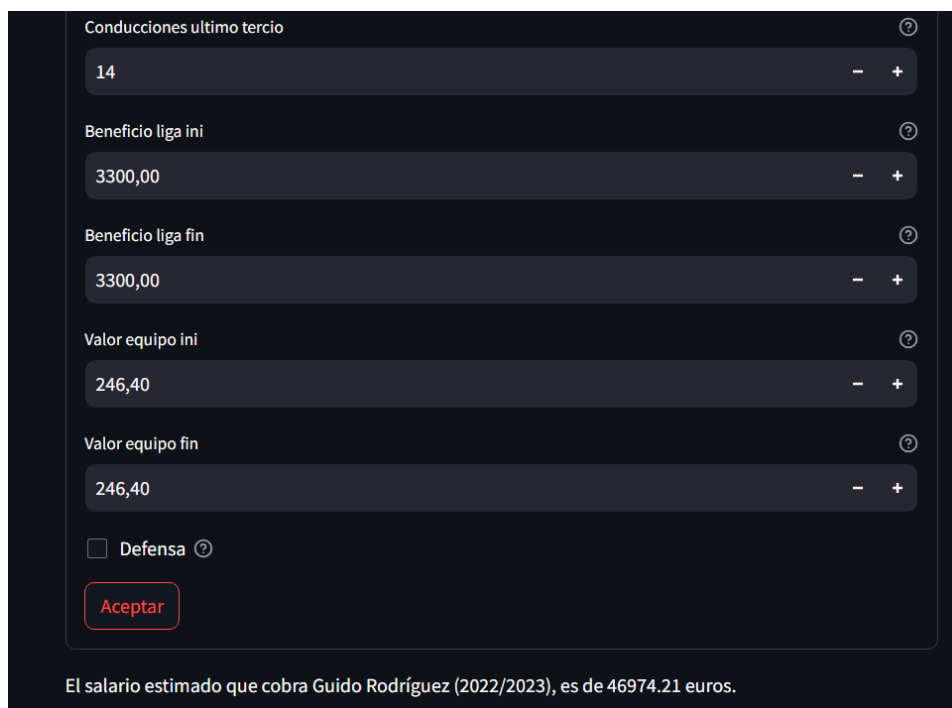
Esta aplicación estima el salario semanal de un futbolista basándose en distintas estadísticas. A continuación, actualice las estadísticas como desee y pulse aceptar para obtener el salario estimado.

Nombre	Guido Rodríguez (2022/2023)	?
Edad	28	- +
Titularidades	33	- +
Minutos	2872	- +
Penaltis lanzados	0	- +
xG	1,20	- +
Disparos	22	- +
Tiros a puerta	6	- +
Faltas lanzadas		?

Figura 5.15: Estructura principal de la aplicación *web*.

Dist con balon	3909,00	?
Conducciones que entran en el último tercio del campo		
Conducciones ultimo tercio	14	- +

Figura 5.16: Ejemplo de leyenda de ayuda de la aplicación *web*.



Conducciones ultimo tercio 14 - +

Beneficio liga ini 3300,00 - +

Beneficio liga fin 3300,00 - +

Valor equipo ini 246,40 - +

Valor equipo fin 246,40 - +

☐ Defensa

Aceptar

El salario estimado que cobra Guido Rodríguez (2022/2023), es de 46974.21 euros.

Figura 5.17: Resultado tras ejecutar el modelo que se imprime en la aplicación *web*.





## Capítulo 6

# Experimentos y análisis de resultados

En este apartado se explican todos los experimentos realizados a lo largo del proyecto. La sección está ordenada de forma cronológica, es decir los experimentos se irán explicando en el orden en que fueron realizados. Resumiendo dicha cronología, primero hago experimentos básicos, sin selección de características, con el objetivo de analizar los resultados de cada modelo del conjunto de hipótesis. Más adelante, con los modelos más prometedores, hago varias pruebas eliminando partes del *dataset*. Por último hago experimentos para seleccionar las mejores características y entreno los mejores modelos con este nuevo conjunto, resultado de la selección de características.

### 6.1. Experimentos iniciales

Para los experimentos iniciales utilizamos el *dataset* con las modificaciones que comentamos en los apartados 5.1.2 y 5.1.3. Adicionalmente, para reducir la dimensionalidad del problema utilizamos dos técnicas temporales, ya que en última instancia usaré métodos de selección de características para esta labor. Estas dos técnicas son:

#### **Coefficiente de correlación de *Pearson***

La correlación [80], en el sentido más amplio, sirve para medir la relación entre variables. En datos correlacionados, el cambio en la magnitud de una variable está asociado con un cambio en la magnitud de otra variable, ya sea en la misma dirección (correlación positiva) o en la dirección opuesta (correlación negativa). El coeficiente de correlación de *Pearson* se utiliza normalmente para datos continuos. En nuestro caso todas las variables son

seleccionables, excepto las de posición ya que estas son variables categóricas. El coeficiente de correlación está escalado de manera que va desde  $-1$  hasta  $+1$ , donde  $0$  indica que no hay asociación lineal y la relación se vuelve más fuerte a medida que el coeficiente se acerca a un valor absoluto de  $1$ .

Una vez calculada la correlación entre todos los pares de características, seleccionamos las que tengan más de un  $99\%$  de correlación. Elijo un valor tan alto para, al eliminar una de estas variables, no perder demasiada información. Las parejas con más de un  $99\%$  de correlación son:

- (**'Titularidades', 'Minutos'**): Es entendible la correlación, una titularidad suele equivaler a jugar entre 65 y 90 minutos. Veo coherente entonces eliminar 'Titularidades', ya que su información está prácticamente implícita en 'Minutos'.
- (**'Pases medios completados', 'Pases medios intentados'**): Estas variables hacen referencia a la cantidad de pases de media distancia intentados y completados. La correlación puede deberse a que al final todos los futbolistas tienen una precisión similar para este tipo de pases, fallando así un porcentaje similar. No obstante creo que mantener ambas variables puede ser bueno, ya que al final aunque los porcentajes de precisión sean similares, estos no son idénticos y es interesante que el modelo los tenga en cuenta.
- (**'Pases cortos completados', 'Pases cortos intentados'**): Misma explicación que las variables anteriores, solo que en este caso son pases cortos.
- (**'Disparos recibidos', 'Salvadas'**): En este caso se correlacionan los disparos a puerta rivales con las paradas. Al final a más disparos a puerta haya, más paradas se hará el portero. Aunque al igual que con los pases es interesante guardar ambas métricas para que el modelo tenga en cuenta que porcentaje de esos disparos acaban en las manos del portero.
- (**'Pases intentados', 'Toques'**): Aquí se correlacionan el número de pases intentados por un jugador con el número de toques dado por el jugador. Entiendo que por como es el fútbol hoy en día, donde los jugadores no retienen mucho tiempo el balón y se suele controlar y pasar, se puede correlacionar ambas variables. Aunque sigo creyendo que ambas proporcionan su propia información, por lo que no eliminaré ninguna de las dos.

## Análisis de componentes principales

El análisis de componentes principales (PCA) [1, 99] es una técnica multivariante que analiza un conjunto de datos, en la cual las observaciones son descritas por varias características dependientes cuantitativas intercorrelacionadas. Su objetivo es extraer la información importante del conjunto, representarla como un conjunto de nuevas variables ortogonales llamadas componentes principales y mostrar el patrón de similitud de las observaciones y de las variables como puntos en mapas. En resumen, al utilizarlo, buscamos reducir el número de características del problema, perdiendo el mínimo posible de información.

En nuestro experimento nos quedamos con 38 componentes principales, de las 68 variables continuas que teníamos en un principio. El motivo para tomar esta decisión es que al observar la figura 6.1 podemos ver como al quedarnos con unas 40 características, la varianza explicada empieza a decrecer a un mayor ritmo. Esto es lo que se conoce como *método del codo* [61]. En ese punto la varianza explicada queda entorno al 99%, por lo que decido quedarme con un número de características que mantengan un  $> 99\%$  de explicabilidad, siendo ese número 38. Todo este proceso lo realizamos en el conjunto de entrenamiento, tras ajustarlo transformamos los datos de test con lo aprendido.

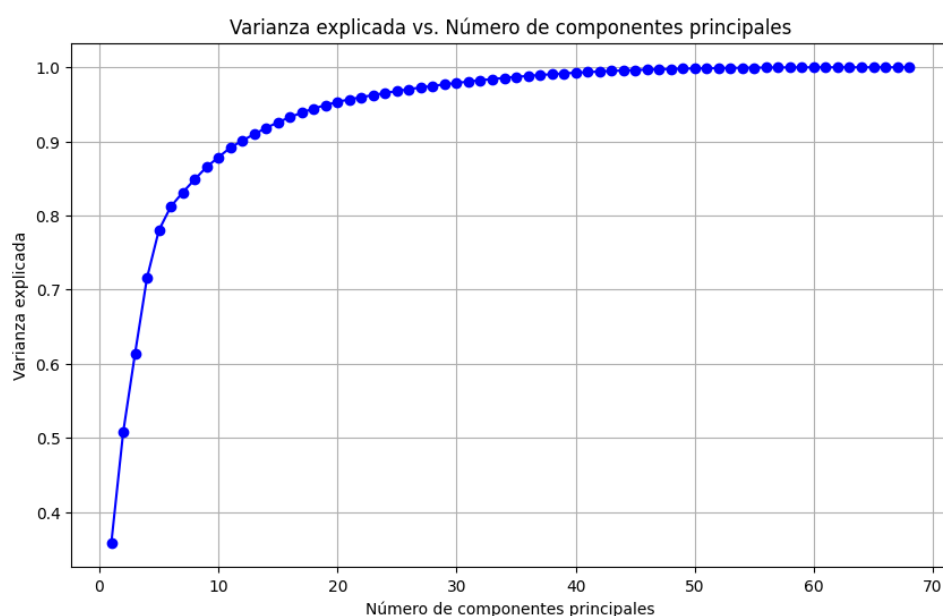


Figura 6.1: Varianza explicada en función del número de componentes principales con el que nos quedamos.

### 6.1.1. k-NN

Como implementación de k-NN utilizaremos la clase *KNeighborsRegressor* de [53]. Los hiperparámetros a tunear son:

- ***n\_neighbors***: Representa el número de vecinos más cercanos que se utilizarán para estimar el valor de regresión para un nuevo punto de datos. Un valor pequeño, como 1 o 3, hará que el modelo sea más sensible a las fluctuaciones en los datos y puede llevar a una predicción más ruidosa. Por otro lado, un valor grande como 10 o más, hará que el modelo sea más suave y robusto al ruido ya que se basa en un mayor número de vecinos cercanos.
- ***weights***: Permite ajustar el impacto de los vecinos en función de su distancia al punto de predicción. Hay dos opciones comunes para el parámetro:
  - ***uniform***: Todos los vecinos más cercanos tienen el mismo peso al realizar la predicción. En otras palabras, cada vecino contribuye igualmente a la predicción.
  - ***distance***: Los vecinos más cercanos tienen un peso inversamente proporcional a su distancia al punto de predicción. Cuanto más cerca esté un vecino, más influencia tendrá en la predicción. Este enfoque refleja la intuición de que los puntos cercanos son más similares y por lo tanto deben contribuir más a la predicción.
- ***p***: Se utiliza para controlar la métrica de distancia utilizada en el cálculo de los vecinos más cercanos. Esta métrica determina cómo se mide la distancia entre los puntos en el espacio de características. El valor de *p* está relacionado con la distancia de *Minkowski*, que es una métrica general que incluye varias métricas comunes de distancia como casos especiales. Dependiendo de la elección de *p*, puedes obtener diferentes métricas de distancia:
  - Cuando  $p = 1$ , se utiliza la distancia de *Manhattan* (también conocida como distancia L1). En este caso, la distancia entre dos puntos se calcula como la suma de las diferencias absolutas de sus coordenadas.
  - Cuando  $p = 2$ , se utiliza la distancia euclidiana (distancia L2). La distancia euclidiana es la distancia 'en línea recta' entre dos puntos en el espacio de características.
  - Cuando *p* toma otros valores (por ejemplo,  $p = 3$ ), se utiliza la distancia de *Minkowski* generalizada, que incluye tanto la distancia de *Manhattan* como la euclidiana como casos especiales.

En la tabla 6.1 podemos observar las opciones finalmente elegidas para cada parámetro. Cabe mencionar, que a la hora de tomar la decisión de qué valores escoger me centro principalmente en que los resultados de dicho valor sean mejores que los del resto de candidatos. De forma más secundaria, busco también que el modelo no sobreajuste demasiado al conjunto de entrenamiento para el valor seleccionado. He considerado que exponer todo este proceso de refinado de parámetros en la memoria puede llegar a resultar demasiado confuso. Es por ello que simplemente muestro una tabla con los valores seleccionados, como puede ser el caso de la tabla 6.1. En caso de que el lector tuviera interés en comprobar el proceso de selección de dichos valores, se encuentra todo perfectamente detallado en el repositorio de *GitHub* anteriormente mencionado (<https://github.com/JMMelcrack/code>).

En cuanto a los resultados finales, obtenemos: **MSE de 0.73, MAE de 0.68 y  $R^2$  de 0.5**. Estos son resultados malos, propios de un modelo muy simple como lo es k-NN, los cuales nos vienen bien como resultados base para poder cuantificar mejor la mejora del resto de modelos.

Cuadro 6.1: Conjunto de hiperparámetros refinados en k-NN.

Nombre hiperparámetro	Rango Valores	Valor seleccionado
$n\_neighbors$	3, 9, 15, 21	9
$weights$	'uniform', 'distance'	'distance'
p	1, 2, 3	2

### 6.1.2. Regresión lineal

Como implementación de regresión lineal utilizaremos la clase *SGDRegressor* de [52]. Los hiperparámetros a tunear son:

- **penalty**: Hace referencia al término de regularización. Considero *l1* como la mejor en este caso por su característica de anular algunas variables y no tenerlas en cuenta, debido a la dimensionalidad del problema. No obstante probaré todos los tipos de regularización para compararlos.
- **alpha**: Es el factor de regularización. A mayor *alpha*, con mayor fuerza se aplicará la regularización.
- **tol**: Representa la tolerancia para el criterio de convergencia del algoritmo de optimización utilizado en el modelo. La tolerancia determina cuándo se considera que el algoritmo ha convergido. Si el cambio en la función objetivo entre dos iteraciones consecutivas es menor que la

tolerancia, se considera que el algoritmo ha convergido y se detiene el entrenamiento.

- ***eta0***: Valor inicial para el *learning rate*.

En la tabla 6.2 podemos observar las opciones finalmente elegidas para cada parámetro. En cuanto a los resultados, obtenemos: **MSE de 0.59, MAE de 0.6 y  $R^2$  de 0.59**. Los resultados mejoran por bastante a k-NN, aunque siguen siendo bajos. La regresión lineal al final es un modelo que no puede capturar las relaciones no lineales del conjunto de datos, haciendo que no pueda obtener grandes resultados. Al final, este modelo también nos servirá como base para comparar.

Cuadro 6.2: Conjunto de hiperparámetros refinados en regresión lineal.

Nombre hiperparámetro	Rango Valores	Valor seleccionado
<i>penalty</i>	'l1', 'l2', ' <i>elasticnet</i> '	'l1'
<i>alpha</i>	1e-4, 1e-3, 0.01, 0.1, 1	0.01
<i>tol</i>	1e-3, 1e-4, 1e-5, <i>None</i>	1e-3
<i>eta0</i>	1e-3, 0.01, 0.1	0.01

### 6.1.3. *Random Forest*

Como implementación de *Random Forest* utilizaremos la clase *RandomForestRegressor* de [50]. Los hiperparámetros a tunear son:

- ***n\_estimators***: Número de árboles totales a generar. A mayor número, más complejo es el modelo resultante y por tanto más puede sobreajustarse a los datos de entrenamiento.
- ***max\_features***: Controla el número máximo de características que se consideran al realizar una división en un nodo. Esto implica limitar la cantidad de características que el algoritmo evaluará para determinar la mejor división. *None* equivale a usar todas las características, que es lo que viene por defecto. Pero debido al peligro de sobreajuste, probaré todas las opciones para comparar resultados.
- ***max\_depth***: Profundidad máxima del árbol, por defecto *None*. No limitar la profundidad del árbol puede causar mucho sobreajuste, por lo que probaré varias opciones para este parámetro dependiendo de la profundidad media de los árboles en el problema.

En la tabla 6.3 podemos observar las opciones finalmente elegidas para cada parámetro. En cuanto a los resultados, obtenemos: **MSE de 0.52,**

**MAE de 0.55 y  $R^2$  de 0.64.** Estos resultados son superiores a los de regresión lineal. El problema es que deberían de ser mucho mejores, ya que *Random Forest* si es un modelo complejo que puede extraer relaciones lineales y no lineales de los datos. No obstante, la figura 6.2 puede resolver nuestra duda. Esta imagen muestra como va actualizándose el  $R^2$  en función del número de ejemplos con el que se entrena. Podemos observar en ella, que el error va mejorando muy despacio según vamos entrenando con más datos. Esto, sumado al enorme sobreajuste que tenemos, nos puede estar diciendo que *Random Forest* es un modelo demasiado complejo para nuestro problema. O dicho de otra forma, que contamos con pocos datos para entrenar un modelo tan complejo.

Cuadro 6.3: Conjunto de hiperparámetros refinados en *Random Forest*.

Nombre hiperparámetro	Rango Valores	Valor seleccionado
<i>n_estimators</i>	20, 50, 100	50
<i>max_features</i>	<i>None</i> , 1.0, 'sqrt', 'log2'	1.0
<i>max_depth</i>	<i>None</i> , 27, 23, 17, 12, 9, 7	12

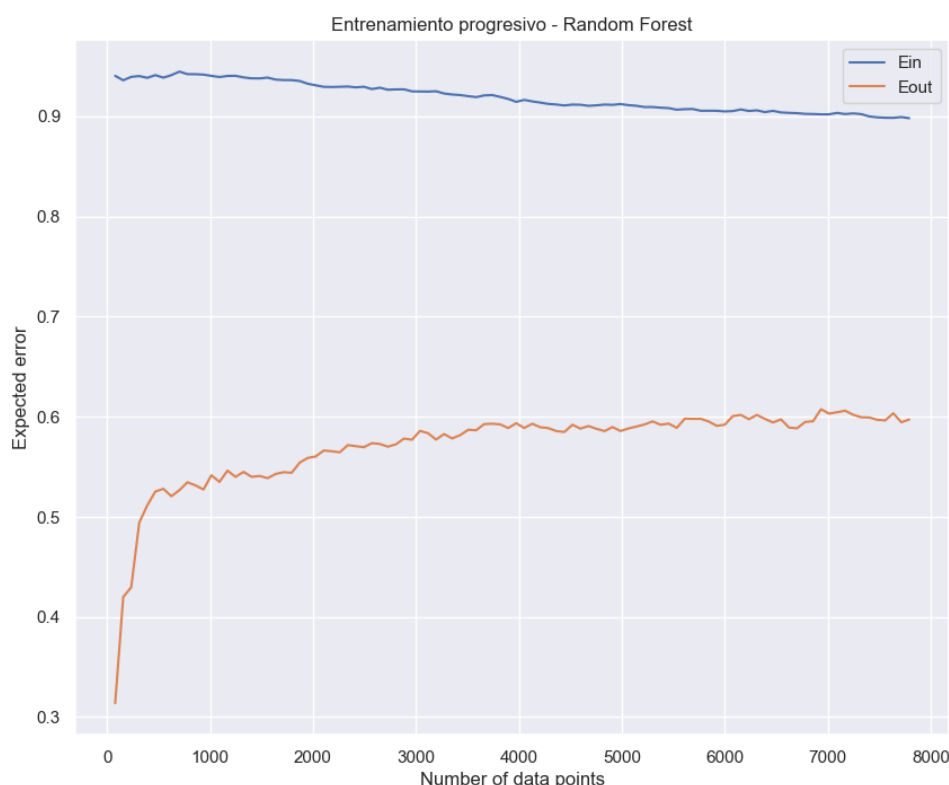


Figura 6.2: Entrenamiento progresivo para el modelo *Random Forest*.

#### 6.1.4. *Gradient Boosting*

Debido a las conclusiones obtenidas con *Random Forest*, decido utilizar un modelo similar pero menos complejo. Este modelo es *Gradient Boosting*, que utiliza árboles más simples que *Random Forest* para realizar las predicciones. Como implementación de *Gradient Boosting* utilizaremos la clase *GradientBoostingRegressor* de [49]. Los hiperparámetros a tunear son:

- ***n\_estimators***: Número de *weak learners* o árboles totales a generar. A mayor número, más puede el modelo sobreajustarse a los datos de entrenamiento.
- ***learning\_rate***: Su principal función es regular la contribución de cada árbol base en el conjunto final.
- ***subsample***: Controla la fracción de las muestras que se utilizan para entrenar cada árbol base en el proceso de *Gradient Boosting*. Un valor de 1.0 significa que se utilizan todas las muestras. Mientras que un valor menor reduce la cantidad de datos utilizados para entrenar cada árbol.
- ***max\_depth***: Profundidad máxima hasta la que pueden crecer los árboles, por defecto es 3.
- ***max\_features***: Controla el número máximo de características que se consideran al realizar una división en un nodo. Esto implica limitar la cantidad de características que el algoritmo evaluará para determinar la mejor división. *None* equivale a usar todas las características, que es lo que viene por defecto. Pero debido al peligro de sobreajuste, probaré todas las opciones para comparar resultados.

En la tabla 6.4 podemos observar las opciones finalmente elegidas para cada parámetro. En cuanto a los resultados, obtenemos: **MSE de 0.48, MAE de 0.53 y  $R^2$  de 0.67**. Los resultados consiguen mejorar algo los registros de *Random Forest*. Aparte de como se puede ver en la figura 6.3, conseguimos mitigar el sobreajuste. No obstante, pensando en que tal vez pueda conseguir mejorar estos resultados decido probar a entrenar otro modelo de complejidad similar. Este va a ser el Perceptrón Multicapa, utilizando una arquitectura pequeña para no aumentar la complejidad del modelo demasiado.

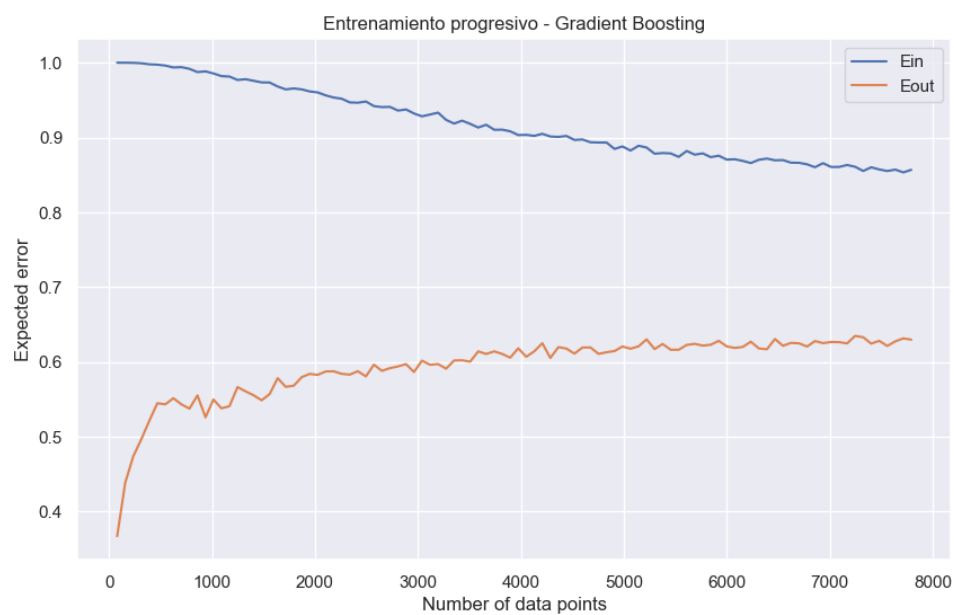
#### 6.1.5. Perceptrón Multicapa

Como implementación del Perceptrón Multicapa utilizaremos la clase *MLPRegressor* de [54]. Los hiperparámetros a tunear son:



Cuadro 6.4: Conjunto de hiperparámetros refinados en *Gradient Boosting*.

Nombre hiperparámetro	Rango Valores	Valor seleccionado
<i>n_estimators</i>	20, 50, 100, 200	100
<i>learning_rate</i>	0.04, 0.08, 0.1, 0.4, 0.8	0.1
<i>subsample</i>	1.0, 0.8, 0.6, 0.4	0.6
<i>max_depth</i>	2, 3, 4, 6	6
<i>max_features</i>	<i>None</i> , 'sqrt', 'log2'	<i>None</i>

Figura 6.3: Entrenamiento progresivo para el modelo *Gradient Boosting*.

- ***hidden\_layer\_sizes***: Este parámetro acepta una tupla que especifica la estructura de capas ocultas de la red. Los elementos de la tupla representan el número de neuronas en cada capa oculta y la longitud de la tupla determina la cantidad de capas ocultas en la red.
- ***activation***: Su objetivo es determinar la función de activación utilizada en las neuronas de la red. La función de activación es una parte esencial de una red neuronal y se aplica a la salida de cada neurona en una capa para determinar su salida final.
- ***solver***: Sirve para especificar el algoritmo de optimización que se utilizará durante el entrenamiento de la red neuronal. El algoritmo de optimización es responsable de ajustar los pesos de la red para minimizar la función de pérdida.
- ***alpha***: Controla el término de regularización L2 (también conocido como regularización de peso o regularización *Ridge*).
- ***learning\_rate\_init***: Valor inicial del *learning rate*.

En la tabla 6.5 podemos observar las opciones finalmente elegidas para cada parámetro. En cuanto a los resultados, obtenemos: **MSE de 0.46, MAE de 0.52 y  $R^2$  de 0.68**. Al final resultados muy similares a los obtenidos en el apartado anterior. No obstante, en lo que el Perceptrón Multicapa destaca es en su buena generalización. Como se puede observar en la figura 6.4, el sobreajuste es mínimo. Hay que comentar también, referente a 6.4, que las fluctuaciones de la gráfica  $E_{in}$  se deben a que en determinados entrenamientos el modelo no llegaba a converger, finalizando el entrenamiento por alcanzar el máximo número de iteraciones y devolviendo así peores resultados, que son los puntos anómalos de la gráfica.

Cuadro 6.5: Conjunto de hiperparámetros refinados en el Perceptrón Multicapa.

Nombre hiperparámetro	Rango Valores	Valor seleccionado
<i>hidden_layer_sizes</i>	28, 56, (28,28), (56,56)	28
<i>activation</i>	'relu', 'tanh'	'tanh'
<i>solver</i>	'adam', 'lbfgs'	'adam'
<i>alpha</i>	1e-3, 0.01, 0.1, 1, 2.5, 5	1
<i>learning_rate_init</i>	1e-3, 0.01, 0.1, 1	1e-3

### 6.1.6. Conclusiones iniciales

Tanto en la tabla 6.6 como en las figuras 6.5 y 6.6, se muestran un resumen con los mejores resultados resultados de cada modelo. En 6.5 se ven los

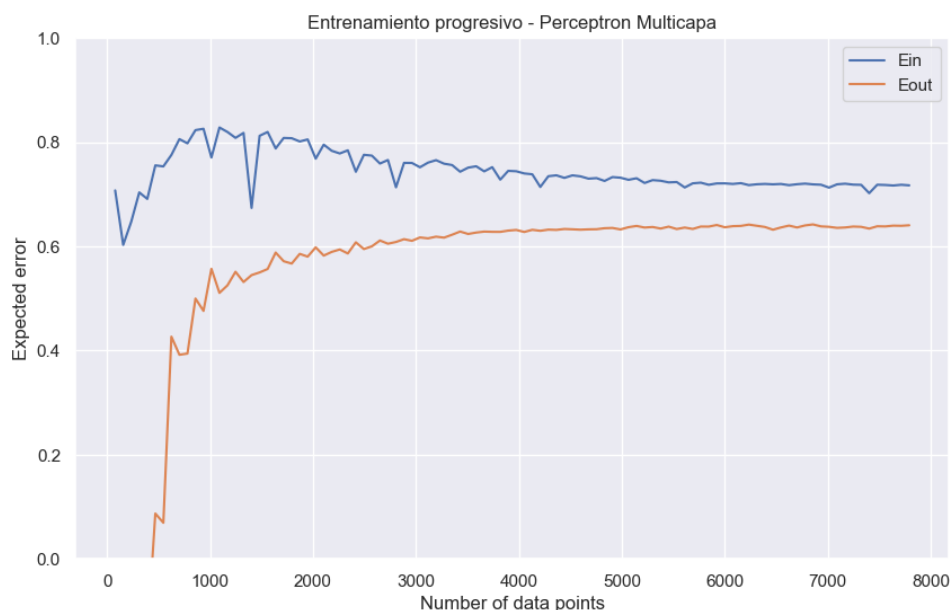


Figura 6.4: Entrenamiento progresivo para el modelo de Perceptrón Multicapa.

resultados para el conjunto de entrenamiento, y en 6.6 para el de validación. *Dummy-mean* (D-m), es un modelo *naive* que simplemente predice el salario medio siempre. Muestro sus resultados para comparar como de superiores son los resultados de mi conjunto de hipótesis a los de un método de predicción trivial. Podemos apreciar que estos resultados son mucho peores, por lo que en un primer análisis los modelos entrenados hacen un buen trabajo.

Por otro lado, como he comentado antes, podemos ver que los mejores modelos son *Gradient Boosting* y el Perceptrón Multicapa. Por un lado GB es más explicativo, lo cual nos puede ser muy útil a la hora de explicar las predicciones del modelo. Por otro lado, MLP generaliza mejor y sus resultados son algo superiores, por lo que reduciremos nuestro conjunto de hipótesis a ambos modelos. A continuación me centraré en reducir el *dataset* en función de las posiciones. Esto debido a que como se puede observar en las figuras 5.10 y 5.8, puede ser interesante entrenar modelos que solo tengan en cuenta una posición ya que los jugadores de una misma posición tienen características más similares y esto puede reducir la complejidad del problema. Del mismo modo, puede ser interesante eliminar a los porteros del *dataset*, ya que como vemos en la figura 5.8 esa es la posición que menos parecido tiene al resto.

Cuadro 6.6: Resultados obtenidos para todos los modelos iniciales.

Métricas error	k-NN	RL	RF	GB	MLP	D-m
$MSE_{train}$	0.0	0.5884	0.1509	0.209	0.4155	1.4602
$MAE_{train}$	0.0	0.5948	0.303	0.3293	0.4906	0.9532
$R^2_{train}$	1.0	0.597	0.8967	0.8569	0.7154	0.0
$MSE_{val}$	0.7289	0.5908	0.5227	0.4822	0.464	1.4603
$MAE_{val}$	0.6763	0.5958	0.5517	0.5262	0.5206	0.9533
$R^2_{val}$	0.5008	0.5955	0.642	0.6698	0.6823	0.0

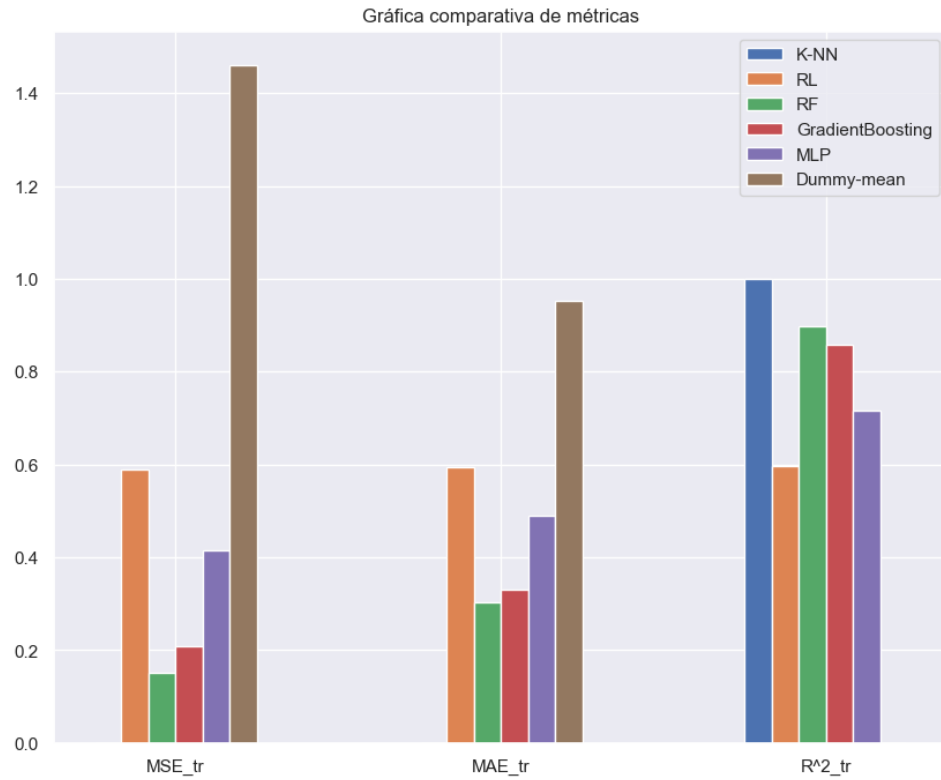


Figura 6.5: Resultados de los modelos iniciales para el conjunto de entrenamiento.

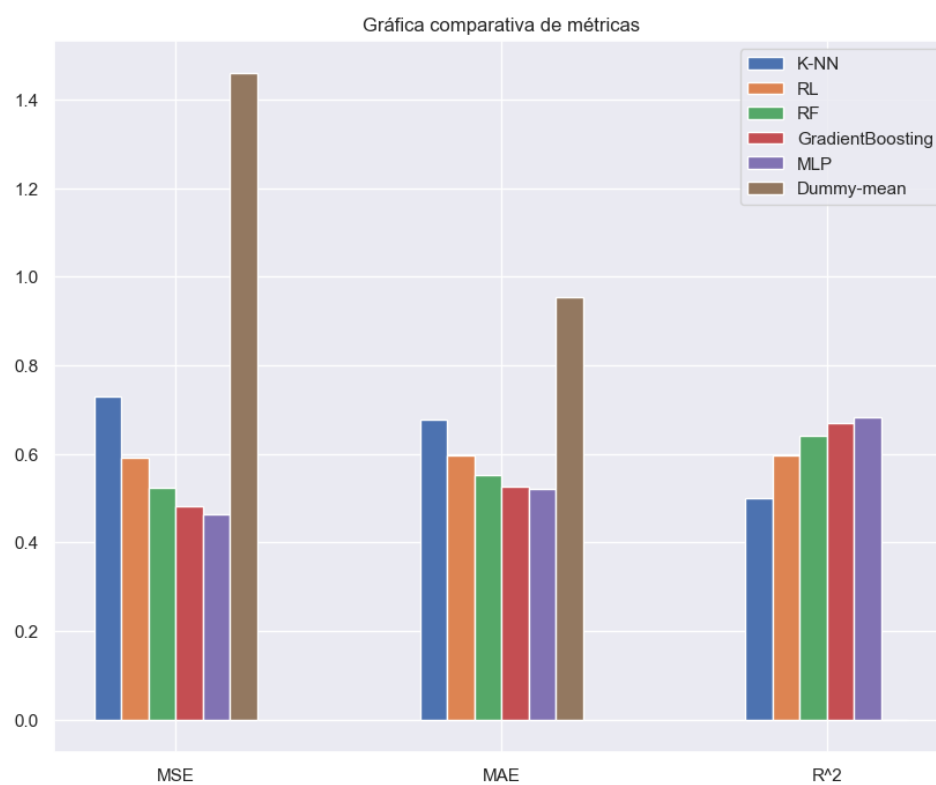


Figura 6.6: Resultados de los modelos iniciales para el conjunto de validación.

## 6.2. Pruebas sin porteros

En las figuras 5.10 y 5.8 se puede ver claramente como la mayor parte de los porteros (los que más minutos juegan) forma un *cluster* independiente al resto de jugadores. Esta distribución tan distinta puede provocar confusión al modelo a la hora de ajustarse al conjunto de datos, lo cual provoca que los resultados empeoren. Por ello decido eliminar a los porteros del *dataset* y entrenar los modelos más prometedores (GB y MLP) con este nuevo conjunto. Aparte de eliminar a los porteros, eliminamos también las características que solo les afectaran a ellos como por ejemplo 'Goles encajados' o 'Salvadas'. Quedándonos así, con **9872** instancias y **64** características en el problema. Por lo demás, realizamos el mismo procesado y análisis al *dataset* que en el apartado 6.1. A diferencia de en ese apartado, como ahora no tenemos porteros al mostrar la distribución de los datos con t-SNE, vemos que todos pertenecen a un único conjunto y se superponen unos a otros, esto se puede observar en la figura 6.7. Finalmente tras aplicar PCA nos quedamos con **37** características.

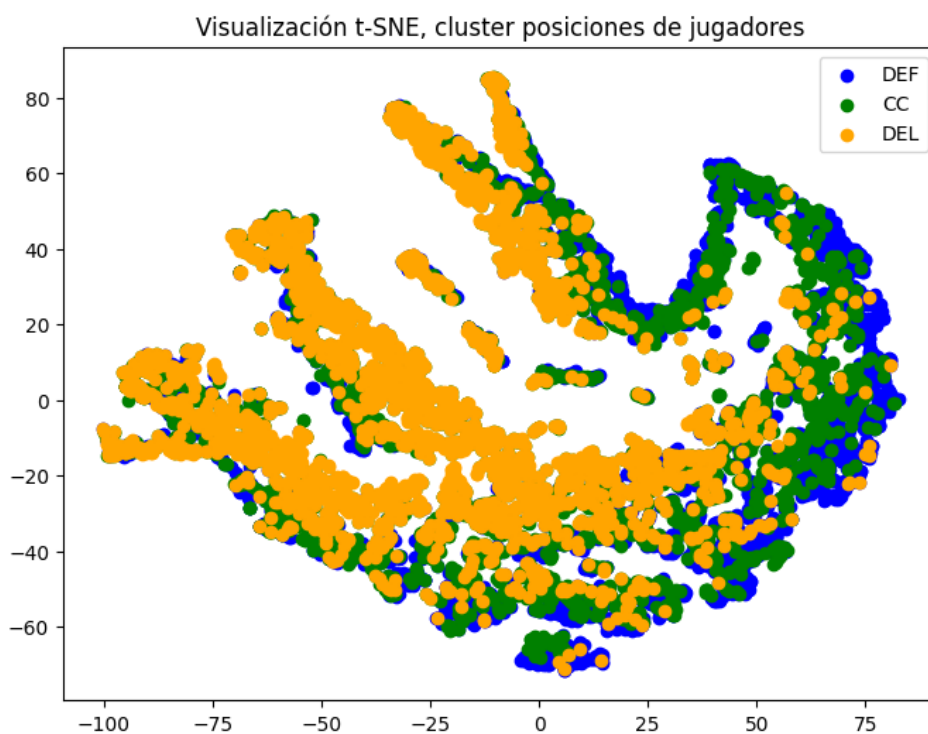


Figura 6.7: Distribución de los jugadores de campo en base a la posición del futbolista.

A la hora de *finetunear* los modelos utilizo los mismos rangos de valores para los mismos hiperparámetros que en el apartado 6.1. En esta ocasión

los parámetros finales obtenidos han sido, para *Gradient Boosting*:

- *n\_estimators*: 100
- *learning\_rate*: 0.1
- *subsample*: 0.6
- *max\_depth*: 4
- *max\_features*: *None*

Y para el Perceptrón Multicapa:

- *hidden\_layer\_sizes*: 28
- *activation*: 'tanh'
- *solver*: 'adam'
- *alpha*: 1
- *learning\_rate\_init*: 0.001

Los resultados de los entrenamientos se pueden ver en la tabla 6.7, así como en las figuras 6.8 (conjunto de entrenamiento) y 6.9 (conjunto de validación). Estos resultados no son mucho mejores que los obtenidos en 6.6, de hecho en el caso de GB estos empeoran, probablemente debido a que disponemos de menos datos para entrenar. Lo único bueno es que el *overfitting* de *Gradient Boosting* disminuye en este caso. Por otro lado MLP si que mejora algo, aunque la insignificante mejora no justifica eliminar a los porteros del conjunto.

Cuadro 6.7: Resultados obtenidos para los modelos entrenados con el *dataset* sin porteros.

Métricas error	GB	MLP
$MSE_{train}$	0.3547	0.4091
$MAE_{train}$	0.4443	0.4855
$R^2_{train}$	0.76	0.7232
$MSE_{val}$	0.4923	0.4554
$MAE_{val}$	0.5335	0.5142
$R^2_{val}$	0.6668	0.6918

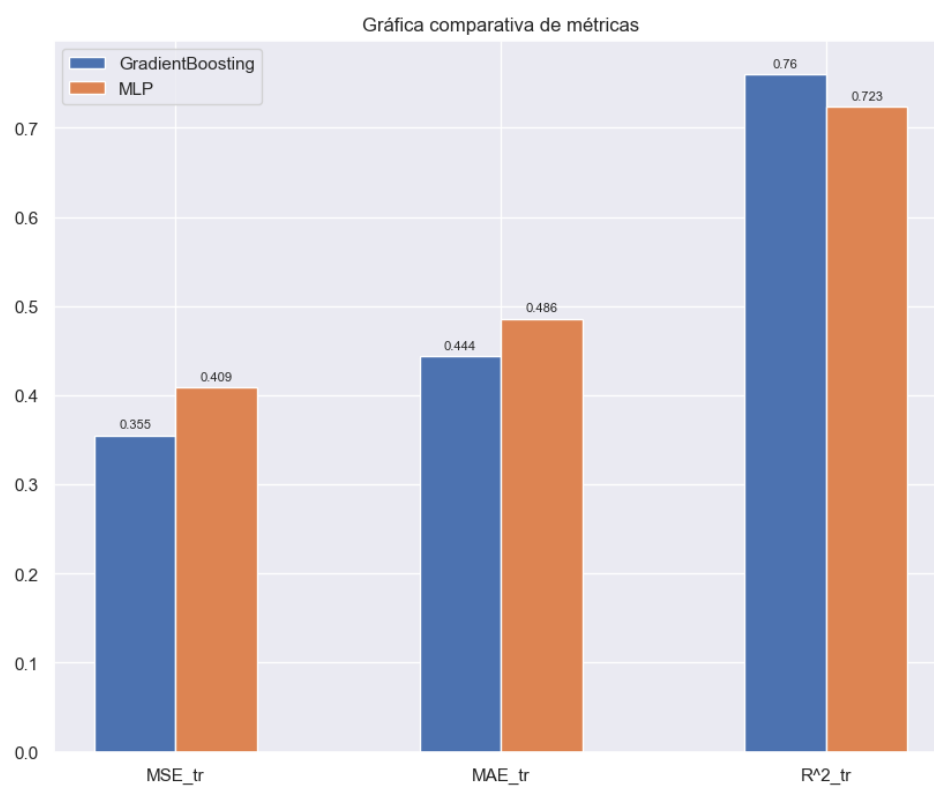


Figura 6.8: Resultados de los modelos entrenados sin porteros para el conjunto de entrenamiento.



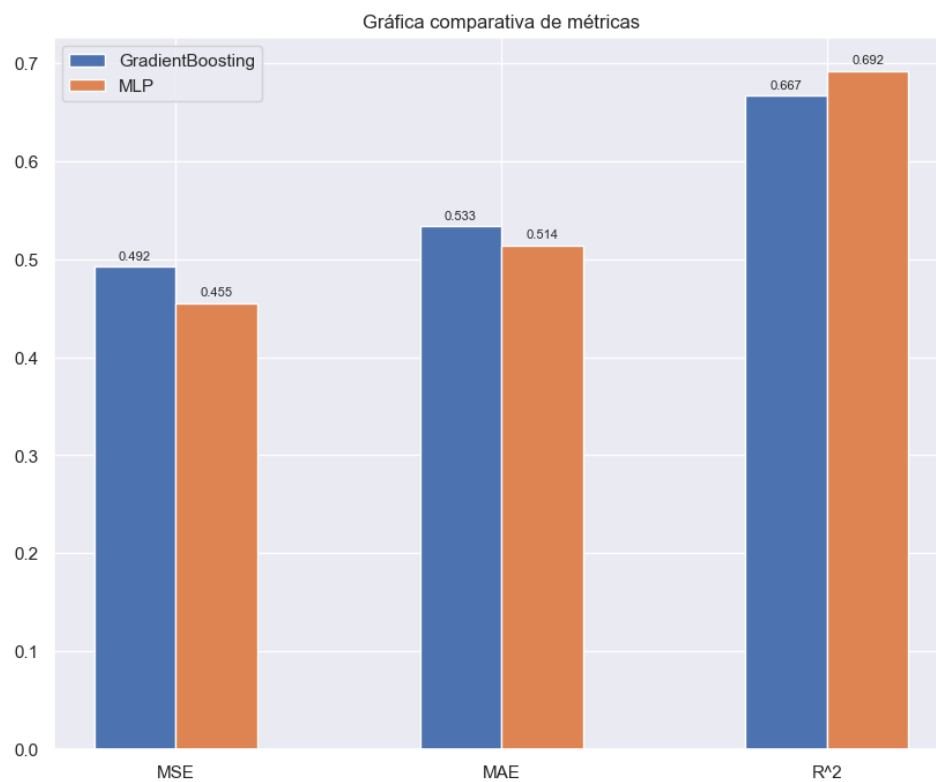


Figura 6.9: Resultados de los modelos entrenados sin porteros para el conjunto de validación.

### 6.3. Pruebas considerando una única posición

En este caso, siguiendo la misma lógica que en el apartado anterior sobre que los jugadores de una única posición son más similares entre sí, seleccionamos solo una posición para entrenar. Elijo usar a los centrocampistas, ya que esta es la clase mayoritaria como se puede observar en la figura 5.4. De nuevo, como en el anterior experimento, utilizamos los modelos más prometedores y eliminamos las características que ya no nos aportan ninguna información. Quedándonos así con **4839** ejemplos y **61** características en el problema. También, de nuevo, realizamos el mismo procesado y análisis al *dataset* que en el apartado 6.1 y tras aplicar PCA nos quedan **35** características.

A la hora de *finetune*ar los modelos utilizo los mismos rangos de valores para los mismos hiperparámetros que en el apartado 6.1. En esta ocasión los parámetros finales obtenidos han sido, para *Gradient Boosting*:

- **n\_estimators**: 100
- **learning\_rate**: 0.1
- **subsample**: 0.6
- **max\_depth**: 4
- **max\_features**: None

Y para el Perceptrón Multicapa:

- **hidden\_layer\_sizes**: 28
- **activation**: 'tanh'
- **solver**: 'adam'
- **alpha**: 2.5
- **learning\_rate\_init**: 0.001

Los resultados de los entrenamientos se pueden ver en la tabla 6.8, así como en las figuras 6.10 (conjunto de entrenamiento) y 6.11 (conjunto de validación). Estos resultados son, en general, peores que los de 6.6. Este suceso se debe tal vez a la falta de datos para entrenar, ya que al usar solo centrocampistas contamos con menos de la mitad del *dataset* original. Podemos entonces concluir en que no sale a cuenta reducir la complejidad del problema a costa de eliminar parte del conjunto de datos para quedarnos solo con una posición.

Cuadro 6.8: Resultados obtenidos para los modelos entrenados con el *dataset* de solo centrocampistas.

Métricas error	GB	MLP
$MSE_{train}$	0.3084	0.4297
$MAE_{train}$	0.4107	0.4995
$R^2_{train}$	0.8016	0.7236
$MSE_{val}$	0.5092	0.4731
$MAE_{val}$	0.5425	0.5256
$R^2_{val}$	0.6711	0.6943

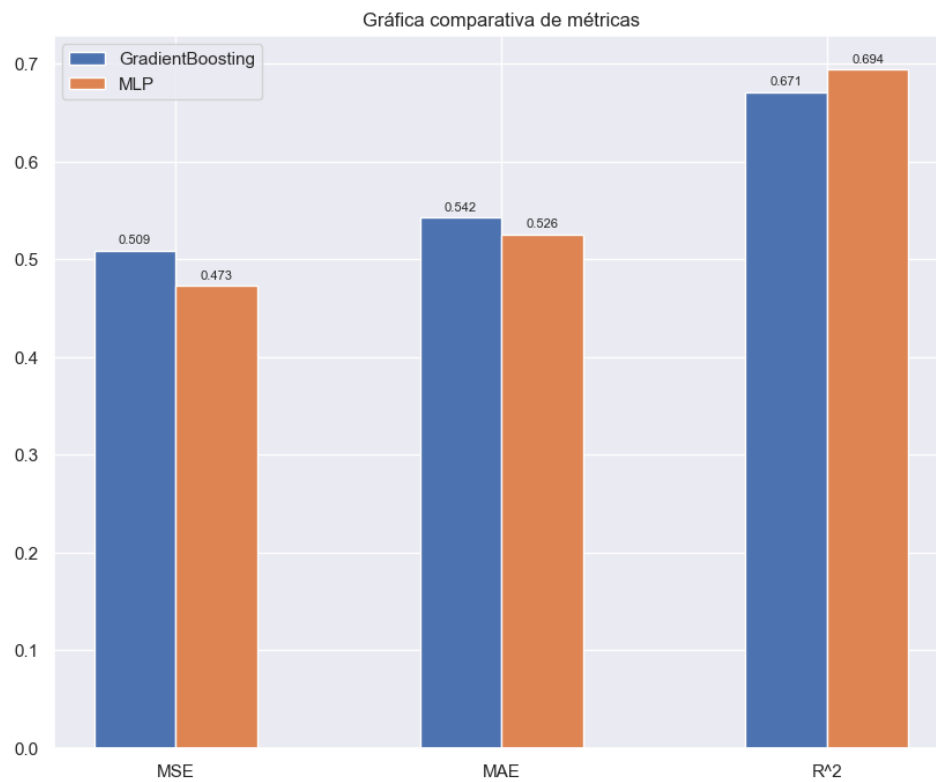


Figura 6.10: Resultados de los modelos entrenados solo con centrocampistas para el conjunto de entrenamiento.

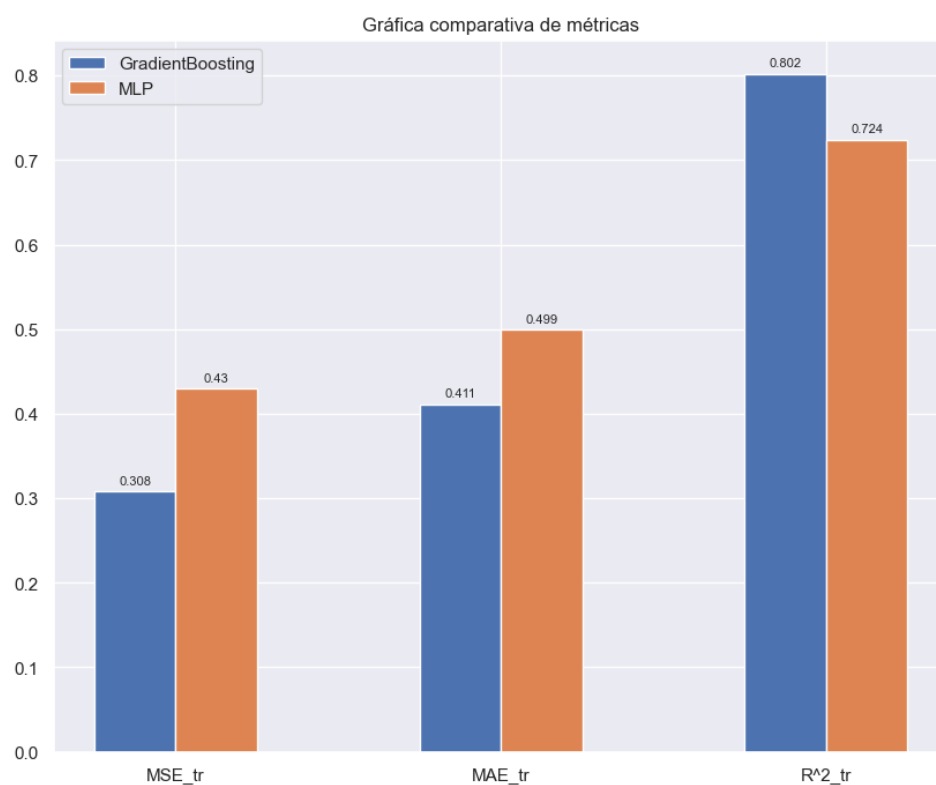


Figura 6.11: Resultados de los modelos entrenados solo con centrocampistas para el conjunto de validación.

## 6.4. Experimentos de selección de características

Tras haber concluido en 6.1 que los mejores modelos son *Gradient Boosting* y el Perceptrón Multicapa. Vamos ahora a realizar la selección de características para reducir la dimensionalidad al igual que hacíamos con PCA, pero en esta ocasión permitiéndonos tener un control de con qué variables nos quedamos.

### 6.4.1. Información Mutua de Selección de Características

Para aplicar esta función, utilizamos la función *SelectKBest* de *scikit-learn* [51], la cual dado un valor  $k$  y una función de puntuación, selecciona las  $k$  mejores características basándose en dicha función. En este caso, la función utilizada es MIFS, en cuanto al valor de  $k$ , usamos los siguientes valores:  $k = \{10, 20, 30, 40, 50\}$ . Una vez obtenidos los conjuntos reducidos, los usamos para entrenar, para poder ver los resultados y así medir la calidad de los *datasets*. El modelo a utilizar es *Gradient Boosting*, con 50 estimadores y el resto de parámetros por defecto.

En el caso de MIFS, como podemos ver en la tabla 6.9 cuando nos quedamos con las 30, 40 y 50 mejores características los resultados son muy similares. Por eso elijo el *dataset* generado con  $k = 30$ , para así reducir la dimensionalidad lo máximo posible.

Cuadro 6.9: Resultados obtenidos por MIFS para distintos números de características ( $k$ ) para el conjunto de validación.

	<b>k = 10</b>	<b>k = 20</b>	<b>k = 30</b>	<b>k = 40</b>	<b>k = 50</b>
MSE	0.6154	0.5946	0.4683	0.4673	0.4674
MAE	0.5839	0.5723	0.5164	0.5156	0.5153
R <sup>2</sup>	0.5807	0.5949	0.681	0.6817	0.6817

### 6.4.2. Coeficiente r de *Pearson* y *F-value*

En este apartado utilizamos el mismo método usado antes con MIFS. En el caso del coeficiente r de *Pearson*, como se puede ver en la tabla 6.10 tenemos el mismo caso que antes, a partir de las 30 características más importantes los resultados se estancan, por lo que seleccionamos dicho *dataset*. Por otro lado, al utilizar la función de *F-value* es a partir de los 40 mejores resultados cuando los resultados dejan de mejorar, como se puede observar en la tabla 6.11, seleccionando así este conjunto de 40 variables como el mejor.

Cuadro 6.10: Resultados obtenidos por *Pearson's r* para distintos números de características ( $k$ ) para el conjunto de validación.

	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 50$
MSE	0.6585	0.5796	0.4676	0.4689	0.4669
MAE	0.6049	0.5632	0.5159	0.5168	0.5147
$R^2$	0.5514	0.6051	0.6815	0.6806	0.682

Cuadro 6.11: Resultados obtenidos por *F value* para distintos números de características ( $k$ ) para el conjunto de validación.

	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 50$
MSE	0.6577	0.5814	0.5748	0.4678	0.4666
MAE	0.6045	0.5639	0.5614	0.516	0.515
$R^2$	0.552	0.6039	0.6084	0.6814	0.6822

### 6.4.3. Eliminación Recursiva de Características

Este algoritmo, al ser un método de envoltorio, necesita un modelo para evaluar la importancia de las características. El método seleccionado es de nuevo *Gradient Boosting* con 50 estimadores y el resto de parámetros por defecto. Para el número de características con las que quedarme, de nuevo utilizo los mismos valores para  $k$  que en los casos anteriores. Los resultados obtenidos los podemos visionar en la siguiente tabla 6.12. Vemos que curiosamente para todos los valores de  $k$  se obtienen resultados similares, siendo los mejores, por poco, los obtenidos escogiendo las 20 mejores características.

Cuadro 6.12: Resultados obtenidos por RFE para distintos números de características ( $k$ ) para el conjunto de validación.

	$k = 10$	$k = 20$	$k = 30$	$k = 40$	$k = 50$
MSE	0.4685	0.4637	0.4656	0.4663	0.4657
MAE	0.5166	0.5129	0.5141	0.5142	0.5139
$R^2$	0.6809	0.6842	0.6828	0.6824	0.6828

### 6.4.4. Selector de Características Las Vegas (filtro)

El método Las Vegas en su versión filtro, en lugar de recibir como argumento el número de características que se desea seleccionar, recibe un número de iteraciones y se ejecuta por ese número, reduciendo el número de características a escoger. Este valor, se fija en  $77 * N$  [58], siendo  $N$  el número

inicial de variables. La idea es encontrar un compromiso entre obtener una solución rápido y que esta sea buena. En cuanto a los resultados, el algoritmo devuelve 3 soluciones con 23 variables. En la tabla 6.13 podemos ver la calidad de las soluciones, la cual es baja. Probablemente esto se deba a que este tipo de problema no termina de encajar con este tipo de algoritmo, el cual necesita que todos los datos estén discretizados. Finalmente, debido a la pésima calidad de los resultados, no selecciono ninguna de las soluciones.

Cuadro 6.13: Resultados obtenidos por LVF para las distintas soluciones obtenidas para el conjunto de validación.

	Solución 1	Solución 2	Solución 3
MSE	0.84	0.7009	0.6451
MAE	0.7048	0.6325	0.5978
R <sup>2</sup>	0.4276	0.5223	0.5607

#### 6.4.5. Selector de Características Las Vegas (envoltorio)

Al dar tan malos resultados LVF decido modificar la función de puntuación, transformando el método Las Vegas a tipo *wrapper*. De nuevo como este método es de tipo envoltura, utilizo *Gradient Boosting* con 50 estimadores para valorar la calidad de las soluciones. En cuanto al número de iteraciones utilizado en LVW, uso de nuevo  $77 * N$ , siendo  $N$  el número inicial de variables. Las soluciones que devuelve el algoritmo LVW vuelven a ser 3, quedándose con las 28 mejores características. Sus resultados se pueden ver en la tabla 6.14, en ella observamos que la solución 1 obtiene resultados similares a los otros métodos, siendo esta la mejor solución de las tres propuestas por LVW.

Cuadro 6.14: Resultados obtenidos por LVW para las distintas soluciones obtenidas para el conjunto de validación.

	Solución 1	Solución 2	Solución 3
MSE	0.4722	0.5984	0.5768
MAE	0.5168	0.5936	0.562
R <sup>2</sup>	0.6783	0.5922	0.607

#### 6.4.6. Conclusiones en la selección de características

Una vez elegidos los conjuntos de características seleccionadas, los comparamos entre ellos para seleccionar el mejor *dataset*. Observando la figura

6.12 podemos ver que el método de Eliminación Recursiva de Características destaca sobre el resto por poco. En la tabla 6.15 se pueden apreciar los valores de dichas diferencias. Aparte de tener los mejores resultados, el *dataset* generado por RFE es el que menos características selecciona. Permitiendo esto reducir al máximo la dimensionalidad del problema. Dicho esto, seleccionamos a RFE como el mejor método de selección de características para este problema y su *dataset* generado como el conjunto con el que entrenar los modelos más prometedores.

Cuadro 6.15: Mejores resultados de las técnicas de selección de características para el conjunto de validación.

Métodos:	MIFS	Coef. r <i>Pearson</i>	<i>F-value</i>	RFE	LVW
MSE	0.4683	0.4676	0.4678	0.4637	0.4722
MAE	0.5164	0.5159	0.516	0.5129	0.5168
R <sup>2</sup>	0.681	0.6815	0.6814	0.6842	0.6783
Nº variables	30	30	40	20	28

## 6.5. Experimentos finales

Una vez tenemos nuestro conjunto de características reducido, ya podemos pasar a los experimentos finales. En este punto, seleccionamos el conjunto de datos inicial (con todas las posiciones). Luego a ese *dataset* le aplicamos las modificaciones comentadas en los apartados 5.1.2 y 5.1.3. Recordar que estas modificaciones son: limpiar el *dataframe* inicial, escalar logarítmicamente la variable a predecir y normalizar el resto de características entre 0 y 1. Adicionalmente, de este *dataset* nos quedamos solo con las características más importantes obtenidas en el apartado anterior, en la selección de características. Con el conjunto de datos ya preparado, entrenamos los modelos más prometedores de la sección 6.1, *Gradient Boosting* y el Perceptrón Multicapa. En este experimento, como ya hemos reducido la dimensionalidad mediante selección de características, no utilizamos PCA ni el coeficiente de correlación de *Pearson*.

Para los mejores modelos, volvemos a realizar una búsqueda de hiperparámetros como en los experimentos iniciales. Probamos los mismos parámetros y los mismos candidatos de las tablas 6.5 y 6.4. No obstante en el caso de MLP, para el parámetro *hidden\_layer\_sizes* utilizamos valores más pequeños, debido a que el *dataset* es menos complejo. Estos nuevos valores son: 15, (15, 15), 30 y (30, 30). Los parámetros finales elegidos han sido, para *Gradient Boosting*:

- *n\_estimators*: 100



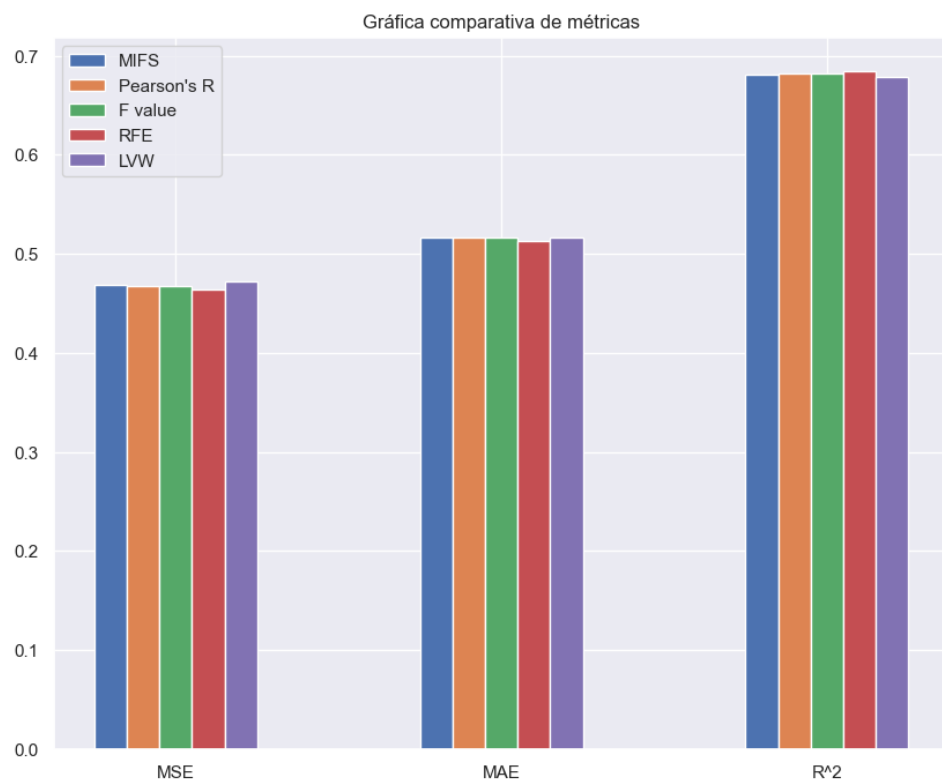


Figura 6.12: Comparativa de las técnicas de selección de características con mejores resultados.

- *learning\_rate*: 0.1
- *subsample*: 0.8
- *max\_depth*: 6
- *max\_features*: *None*

Y para el Perceptrón Multicapa:

- *hidden\_layer\_sizes*: 30
- *activation*: 'tanh'
- *solver*: 'adam'
- *alpha*: 0.0001
- *learning\_rate\_init*: 0.001

Tras entrenar ambos modelos con el *dataset* reducido, analizamos sus resultados. Por un lado en la figura 6.13 se pueden ver los resultados del conjunto de entrenamiento y en la figura 6.14, los del conjunto de validación. Podemos observar que *Gradient Boosting* es superior a MLP en todas las métricas. GB tiene también más *overfitting*, pero al final para ejemplos de fuera de la muestra obtiene mejores resultados que MLP, aunque generalice peor. Con tan clara diferencia, seleccionamos *Gradient Boosting* como el mejor modelo del conjunto de hipótesis inicial.

Una vez tenemos seleccionado el modelo con el que resolver el problema, debemos comprobar que resultados obtiene para el conjunto de test. A este conjunto, que fue el que separamos en el apartado 5.1.3, le aplicamos la selección de características aprendida por RFE. Una vez lo tenemos en el formato correcto, entrenamos el modelo de GB. **Esta vez utilizando todo el *dataset* de entrenamiento, sin aplicar *Cross validation*.** Posteriormente realizamos las predicciones en el conjunto de test. Los resultados obtenidos en dicho conjunto son:

- **MSE**: 0.4573
- **MAE**: 0.4978
- $R^2$ : 0.7034

Al comparar los resultados con los obtenidos en la figura 6.14, podemos afirmar que el modelo generaliza bien para ejemplos de fuera de la muestra.

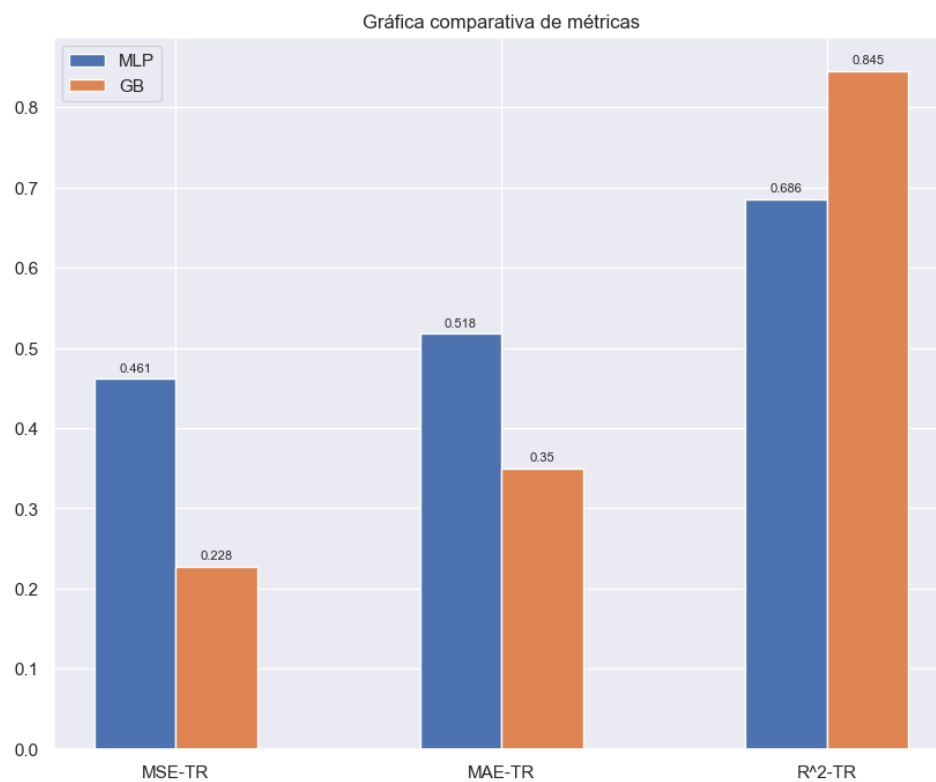


Figura 6.13: Resultados para el conjunto reducido de entrenamiento de los mejores modelos.

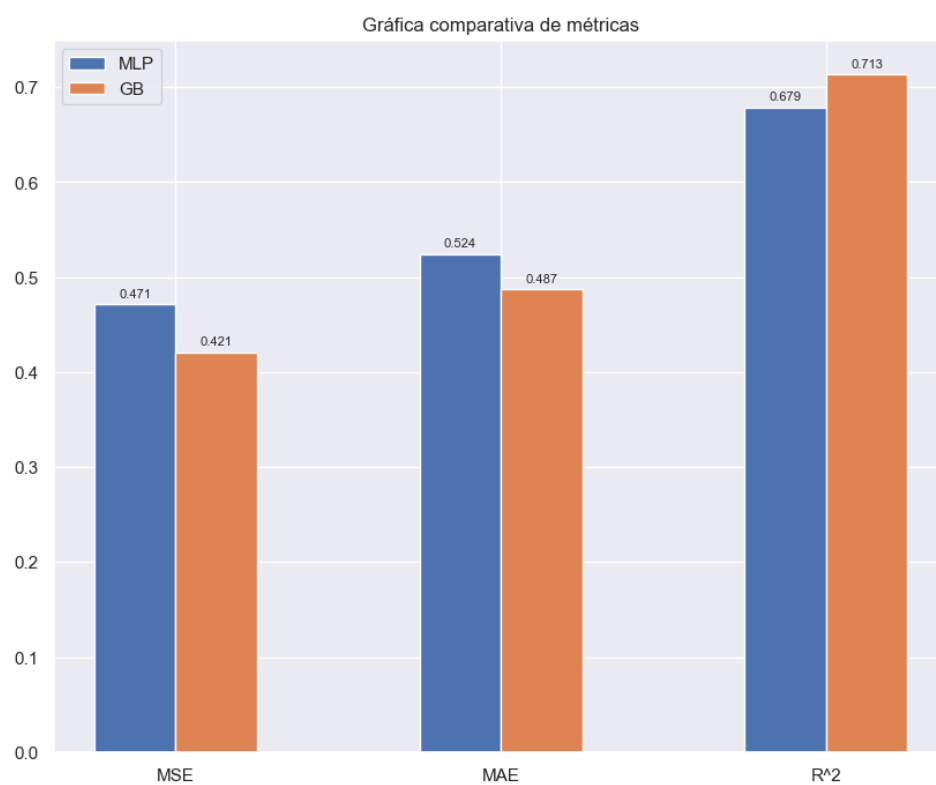


Figura 6.14: Resultados para el conjunto reducido de validación de los mejores modelos.

Aparte de analizar los resultados obtenidos para las métricas de error seleccionadas, también es interesante observar el margen de error del modelo a la hora de realizar las predicciones. En la figura 6.15 se puede observar la dispersión del residuo<sup>1</sup>, donde permitiendo un margen de error del 10 % observamos que la mayoría de predicciones son correctas. Más exactamente, de los 1079 ejemplos de test, 958 se predicen dentro del margen, 88 se sobrestiman y 33 se infraestiman. O dicho de otra forma, el **88.79 %** de los ejemplos se estiman correctamente dentro del margen de error del 10 %.

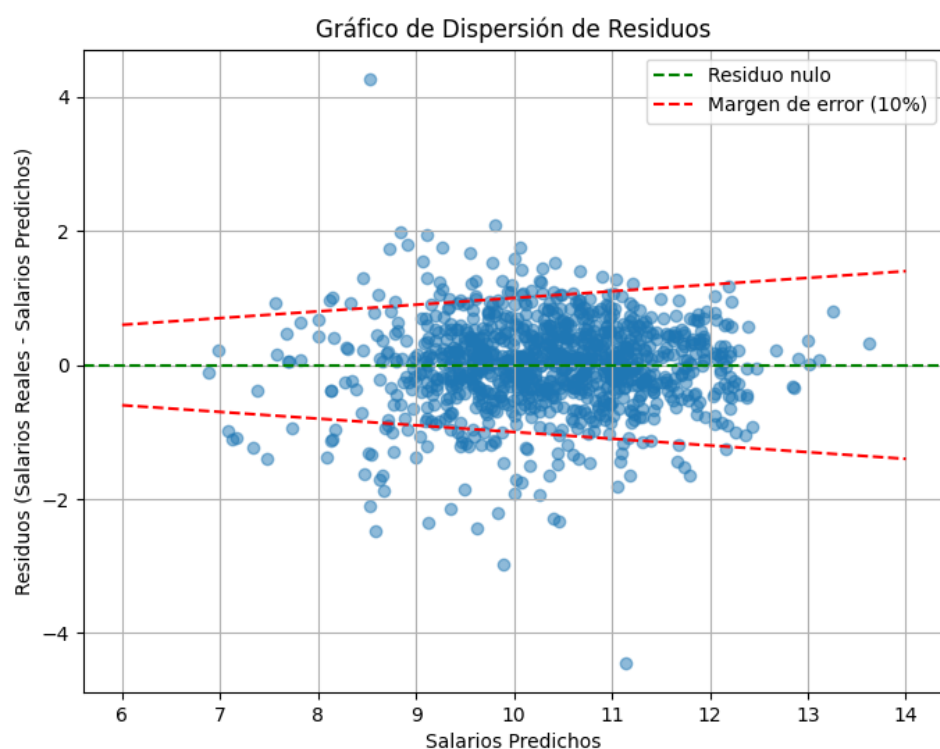


Figura 6.15: Dispersión de las predicciones del conjunto de test.

### 6.5.1. Explicabilidad de las predicciones

A la hora de obtener la importancia de las características, la clase *Gradient Boosting* de *Scikit Learn* tiene un atributo, *feature\_importances\_*, que ya devuelve estos valores. Dichos valores se calculan utilizando el *ranking* relativo de las variables. Las características utilizadas en la parte superior del árbol contribuyen a la decisión de predicción final de una mayor fracción de las muestras de entrada. Por lo tanto, la fracción esperada de las muestras a

<sup>1</sup>El residuo de un ejemplo, se calcula restando el valor real de dicho ejemplo, menos la predicción obtenida.

las que contribuyen puede usarse como una estimación de la importancia relativa de las características. En *scikit-learn*, la fracción de muestras a las que contribuye una característica se combina con la disminución de la impureza al dividir las para crear una estimación normalizada del poder predictivo de esa característica. Al promediar las estimaciones de la capacidad predictiva sobre varios árboles aleatorizados, se puede reducir la varianza de dicha estimación y usarla para la selección de características. Esto se conoce como la disminución media de la impureza, o MDI [59].

Una vez calculadas las importancias, podemos ver como queda el *ranking* en la siguiente tabla, 6.16. La variable más importante por mucho es el 'Valor equipo fin', que es el valor del equipo en el que jugará el futbolista la temporada para la que se negocia el salario. Tiene bastante sentido que esto sea así, ya que es el club el que paga el sueldo. A esta variable le siguen el 'Valor equipo ini' y la 'Edad', tras todas ellas, en un tercer escalón, se encuentran ya el resto de características, todas aproximadamente con la misma importancia. En las figuras 6.16 y 6.17 podemos observar de forma más visual estas diferencias de importancia entre las características. Por un lado, 6.16 muestra el gráfico de las variables más importantes, estas son las que tienen una importancia mayor al 2%. Y por otro, 6.17 enseña los porcentajes de importancia de las características menos importantes.

Posición	Variable	Importancia
1	Valor equipo fin	0.3988
2	Valor equipo ini	0.1842
3	Edad	0.1394
4	Minutos	0.0449
5	Valor liga fin	0.04
6	Valor liga ini	0.024
7	xG	0.0239
8	Dist con balon	0.0202
9	Pases cortos intentados	0.0169
10	Disparos	0.0158
11	Pases largos completados	0.0137
12	Toques	0.0134
13	Conducciones ultimo tercio	0.0128
14	Controles	0.0126
15	Pases clave	0.0103
16	Tiros a puerta	0.0085
17	Titularidades	0.0075
18	Faltas lanzadas	0.0048
19	Posicion_2	0.0045
20	Penaltis lanzados	0.0037

Cuadro 6.16: *Ranking* de importancia de las características en el modelo

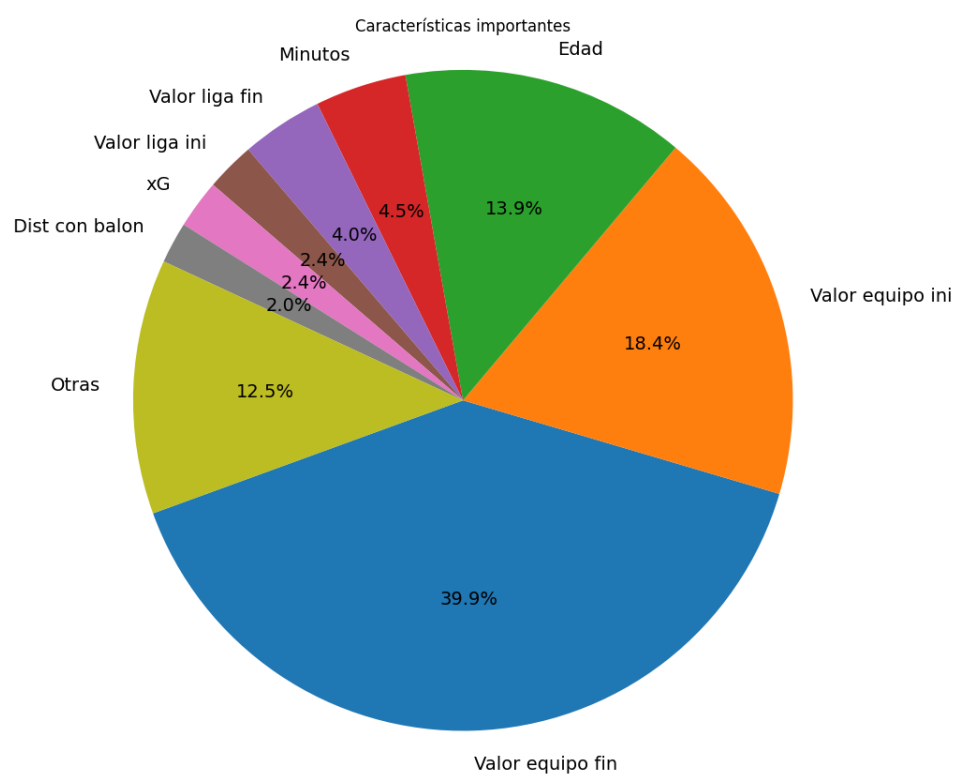


Figura 6.16: Gráfico de pastel de las características con más de un 2% de importancia.



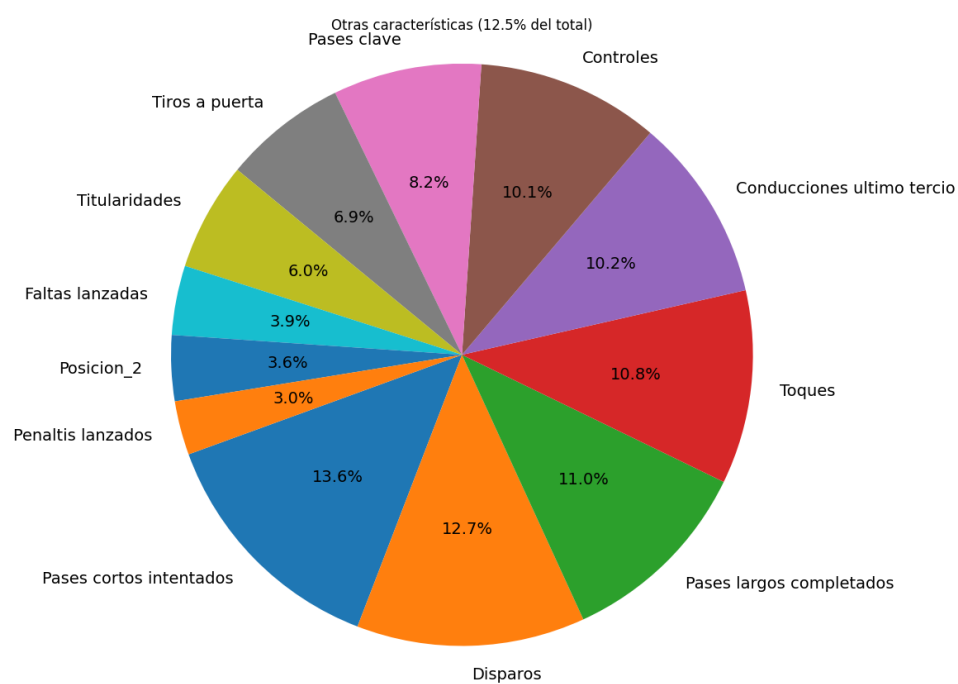


Figura 6.17: Gráfico de pastel de las características con menos de un 2% de importancia.



## Capítulo 7

# Conclusiones y trabajos futuros

En este TFG hemos afrontado el problema del análisis del salario de futbolistas profesionales, utilizando para ello modelos de Aprendizaje Automático. Por otro lado, hemos realizado un selecto análisis de las características del problema mediante el uso de métodos de selección de características. También, debido a la naturaleza del conjunto de datos, hemos utilizado distintas versiones del mismo: sin porteros, solo usando una posición, etc. Finalmente tras obtener y analizar los resultados obtenidos en todas las áreas, llegamos a las siguientes conclusiones:

- Por un lado, utilizar por ejemplo una posición, en lugar del conjunto completo, es más perjudicial por la pérdida de información al no utilizar todo el *dataset*, que beneficioso por reducir la complejidad del conjunto al quedarnos con datos más similares.
- Por otro lado, concluimos en que *Gradient Boosting* es el modelo que mejor desempeño tiene, obteniendo un coeficiente de determinación de 0.7.

Estos resultados obtenidos en nuestro estudio son bastante competitivos, como menciono en el apartado 4.1, demostrando así que es posible utilizar un *dataset* con características reales, en lugar de obtenerlas de la base de datos de algún videojuego. No obstante, estudios como [9] consiguen mejores resultados. Estos trabajos, eso sí, buscan predecir el valor de mercado, no el sueldo, aunque como ya comenté anteriormente considero ambos problemas muy similares. En cuanto a proyectos puramente de predicción de salarios, nuestro trabajo sí que se posiciona entre los que mejores resultados obtiene. Por ejemplo al compararlo con [13], la diferencia de  $R^2$  es de solo 0.04 puntos.

En cuanto a la utilidad del estudio, creo que las conclusiones obtenidas en cuanto a explicabilidad de las predicciones (6.16, 6.17) pueden ser de gran valor para el mundo del fútbol. Es cierto que algunos resultados como que la estadística más importante es el valor del equipo resulta bastante obvia, aunque otras como la edad del futbolista o los minutos que juega, pueden ser de gran interés para clubes y agentes deportivos. Por otro lado, esta explicación de la importancia de las características añade novedades al estado del arte. Aunque ya existían estudios que analizaban las características como [55], éstos no aplicaban una selección inicial de características sino que seleccionaban de forma subjetiva las que creían convenientes. El mejor modelo obtenido como tal también puede ser de gran utilidad, puesto que al embeberlo en la aplicación *web*, cualquier usuario puede utilizarlo fácilmente y sin conocimientos de ML para obtener las estimaciones de salario que necesite.

Echando la vista atrás, podemos decir que hemos cumplido todos los objetivos propuestos del proyecto. Por un lado hemos completado el doble objetivo principal que nos proponíamos en el apartado 1.3. Este objetivo era por un lado obtener un estimador fiable del salario que cobra un futbolista y por otro conseguir una explicación de cómo se llega a dicha estimación. Para llegar a completar este objetivo principal, había también que cumplir una serie de objetivos específicos en los que habíamos desglosado la tarea principal. Tras analizar estos subobjetivos, los cuales también se encuentran en el apartado 1.3, podemos decir que hemos completado todos los puntos propuestos. Adicionalmente hemos desarrollado una pequeña aplicación *web* en la que poder probar el estimador entrenado. Este no era uno de los subobjetivos iniciales, aunque más adelante durante el proyecto nos dimos cuenta de que nos sería de gran utilidad de cara a cumplir de mejor forma el doble objetivo principal antes comentado.

Como trabajos futuros, creo muy conveniente utilizar métodos más sofisticados de *web scrapping* para obtener un conjunto de datos más grande. Tal vez así, modelos más complejos como *Random Forest* generalicen mejor para datos de fuera de la muestra. Como vimos durante nuestra experimentación (figura 6.2), para el conjunto de entrenamiento el modelo obtiene resultados excepcionales. Por otro lado, utilizar algún otro tipo de heurística, como Búsqueda Local [43] para la parte de selección de características puede ser también interesante. Otros algoritmos como los genéticos también tienen un buen desempeño para problemas de selección de características [66]. Por último una idea que se me ocurre también es utilizar información de los futbolistas de varias temporadas. El problema es que en este caso se incrementaría muchísimo la complejidad del estudio, ya que tener variables de varios años implicaría utilizar series temporales a la par que aumenta la dimensionalidad del problema. Por no hablar de que actualmente para tener el máximo número de ejemplos, uso las estadísticas de un mismo futbolista

en distintas temporadas, lo cual no podría hacer en este supuesto proyecto.

Por otro lado, la *app web* también tiene bastantes aspectos a mejorar, debido a que no era una idea inicial del TFG y por tanto solo he desarrollado una versión simplificada. La principal mejora a aplicar sería permitir al usuario solo escribir el nombre del futbolista y la temporada, y que la propia *web* sea capaz de obtener las estadísticas. Éstas se recogerían de las mismas páginas utilizadas para crear el *dataset* usado en el proyecto. También se debería permitir al usuario poder subir un fichero .csv o similar, que contenga los datos de varios jugadores, para poder obtener las predicciones de todos a la vez. Alternativamente, cuando se introdujeran los datos de un jugador y se calculara automáticamente su sueldo, se podrían mostrar jugadores similares, bien en características o bien en sueldo, mostrando el sueldo de éstos. Una mejora estética también sería interesante, para hacer la página más vistosa e intuitiva para el usuario, así como para poder seleccionar que opción de entrada usar.

Por lo que, en resumen, este trabajo obtiene unos resultados altamente competitivos, al compararlo con otros estudios del mismo campo. Este resultado es destacable dado que utilizamos un conjunto de datos poco común en el estado del arte y técnicas de selección de características también poco consideradas. El proyecto incluye también una revisión de las características más importantes a tener en cuenta al predecir un salario, así como una pequeña *app web* donde poder probar el modelo desarrollado. Los puntos principales a mejorar son la obtención de más datos y el uso de algoritmos de selección de características más sofisticados. Por otro lado, en la aplicación se debe mejorar la forma de introducir los datos en general.



# Bibliografía

- [1] Hervé Abdi y Lynne J Williams. «Principal component analysis». En: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), págs. 433-459.
- [2] Yaser S Abu-Mostafa, Malik Magdon-Ismail y Hsuan-Tien Lin. *Learning from data*. Vol. 4. AMLBook New York, 2012.
- [3] Yaser S. Abu-Mostafa, Malik Magdon-Ismail y Hsuan-Tien Lin. *Learning from Data: A Short Course*. Lecture 17. AMLbook, 2012. Cap. 5.
- [4] Amina Adadi y Mohammed Berrada. «Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)». En: *IEEE access* 6 (2018), págs. 52138-52160.
- [5] David Adam. «Science and the World Cup: how big data is transforming football». En: *Nature* 611.7936 (2022), págs. 444-446.
- [6] Luis B Almeida. «Multilayer perceptrons». En: *Handbook of Neural Computation*. CRC Press, 2020, págs. C1-2.
- [7] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [8] Alejandro Barredo Arrieta et al. «Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI». En: *Information fusion* 58 (2020), págs. 82-115.
- [9] Mustafa A Al-Asadi y Sakir Tasdemir. «Predict the value of football players using FIFA video game data and machine learning techniques». En: *IEEE access* 10 (2022), págs. 22631-22645.
- [10] GEAPA Batista, Diego Furtado Silva et al. «How k-nearest neighbor parameters affect its performance». En: *Argentine symposium on artificial intelligence*. Citeseer. 2009, págs. 1-12.
- [11] Roberto Battiti. «Using mutual information for selecting features in supervised neural net learning». En: *IEEE Transactions on neural networks* 5.4 (1994), págs. 537-550.
- [12] BBC. *How does a football transfer work?* URL: <https://www.bbc.com/worklife/article/20170829-how-does-a-football-transfer-work> (visitado 20-04-2024).

- [13] Iman Behravan y Seyed Mohammad Razavi. «A novel machine learning method for estimating football players' value in the transfer market». En: *Soft Computing* 25.3 (2021), págs. 2499-2511.
- [14] Christopher M Bishop. «Pattern recognition and machine learning». En: *Springer google schola* 2 (2006), págs. 645-678.
- [15] Verónica Bolón-Canedo, Noelia Sánchez-Marroño y Amparo Alonso-Betanzos. «A review of feature selection methods on synthetic data». En: *Knowledge and information systems* 34 (2013), págs. 483-519.
- [16] Ross Booth, Robert Brooks y Neil Diamond. *Player salaries and revenues in the Australian football league 2001–2009: Theory and evidence*. 2012.
- [17] Leo Breiman. «Random forests». En: *Machine learning* 45 (2001), págs. 5-32.
- [18] Alessandro Bucciol, Nicolai J Foss y Marco Piovesan. «Pay dispersion and performance in teams». En: *PloS one* 9.11 (2014), e112631.
- [19] Xue-wen Chen y Jong Cheol Jeong. «Enhanced recursive feature elimination». En: *Sixth international conference on machine learning and applications (ICMLA 2007)*. IEEE. 2007, págs. 429-435.
- [20] Aaron Clauset, Cosma Rohilla Shalizi y Mark EJ Newman. «Power-law distributions in empirical data». En: *SIAM review* 51.4 (2009), págs. 661-703.
- [21] Adele Cutler, D Richard Cutler y John R Stevens. «Random forests». En: *Ensemble machine learning: Methods and applications* (2012), págs. 157-175.
- [22] Deloitte. «Annual Review of Football Finance 2023». En: (2023), pág. 8. URL: <https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/sports-business-group/deloitte-uk-annual-review-of-football-finance-2023.pdf>.
- [23] Alin Dobra. «Classification and regression tree construction». En: *Retrieved September 18* (2002), pág. 2011.
- [24] Finale Doshi-Velez y Been Kim. «Towards a rigorous science of interpretable machine learning». En: *arXiv preprint arXiv:1702.08608* (2017).
- [25] United Nations Office on Drugs y Crime. «Corruption and abuse in sport». En: (2021), pág. 18. URL: [https://www.unodc.org/res/safeguardingsport/grcs/section-7\\_html/SPORTS\\_CORRUPTION\\_2021\\_S7.pdf](https://www.unodc.org/res/safeguardingsport/grcs/section-7_html/SPORTS_CORRUPTION_2021_S7.pdf).



- [26] Union of European Football Associations (UEFA). *UEFA Club Licensing and Financial Fair Play Regulations, Edition 2012*. URL: [https://www.uefa.com/MultimediaFiles/Download/Tech/uefaorg/General/01/80/54/10/1805410\\_DOWNLOAD.pdf](https://www.uefa.com/MultimediaFiles/Download/Tech/uefaorg/General/01/80/54/10/1805410_DOWNLOAD.pdf) (visitado 10-04-2024).
- [27] Artur J Ferreira y Mário AT Figueiredo. «An unsupervised approach to feature discretization and selection». En: *Pattern Recognition* 45.9 (2012), págs. 3048-3060.
- [28] Jerome H Friedman. «Greedy function approximation: a gradient boosting machine». En: *Annals of statistics* (2001), págs. 1189-1232.
- [29] Jerome H Friedman. «Stochastic gradient boosting». En: *Computational statistics & data analysis* 38.4 (2002), págs. 367-378.
- [30] Tadayoshi Fushiki. «Estimation of prediction error by using K-fold cross-validation». En: *Statistics and Computing* 21 (2011), págs. 137-146.
- [31] Liga Nacional de Fútbol Profesional. *La industria del fútbol profesional genera 185.000 empleos, 4.100 M€ en impuestos y una facturación equivalente al 1,37% del PIB en España*. URL: <https://newsletter.laliga.es/futbol-global/la-industria-del-futbol-profesional-genera-185-000-empleos-4-100-me-en-impuestos-y-una-facturacion-equivalente-al-137-del-pib-en-espana-1> (visitado 08-04-2024).
- [32] Fxtop. *Cálculo de la inflación y de la evolución de los precios entre 2 fechas*. URL: <https://fxtop.com/es/calculadora-de-inflacion.php#presentation> (visitado 09-05-2024).
- [33] Leilani H Gilpin et al. «Explaining explanations: An overview of interpretability of machine learning». En: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE. 2018, págs. 80-89.
- [34] A. Gohritz, G. Hovemann y P. Ehnold. «Opportunistic behaviour of players' agents in football and its monitoring by the players—an empirical analysis from the perspective of the players». En: *German Journal of Exercise and Sport Research* 53.3 (2023), págs. 275-287.
- [35] Michel L Goldstein, Steven A Morris y Gary G Yen. «Problems with fitting to the power-law distribution». En: *The European Physical Journal B-Condensed Matter and Complex Systems* 41 (2004), págs. 255-258.
- [36] Jürgen Groß. *Linear regression*. Vol. 175. Springer Science & Business Media, 2003.
- [37] Riccardo Guidotti et al. «A survey of methods for explaining black box models». En: *ACM computing surveys (CSUR)* 51.5 (2018), págs. 1-42.

- [38] Isabelle Guyon y André Elisseeff. «An introduction to variable and feature selection». En: *Journal of machine learning research* 3.Mar (2003), págs. 1157-1182.
- [39] Tugbay Inan y Levent Cavas. «Estimation of market values of football players through artificial neural network: a model study from the turkish super league». En: *Applied Artificial Intelligence* 35.13 (2021), págs. 1022-1042.
- [40] Snowflake Inc. *Streamlit documentation*. URL: <https://docs.streamlit.io/> (visitado 03-06-2024).
- [41] Infobae. *Sin representante, Kevin De Bruyne utilizó un novedoso método para firmar una renovación millonaria con el Manchester City*. URL: <https://www.infobae.com/america/deportes/2021/04/08/sin-representante-kevin-de-bruyne-utilizo-un-novedoso-metodo-para-firmar-una-renovacion-millonaria-con-el-manchester-city/> (visitado 09-04-2024).
- [42] Jooble. *¿Cuánto gana Investigador senior en España?* URL: <https://es.jooble.org/salary/investigador-senior#:~:text=%C2%BFCu%C3%A1nto%20gana%20Investigador%20senior%20en,%2030%20C07%20%E2%82%AC%20por%20hora.> (visitado 25-04-2024).
- [43] Md Monirul Kabir, Md Shahjahan y Kazuyuki Murase. «A new local search based hybrid genetic algorithm for feature selection». En: *Neurocomputing* 74.17 (2011), págs. 2914-2928.
- [44] Moaiad Ahmad Khder. «Web scraping or web crawling: State of art, techniques, approaches and application.» En: *International Journal of Advances in Soft Computing & Its Applications* 13.3 (2021).
- [45] Ron Kohavi y George H John. «Wrappers for feature subset selection». En: *Artificial intelligence* 97.1-2 (1997), págs. 273-324.
- [46] Oliver Kramer y Oliver Kramer. «K-nearest neighbors». En: *Dimensionality reduction with unsupervised nearest neighbors* (2013), págs. 13-23.
- [47] Vipin Kumar y Sonajharia Minz. «Feature selection». En: *SmartCR* 4.3 (2014), págs. 211-229.
- [48] Scikit Learn. *Effect of transforming the targets in regression model*. URL: [https://scikit-learn.org/stable/auto\\_examples/compose/plot\\_transformed\\_target.html](https://scikit-learn.org/stable/auto_examples/compose/plot_transformed_target.html) (visitado 11-05-2024).
- [49] Scikit Learn. *sklearn.ensemble.GradientBoostingRegressor*. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html> (visitado 18-05-2024).
- [50] Scikit Learn. *sklearn.ensemble.RandomForestRegressor*. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html> (visitado 18-05-2024).

- [51] Scikit Learn. *sklearn.feature\_selection.SelectKBest*. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.SelectKBest.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html) (visitado 19-05-2024).
- [52] Scikit Learn. *sklearn.linear\_model.SGDRegressor*. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.SGDRegressor.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html) (visitado 18-05-2024).
- [53] Scikit Learn. *sklearn.neighbors.KNeighborsRegressor*. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html> (visitado 17-05-2024).
- [54] Scikit Learn. *sklearn.neural\_network.MLPRegressor*. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPRegressor.html](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html) (visitado 18-05-2024).
- [55] Chenyao Li, Stylianos Kampakis y Philip Treleaven. «Machine learning modeling to evaluate the value of football players». En: *arXiv preprint arXiv:2207.11361* (2022).
- [56] Hongyi Li, Chunhai Cui y Shuai Jiang. «Strategy for improving the football teaching quality by AI and metaverse-empowered in mobile internet environment». En: *Wireless Networks* (2022), págs. 1-10.
- [57] Zachary C Lipton. «The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.» En: *Queue* 16.3 (2018), págs. 31-57.
- [58] Huan Liu, Rudy Setiono et al. «A probabilistic approach to feature selection-a filter solution». En: *ICML*. Vol. 96. 1996, págs. 319-327.
- [59] Gilles Louppe. «Understanding random forests». En: *Cornell University Library* 10 (2014).
- [60] Daily Mail. *Man City 'secretly paid Roberto Mancini over double his contracted wage of £1.45m by funnelling extra £1.75m a year through Sheik Mansour's football club in Arabian Gulf League'*. URL: <https://www.dailymail.co.uk/sport/sportsnews/article-6366745/Man-City-secretly-paid-Roberto-Mancini-double-contracted-wage.html> (visitado 10-04-2024).
- [61] Dhendra Marutho, Sunarna Hendra Handaka, Ekaprana Wijaya et al. «The determination of cluster number at k-mean using elbow method and purity evaluation on headline news». En: *2018 international seminar on application for technology of information and communication*. IEEE. 2018, págs. 533-538.
- [62] Tomonori Masui. *All You Need to Know about Gradient Boosting Algorithm. Part 1. Regression*. URL: <https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502> (visitado 29-04-2024).

- [63] Ian G McHale y Benjamin Holmes. «Estimating transfer fees of professional footballers using advanced performance metrics and machine learning». En: *European Journal of Operational Research* 306.1 (2023), págs. 389-399.
- [64] Tom M Mitchell. «Does machine learning really work?». En: *AI magazine* 18.3 (1997), págs. 11-11.
- [65] NumFOCUS. *Pandas site*. URL: <https://pandas.pydata.org/> (visitado 25-04-2024).
- [66] Il-Seok Oh, Jin-Seon Lee y Byung-Ro Moon. «Hybrid genetic algorithms for feature selection». En: *IEEE Transactions on pattern analysis and machine intelligence* 26.11 (2004), págs. 1424-1437.
- [67] Emirhan Özbalta, Mücahit Yavuz y Tolga Kaya. «National Basketball Association Player Salary Prediction Using Supervised Machine Learning Methods». En: *Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation: Proceedings of the INFUS 2021 Conference, held August 24-26, 2021. Volume 2*. Springer. 2022, págs. 189-196.
- [68] PCcomponentes. *ASUS Dual GeForce RTX 4060 EVO OC Edition 8GB GDDR6 DLSS3*. URL: <https://www.pccomponentes.com/asus-dual-geforce-rtx-4060-evo-oc-edition-8gb-gddr6-dlss3> (visitado 25-04-2024).
- [69] PCcomponentes. *Corsair Vengeance LPX DDR4 3200MHz PC4-25600 16GB CL16 Negro*. URL: <https://www.pccomponentes.com/corsair-vengeance-lpx-ddr4-3200mhz-pc4-25600-16gb-cl16-negro> (visitado 25-04-2024).
- [70] PCcomponentes. *Intel Core i5-10400F 2.90 GHz*. URL: [https://www.pccomponentes.com/intel-core-i5-10400f-290-ghz?campaigntype=eshopping&campaignchannel=shopping&gad\\_source=1&gclid=Cj0KCQjw\\_qexBhCoARIsAFgBleuXttgvzkgI7PWBEUPjArr1GxH7Eq-8jR43yEJkje61NB7gCuLo-YaAirwEALw\\_wcB](https://www.pccomponentes.com/intel-core-i5-10400f-290-ghz?campaigntype=eshopping&campaignchannel=shopping&gad_source=1&gclid=Cj0KCQjw_qexBhCoARIsAFgBleuXttgvzkgI7PWBEUPjArr1GxH7Eq-8jR43yEJkje61NB7gCuLo-YaAirwEALw_wcB) (visitado 25-04-2024).
- [71] PCcomponentes. *Kioxia Exceria G2 Unidad SSD 500GB NVMe M.2 2280*. URL: <https://www.pccomponentes.com/kioxia-exceria-g2-unidad-ssd-500gb-nvme-m2-2280> (visitado 25-04-2024).
- [72] Roxana Pérez-Rubido. «Una revisión a algoritmos de selección de atributos que tratan la redundancia en datos microarreglos». En: *Revista Cubana de Ciencias Informáticas* 7.4 (2013), págs. 16-30.
- [73] ProSport. *£20 a week to £3 million a year: how footballers' wages have changed over the last 150 years*. URL: <https://prosportwealth.co.uk/20-a-week-to-3-million-a-year-how-footballers-wages-have-changed-over-the-last-150-years/> (visitado 02-05-2024).

- [74] Anita Rácz, Dávid Bajusz y Károly Héberger. «Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification». En: *Molecules* 26.4 (2021), pág. 1111.
- [75] Keshav Rathi et al. «Applications of artificial intelligence in the game of football: The global perspective». En: *Researchers World* 11.2 (2020), págs. 18-29.
- [76] Leonard Richardson. *Beautiful Soup Documentation*. URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> (visitado 25-04-2024).
- [77] Raul Rojas y Raúl Rojas. «The backpropagation algorithm». En: *Neural networks: a systematic introduction* (1996), págs. 149-182.
- [78] Stuart J. Russell y Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.
- [79] Ismael Gómez Schmidt. *Analítica de fútbol: Crea tu propio modelo xG*. URL: <https://medium.com/datos-y-ciencia/anal%C3%ADtica-de-f%C3%BAtbol-crea-tu-propio-modelo-xg-a14d493c1687> (visitado 11-04-2024).
- [80] Patrick Schober, Christa Boer y Lothar A Schwarte. «Correlation coefficients: appropriate use and interpretation». En: *Anesthesia & analgesia* 126.5 (2018), págs. 1763-1768.
- [81] scikit-learn. *3.3. Metrics and scoring: quantifying the quality of predictions*. URL: [https://scikit-learn.org/stable/modules/model\\_evaluation.html#regression-metrics](https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics) (visitado 29-04-2024).
- [82] scikit-learn. *sklearn.feature\_selection.f\_regression*. URL: [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.f\\_regression.html#sklearn.feature\\_selection.f\\_regression](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html#sklearn.feature_selection.f_regression) (visitado 02-05-2024).
- [83] Philip Sedgwick. «Pearson's correlation coefficient». En: *Bmj* 345 (2012).
- [84] Prabhnoor Singh y Puneet Singh Lamba. «Influence of crowdsourcing, popularity and previous year statistics in market value estimation of football players». En: *Journal of Discrete Mathematical Sciences and Cryptography* 22.2 (2019), págs. 113-126.
- [85] David J Smyth y Seamus J Smyth. «Major league baseball division standings, sports journalists' predictions and player salaries». En: *Managerial and decision economics* 15.5 (1994), págs. 421-429.
- [86] SPORTbible. *The Swap Deal Involving Miralem Pjanic And Arthur Melo Was 'Illegal'*. URL: <https://www.sportbible.com/football/news-the-swap-deal-involving-miralem-pjanic-and-arthur-melo-was-illegal-20211119> (visitado 10-04-2024).

- [87] EA Sports. *Página web de la saga FIFA*. URL: <https://www.ea.com/es-es/games/fifa> (visitado 12-04-2024).
- [88] Rade Stanojevic y Laszlo Gyarmati. «Towards data-driven football player assessment». En: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2016, págs. 167-172.
- [89] Statista. *Revenue of the Big Five soccer leagues in Europe from 2012/13 to 2021/22, with a forecast to 2023/24, by league*. URL: <https://www.statista.com/statistics/261218/big-five-european-soccer-leagues-revenue/> (visitado 10-05-2024).
- [90] NumPy team. *NumPy site*. URL: <https://numpy.org/> (visitado 25-04-2024).
- [91] Scikit-learn team. *Scikit-learn site*. URL: <https://scikit-learn.org/stable/> (visitado 25-04-2024).
- [92] The Matplotlib development team. *Matplotlib site*. URL: <https://matplotlib.org/> (visitado 25-04-2024).
- [93] Giorgio Valentini y Francesco Masulli. «Ensembles of learning machines». En: *Neural Nets: 13th Italian Workshop on Neural Nets, WIRN VIETRI 2002 Vietri sul Mare, Italy, May 30–June 1, 2002 Revised Papers 13*. Springer. 2002, págs. 3-20.
- [94] Laurens Van der Maaten y Geoffrey Hinton. «Visualizing data using t-SNE.» En: *Journal of machine learning research* 9.11 (2008).
- [95] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik et al. «Dimensionality reduction: a comparative». En: *J Mach Learn Res* 10.66-71 (2009).
- [96] Michael Waskom. *Seaborn site*. URL: <https://seaborn.pydata.org/> (visitado 25-04-2024).
- [97] Eric W. Weisstein. *Correlation Coefficient*. URL: <https://mathworld.wolfram.com/CorrelationCoefficient.html> (visitado 01-05-2024).
- [98] Halbert White. «A reality check for data snooping». En: *Econometrica* 68.5 (2000), págs. 1097-1126.
- [99] Svante Wold, Kim Esbensen y Paul Geladi. «Principal component analysis». En: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), págs. 37-52.
- [100] L Yaldo y Lior Shamir. «Computational estimation of football player wages». En: *International Journal of Computer Science in Sport* 16.1 (2017), págs. 18-38.
- [101] Ahmet Talha Yiğit, Barış Samak y Tolga Kaya. «Football player value assessment using machine learning techniques». En: *Intelligent and Fuzzy Techniques in Big Data Analytics and Decision Making (INFUS) Conference*. 2020, págs. 289-297.

## Apéndice A

# Características del dataset

En este apéndice se explica el significado de cada característica del *dataset*:

- **Jugador:** Nombre del futbolista.
- **Edad:** Edad del jugador.
- **Temporada:** Año en el que empieza la temporada a la que pertenecen las estadísticas. Por ejemplo si 'Temporada' vale 2015, las estadísticas pertenecen a la temporada 2015/2016.
- **Posicion:** Posición en la que juega el futbolista, para jugadores que pueden jugar en varias se guarda todas. Esta variable se codifica en cuatro variables binarias, una por posición. 'Posicion\_0' para indicar si el futbolista juega de portero. 'Posicion\_1' para defensa. 'Posicion\_2' para centrocampista. 'Posicion\_3' para delantero.
- **Valor liga ini:** Beneficio de la liga en la que el jugador empieza la temporada, el beneficio se corresponde con el de la actual temporada. Las unidades están en millones.
- **Partidos:** Número de partidos jugados.
- **Titularidades:** Número de partidos en los que fue titular.
- **Minutos:** Minutos totales jugados.
- **Goles:** Goles marcados.
- **Asistencias:** Asistencias dadas.
- **Goles penalti:** Goles de penalti.

- **Penaltis lanzados:** Total penaltis intentados, incluye marcados y fallados.
- **Amarillas:** Veces amonestado con tarjeta amarilla.
- **Rojas:** Veces amonestado con tarjeta roja.
- **xG:** Valor de los goles que el jugador debería haber marcado en realidad (*expected goals*).
- **xG sin penaltis:** *Expected goals* sin contar penaltis.
- **xAG:** Porcentaje de gol esperado que dan las asistencias de el futbolista.
- **Conducciones progresivas:** Conducciones hacia adelante.
- **Pases progresivos:** Pases hacia adelante.
- **Pases progresivos recibidos:** Pases progresivos recibidos.
- **Goles encajados:** Goles que ha encajado un portero.
- **Disparos recibidos:** Disparos a puerta que recibe un portero.
- **Salvadas:** Paradas del portero.
- **Porterías a cero:** Porterías a cero del portero.
- **Penaltis en contra:** Penaltis lanzados en contra a un portero.
- **Penaltis encajados:** Goles de penalti que un portero encaja.
- **Penaltis detenidos:** Penaltis que para el portero.
- **Penaltis fallados (en contra):** Penaltis que van fuera o al palo siendo portero.
- **2a amarilla:** Expulsiones por segunda amarilla.
- **Faltas cometidas:** Faltas cometidas.
- **Faltas recibidas:** Faltas recibidas.
- **Fueras de juego:** Fuera de juego.
- **Balones centrados:** Centros intentados.
- **Intercepciones:** Pases cortados por el jugador.
- **Entradas ganadas:** Acciones en las que el jugador se lanza al suelo para robar el balón al rival conseguidas con éxito.



- **Penaltis concedidos:** Penaltis provocados.
- **Goles en propia:** Goles marcados en propia.
- **Recuperaciones:** Balones recuperados por el jugador.
- **Aereos ganados:** Duelos aéreos ganados.
- **Aereos perdidos:** Duelos aéreos perdidos.
- **Disparos:** Total de disparos efectuados por el jugador.
- **Tiros a puerta:** Disparos que van entre los tres palos.
- **Distancia tiros:** Distancia media de los tiros en yardas.
- **Faltas lanzadas:** Faltas lanzadas por el jugador.
- **Pases completados:** Pases que llegan a un compañero.
- **Pases intentados:** Pases intentados, cuenta los pases completados y los fallados.
- **Distancia pases:** Distancia total de los pases dados en yardas.
- **Dist progresiva pases:** Distancia total en dirección a campo rival de los pases dados, en yardas.
- **Pases cortos completados:** Pases entre 5 y 15 yardas completados.
- **Pases cortos intentados:** Pases entre 5 y 15 yardas intentados.
- **Pases medios completados:** Pases entre 15 y 30 yardas completados.
- **Pases medios intentados:** Pases entre 15 y 30 yardas intentados.
- **Pases largos completados:** Pases de más de 30 yardas completados.
- **Pases largos intentados:** Pases de más de 30 yardas intentados.
- **Pases clave:** Pases que conducen a un disparo dados.
- **Pases ultimo tercio:** Pases dados que llegan al último tercio del campo.
- **Pases al area:** Pases completados al área de penalti.
- **Centros al area:** Centros completados al área de penalti.
- **Toques:** Número de veces que el jugador toca el balón.

- **Regates intentados:** Regates intentados.
- **Regates exitosos:** Regates completados exitosamente.
- **Controles:** Veces que el jugador controló el balón.
- **Dist con balon:** Distancia total de los desplazamientos del jugador con balón en yardas.
- **Dist progresiva:** Distancia total de las conducciones en dirección a campo rival del jugador en yardas.
- **Conducciones ultimo tercio:** Conducciones que entran en el último tercio del campo.
- **Conducciones en area:** Conducciones que entran en el área rival.
- **Errores de control:** Errores del futbolista al controlar el balón que acaban en una pérdida del mismo.
- **Salario semanal:** Variable a predecir, salario semanal en euros que cobra el futbolista.
- **Valor liga fin:** Beneficio total de la liga en la que el jugador milita al final de temporada, es decir si ficha por un equipo de otra liga en verano es esa la liga que aparece, el beneficio se corresponde con el de la actual temporada. Las unidades están en millones.
- **Posicion liga ini:** Posición en liga en la que queda el jugador con su equipo en la presente temporada.
- **Posicion liga fin:** Posición en liga en la que queda el equipo (en la presente temporada) en el que jugará el jugador la próxima temporada.
- **Valor equipo ini:** Valor del equipo por el que juega el jugador en la presente temporada, el valor es el de la temporada actual. Las unidades están en millones.
- **Valor equipo fin:** Valor del equipo en el que jugará el jugador en la siguiente campaña, el valor es el de la presente temporada. Las unidades están en millones.

## Apéndice B

# Resumen de las variables del problema

En este apéndice muestra un resumen de las características del problema, tanto las continuas, como las categóricas:

Cuadro B.1: Resumen de las variables categóricas

Variable	Número Ejemplos	Rango Valores	Valores Faltantes
Posicion_1	9168	False - True	0
Posicion_2	9168	False - True	0
Posicion_3	9168	False - True	0
Posicion_4	9168	False - True	0

Cuadro B.2: Resumen de las variables continuas. Parte 1

Variable	Media	Mediana	Desviación Típica
Edad	25.33	25.00	4.26
Valor liga ini	3328.37	3114.00	1439.62
Partidos	22.92	25.00	10.41
Titularidades	18.07	18.00	10.96
Minutos	1613.52	1629.00	941.39
Goles	2.36	1.00	3.86
Asistencias	1.63	1.00	2.28
Goles penalti	0.22	0.00	0.87
Penaltis lanzados	0.28	0.00	1.02
Amarillas	3.24	3.00	2.87
Rojas	0.15	0.00	0.42
xG	2.33	1.10	3.43
xG sin penaltis	2.11	1.10	2.96
xAG	1.64	1.00	1.98
Conducciones progresivas	30.22	20.00	32.26
Pases progresivos	64.54	52.00	58.01
Pases progresivos recibidos	63.68	36.00	73.75
Goles encajados	2.23	0.00	9.32
Disparos recibidos	7.02	0.00	29.19
Salvadas	4.80	0.00	20.11
Porterias a cero	0.47	0.00	2.12
Penaltis en contra	0.27	0.00	1.24
Penaltis encajados	0.21	0.00	1.00
Penaltis detenidos	0.04	0.00	0.28
Penaltis fallados (en contra)	0.01	0.00	0.14
2a amarilla	0.07	0.00	0.26
Faltas cometidas	19.96	18.00	15.44
Faltas recibidas	19.14	15.00	17.46
Fueras de juego	3.03	1.00	5.52
Balones centrados	30.01	11.00	45.03
Intercepciones	17.32	13.00	16.67
Entradas ganadas	16.19	13.00	14.10
Penaltis concedidos	0.26	0.00	0.55
Goles en propia	0.07	0.00	0.27
Recuperaciones	87.21	77.00	63.71
Aereos ganados	25.79	16.00	29.01
Aereos perdidos	25.25	20.00	25.08
Disparos	20.60	13.00	22.89

Cuadro B.3: Resumen de las variables continuas. Parte 2

Variable	Media	Mediana	Desviación Típica
Tiros a puerta	6.97	4.00	9.10
Distancia tiros	16.34	16.90	8.96
Faltas lanzadas	0.84	0.00	2.52
Pases completados	649.86	560.50	499.73
Pases intentados	823.26	745.00	590.87
Distancia pases	11602.08	9460.00	9451.29
Dist progresiva pases	4256.36	2954.00	4336.66
Pases cortos completados	291.32	246.50	231.06
Pases cortos intentados	332.04	289.00	254.15
Pases medios completados	264.86	205.00	236.10
Pases medios intentados	311.39	253.00	259.20
Pases largos completados	73.09	48.00	78.13
Pases largos intentados	130.93	89.00	146.14
Pases clave	15.49	10.00	16.95
Pases ultimo tercio	49.67	36.00	48.02
Pases al area	12.86	8.00	14.74
Centros al area	3.41	1.00	5.33
Toques	1000.55	939.00	675.15
Regates intentados	27.31	17.00	30.98
Regates exitosos	15.29	10.00	17.54
Controles	566.16	515.00	406.46
Dist con balon	3047.13	2710.50	2247.29
Dist progresiva	1580.81	1358.00	1255.56
Conducciones ultimo tercio	22.57	16.00	23.32
Conducciones en area	7.21	2.00	11.80
Errores de control	24.30	18.00	23.28
Valor liga fin	3329.66	3114.00	1436.65
Posicion liga ini	9.19	9.00	5.21
Posicion liga fin	8.87	9.00	4.91
Valor equipo ini	325.87	234.25	265.06
Valor equipo fin	329.58	237.33	263.47

Cuadro B.4: Resumen de las variables continuas. Parte 3

Variable	Máximo	Mínimo	Valores Faltantes
Edad	41.0	15.0	0.0
Valor liga ini	6605.0	1598.0	0.0
Partidos	38.0	1.0	0.0
Titularidades	38.0	0.0	0.0
Minutos	3420.0	1.0	0.0
Goles	41.0	0.0	0.0
Asistencias	21.0	0.0	0.0
Goles penalti	14.0	0.0	0.0
Penaltis lanzados	15.0	0.0	0.0
Amarillas	17.0	0.0	0.0
Rojas	4.0	0.0	0.0
xG	33.2	0.0	0.0
xG sin penaltis	29.3	0.0	0.0
xAG	20.0	0.0	0.0
Conducciones progresivas	276.0	0.0	0.0
Pases progresivos	424.0	0.0	0.0
Pases progresivos recibidos	674.0	0.0	0.0
Goles encajados	91.0	0.0	0.0
Disparos recibidos	233.0	0.0	0.0
Salvadas	162.0	0.0	0.0
Porterias a cero	26.0	0.0	0.0
Penaltis en contra	14.0	0.0	0.0
Penaltis encajados	13.0	0.0	0.0
Penaltis detenidos	4.0	0.0	0.0
Penaltis fallados (en contra)	3.0	0.0	0.0
2a amarilla	2.0	0.0	0.0
Faltas cometidas	93.0	0.0	0.0
Faltas recibidas	167.0	0.0	0.0
Fueras de juego	56.0	0.0	0.0
Balones centrados	393.0	0.0	0.0
Intercepciones	112.0	0.0	0.0
Entradas ganadas	98.0	0.0	0.0
Penaltis concedidos	5.0	0.0	0.0
Goles en propia	4.0	0.0	0.0
Recuperaciones	345.0	0.0	0.0
Aereos ganados	307.0	0.0	0.0
Aereos perdidos	234.0	0.0	0.0
Disparos	193.0	0.0	0.0

Cuadro B.5: Resumen de las variables continuas. Parte 4

Variable	Máximo	Mínimo	Valores Faltantes
Tiros a puerta	91.0	0.0	0.0
Distancia tiros	82.9	0.0	0.0
Faltas lanzadas	48.0	0.0	0.0
Pases completados	2919.0	0.0	0.0
Pases intentados	3365.0	0.0	0.0
Distancia pases	58361.0	0.0	0.0
Dist progresiva pases	36334.0	0.0	0.0
Pases cortos completados	1611.0	0.0	0.0
Pases cortos intentados	1732.0	0.0	0.0
Pases medios completados	1638.0	0.0	0.0
Pases medios intentados	1740.0	0.0	0.0
Pases largos completados	677.0	0.0	0.0
Pases largos intentados	1348.0	0.0	0.0
Pases clave	136.0	0.0	0.0
Pases ultimo tercio	358.0	0.0	0.0
Pases al area	140.0	0.0	0.0
Centros al area	57.0	0.0	0.0
Toques	3769.0	0.0	0.0
Regates intentados	306.0	0.0	0.0
Regates exitosos	185.0	0.0	0.0
Controles	2510.0	0.0	0.0
Dist con balon	15364.0	0.0	0.0
Dist progresiva	10671.0	0.0	0.0
Conducciones ultimo tercio	222.0	0.0	0.0
Conducciones en area	140.0	0.0	0.0
Errores de control	149.0	0.0	0.0
Valor liga fin	6605.0	1598.0	0.0
Posicion liga ini	20.0	1.0	0.0
Posicion liga fin	18.0	1.0	0.0
Valor equipo ini	1200.0	28.9	0.0
Valor equipo fin	1200.0	41.2	0.0





