



**Universidad
Europea**

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

TRABAJO FIN DE MÁSTER

**Aplicación de Machine Learning e IA
Generativa para la predicción y consulta
inteligente sobre datos de jugadores de
fútbol**

SERGIO PÉREZ-BAQUERO ROBISCO

Dirigido por

ALEJANDRO VICENTE GONZÁLEZ

YUDITH CARDINALE

CURSO 2024 - 2025

Índice general

1. Introducción	7
1.1. Aplicación práctica: Análisis de datos deportivos	7
1.2. Motivación y necesidad de la plataforma	8
1.3. Objetivos	8
1.3.1. Objetivos principales	9
1.3.2. Objetivos específicos	9
2. Estado del arte	10
2.1. Predicción del rendimiento	10
2.2. Consulta inteligente de la información	10
3. Marco teórico	13
3.1. Aprendizaje automático: conceptos generales	13
3.1.1. Aprendizaje supervisado	13
3.1.2. Aprendizaje no supervisado	14
3.1.3. Aprendizaje por refuerzo	15
3.1.4. Generalización, división de los datos y complejidad	16
3.1.5. Ajuste de hiperparámetros	19
3.2. Modelos de aprendizaje automático	19
3.2.1. Redes neuronales	19
3.2.2. Algoritmo de random forest	25
3.2.3. Regresión logística	26
3.3. Métricas	30
3.3.1. Métricas de regresión	30
3.3.2. Métricas de clasificación	30
4. Metodología	34
4.1. Extracción de los datos.	34
4.2. Análisis exploratorio de los datos.	34
4.3. Diseño de la arquitectura del sistema	36
4.4. Desarrollo e implementación	38
4.4.1. Procesamiento de datos	38
4.4.2. Fuga de datos: Solución	41
4.4.3. División de datos: entrenamiento, validación y test	42
4.4.4. Herramientas	43
4.5. Cronograma	43
4.5.1. Planificación Scrum	45
4.5.2. Planificación en ola	46
4.6. Presupuesto	46
5. Resultados y discusión	49
5.1. Resultados y discusión bloque de IA Generativa	49

5.2. Resultados y discusión del bloque de predicciones.	54
5.2.1. Resultados predicción titularidad	54
5.2.2. Resultados predicción rendimiento	57
5.3. Interfaz gráfica: Resultado final	59
6. Conclusiones y trabajo futuro	61
6.1. Conclusiones	61
6.2. Trabajo futuro	62
Bibliografía	68
7. Apéndice I: Gráficas complementarias.	69

Índice de Figuras

2.1. Arquitectura RAG diseñada. Fuente: Elaboración propia	11
3.1. Estructura en el aprendizaje por refuerzo. Fuente:[31]	15
3.2. Ejemplo de subajuste, buen ajuste y sobreajuste. Fuente:[54]	17
3.3. Ejemplo de red neuronal con 2 capas ocultas. Fuente:[37]	20
3.4. Comportamiento y funcionamiento de una neurona. Fuente:[35]	21
3.5. Ejemplo de arquitectura de perceptrón multicapa. Fuente: [53]	21
3.6. Cortes en un árbol de decisión y partición en regiones del espacio de caracte- rísticas. Fuente: [11]	23
3.7. Ejemplo de arquitectura y funcionamiento de <i>random forest</i> con n árboles de decisión. Fuente: [24]	26
3.8. Función Sigmoide	27
3.9. Ejemplo de matriz de confusión.	31
3.10. Ejemplo de curva ROC y su correspondiente valor AUC.	33
4.1. Arquitectura y flujo de predicción del rendimiento de jugadores. Fuente: Elabo- ración propia	37
4.2. Componentes principales de la arquitectura del segundo gran bloque. Fuente: Elaboración propia.	37
4.3. Flujo de herramientas proporcionadas al agente. Fuente: Elaboración propia. .	39
4.4. División de datos en entrenamiento, validación y test.	42
4.5. Estructura de desglose de trabajo o EDT. Fuente: Elaboración propia realizada con herramienta Draw.io	44
4.6. Diagrama ciclo Scrum del proyecto. Fuente: Elaboración propia con herramien- ta Draw.io	45
5.1. Interfaz portada principal	59
5.2. Funcionalidades de las variables en la interfaz	60
5.3. Consulta al agente conversacional	60
7.1. Gráfico de la importancia de variables en modelo de <i>random forest</i> en la pre- dicción de titularidades.	69
7.2. Diagrama de Gantt del proyecto. Fuente: Elaboración propia.	70
7.3. Matriz de correlación de variables numéricas	71

Índice de Tablas

4.1. Variables del dataset de partidos temporada 2023/24	34
4.2. Variables del dataset de jugadores temporada 2023/24	35
4.3. Ejemplo de un jugador tras técnica de One Hot Encoding	40
4.4. Ejemplo de Label Encoder de varios jugadores	40
4.5. Ejemplo de Label Encoder para fecha de un partido	40
4.6. Ejemplo conjunto de datos a nivel jugador	41
4.7. Ejemplo conjunto de datos a nivel equipo	42
4.8. Conjunto de datos general para los modelos.	42
4.9. Coste de tareas realizadas en el proyecto	47
4.10. Costes totales del proyecto	48
5.1. Valores de hiperparámetros posibles tras búsqueda en cuadrícula en modelos de clasificación	55
5.2. Hiperparámetros óptimos tras búsqueda en cuadrícula en modelos de clasifi- cación	55
5.3. Resultados en conjunto de validación de cada técnica con sus métricas	56
5.4. Resultados en conjunto de test de cada técnica con sus métricas	56
5.5. Espacio de búsqueda de hiperparámetros considerado en la optimización de cada modelo	57
5.6. Hiperparámetros óptimos tras búsqueda en cuadrícula en modelos de regresión	58
5.7. Resultados de los modelos de regresión en el conjunto de test	58

RESUMEN

El proyecto busca facilitar a usuarios no técnicos en programación el acceso a análisis avanzados de datos futbolísticos mediante una plataforma interactiva que se compone de dos bloques principales. La solución integra un agente conversacional para responder consultas sobre cálculos de estadísticas de equipos y jugadores de la liga española, junto con modelos de predicción personalizados, ofreciendo una herramienta accesible y de gran valor para el análisis deportivo.

En primer lugar, el primer bloque consiste en la revisión de técnicas de modelos de aprendizaje automático para ser aplicados en la predicción del rendimiento de jugadores. Para abordar este bloque, ha sido necesario completar un proceso ETL, que se basa en la extracción, procesamiento y carga de datos, seguido de la división de los datos en entrenamiento, validación y *test* con el objetivo de obtener los hiperparámetros óptimos de los modelos y poder entrenar con una gran cantidad de observaciones para lograr valores de métricas aceptables en el conjunto de *test*.

En segundo lugar, el segundo bloque consiste en el desarrollo de una arquitectura basada en un agente conversacional de inteligencia artificial que utiliza unas herramientas implementadas para realizar unas tareas específicas y completar el flujo de trabajo diseñado para responder a las consultas del usuario. Mediante técnicas de *prompt engineering*, el agente se enriquece de unas instrucciones para ser capaz de utilizar las herramientas y realizar cálculos con las estadísticas de los jugadores y equipos y así poder generar la respuesta a la consulta realizada por el usuario.

Finalmente, la combinación de ambos bloques se unifica en una interfaz web interactiva, dinámica y sencilla, permitiendo a usuarios no técnicos en programación obtener tanto predicciones del rendimiento de jugadores como consultas sobre información útil para el análisis de partidos o *scouting* de jugadores.

En conclusión, se ha buscado el desarrollo de una plataforma líder en este campo y flexible, cuya arquitectura puede adaptarse a distintos dominios mediante la sustitución de los datos de entrada y el ajuste del *prompt* del agente. Esta capacidad de generalización refuerza el valor del proyecto y sienta las bases para futuras aplicaciones más allá del ámbito futbolístico.

Palabras clave: Inteligencia artificial, LLMs, *frameworks*, agentes, aprendizaje automático, rendimiento.

ABSTRACT

The project aims to provide users with no programming experience with access to advanced football data analysis through an interactive platform consisting of two main blocks. The solution integrates a conversational agent to answer queries about statistics calculations for teams and players in the Spanish league, together with customised prediction models, offering an accessible and valuable tool for sports analysis.

Firstly, the first block consists of reviewing machine learning model techniques to be applied in predicting player performance. To address this block, it has been necessary to complete an ETL process, which is based on data extraction, processing and loading, followed by the division of the data into training, validation and testing with the aim of obtaining the optimal hyperparameters of the models and being able to train with a large number of observations to achieve acceptable metric values in the test set.

Secondly, the second block consists of the development of an architecture based on an artificial intelligence conversational agent that uses tools implemented to perform specific tasks and complete the workflow designed to respond to user queries. By using prompt engineering techniques, the agent is enriched with instructions to be able to make use of the tools and perform calculations with player and team statistics in order to generate the response to the user's query.

Finally, the combination of both blocks is unified in an interactive, dynamic and simple web interface, allowing users who are not technical in programming to obtain both predictions of player performance and queries about useful information for match analysis or player scouting.

In conclusion, the aim has been to develop a leading and flexible platform in this field, whose architecture can be adapted to different domains by replacing the input data and adjusting the agent's prompt. This capacity for generalisation reinforces the value of the project and lays the foundations for future applications beyond the field of football.

Keywords: *Artificial intelligence, LLMs, frameworks, Agents, Machine Learning, performance.*

Capítulo 1. Introducción

En los últimos años, debido al creciente volumen de datos disponibles, seguida de la evolución y desarrollo de la inteligencia artificial, los modelos de lenguaje de gran escala (*Large Language Model*) o también conocidos como *LLMs* y en general, los recientes *frameworks* de agentes, han transformado la manera de interpretar y analizar la información, facilitando la creación de asistentes conversacionales capaces de realizar tareas complejas. Estas tareas son posibles de abordar gracias a la irrupción de los *LLMs* y sus grandes ventajas y beneficios, explicados en los artículos de Yue Zhang y Zhibo Sun [39], que hablan sobre los aspectos positivos y negativos de los *LLMs* en seguridad y privacidad, o incluso el artículo de Rajvardhan Patil y Venkat Gudivada [38], que aborda una comparación de técnicas, retos y un análisis de la situación actual de los *LLMs*. Paralelamente, la integración de modelos de aprendizaje automático en entornos accesibles se ha convertido en una prioridad para lograr explotar el valor de los datos. En este contexto, este trabajo propone una solución integral combinando técnicas de procesamiento de datos, algoritmos de aprendizaje automático e implementación de agentes de inteligencia artificial generativa con el objetivo de facilitar tanto la predicción como la consulta inteligente de datos. Por ello, la creación de esta plataforma se basa en dos pilares fundamentales.

Por un lado, se utilizan técnicas de extracción, procesamiento y análisis de datos, incluyendo la detección de valores atípicos, faltantes, transformación de variables y análisis de correlaciones, entre otros, para la aplicación de modelos de aprendizaje automático y su posterior evaluación del rendimiento mediante métricas y comparación de resultados.

Por otro lado, y como elemento diferencial del proyecto, se ha implementado un agente de inteligencia artificial generativa capaz de responder preguntas y consultas sobre los datos procesados, empleando *frameworks* de agentes y técnicas de *prompt engineering*, utilización de *LLMs* y *tools*. Este enfoque permite tanto la predicción de variables como la consulta inteligente de datos, mejorando la accesibilidad y el análisis de la información.

1.1 Aplicación práctica: Análisis de datos deportivos

Todos estos conceptos, técnicas y herramientas han sido utilizados para un caso de uso representativo y realista, basándose en los datos obtenidos de una de las competiciones deportivas más importantes del mundo, la liga española de fútbol. Mediante conexión vía API, no solo se consiguieron los datos de las participaciones de todos los jugadores de todos los equipos en todos los partidos de la temporada 2023-2024, sino también información sobre los mismos partidos de cada jornada, incluyendo fecha, resultado y estadio de cada partido, entre otros. La obtención de variables numéricas, categóricas y temporales, así como relaciones entre entidades como jugadores, equipos y partidos, permite la posibilidad de enriquecer al sistema implementado para que no solo los modelos de aprendizaje automático puedan realizar predicciones relevantes, como el rendimiento de un jugador o si será titular o no, sino también enriquecer al agente conversacional para dar respuestas a consultas complejas del usuario basándose únicamente en los datos proporcionados.

Finalmente, mediante una interfaz web interactiva y sencilla, se da la posibilidad a usuarios

no expertos de formular preguntas y obtener respuestas directamente basadas en los datos. Además, esta plataforma permitirá realizar predicciones en su interfaz sobre el rendimiento de los jugadores e incluso estimar si un jugador será titular en un partido, gracias a los modelos entrenados con los datos de la temporada 23/24. Para ello, el usuario podrá introducir variables clave como el rival al que se enfrentará ese jugador, su estado de forma reciente, el rendimiento en los últimos partidos de ambos equipos previo al partido a predecir y muchas más características relevantes que afectan el desarrollo de un partido, como el estadio y fecha, entre otras.

En conclusión, el enfoque adoptado en este trabajo permite al usuario una experiencia fluida e interactiva, adaptada a distintos niveles de conocimiento técnico. Además, la plataforma desarrollada podría ser de una gran utilidad para analistas de partidos, equipos de *scouting* y cuerpos técnicos de alto nivel para el estudio del rival, rendimiento de jugadores o predicción de alineaciones probables en partidos específicos.

1.2 Motivación y necesidad de la plataforma

En un entorno donde el volumen de datos deportivos crece constantemente y los equipos técnicos demandan información procesada en tiempo real, surge la necesidad de herramientas que permitan tanto la predicción como la exploración inteligente de datos, como el trabajo de Zhiqiang Pu y Yi Pan [40], donde utilizan la inteligencia artificial y analíticas de fútbol para llevar a cabo un proceso de observación, orientación, decisión y acción (*OODA*, según cita el trabajo), con el objetivo de reducir la brecha entre los dominios reales y virtuales sobre la toma de decisiones y análisis de partidos de fútbol. Sin embargo, muchas soluciones actuales requieren conocimientos avanzados, lo que limita su adopción por parte de perfiles no técnicos.

Este trabajo busca reducir esa brecha mediante el desarrollo de una plataforma accesible, intuitiva e interactiva, que permita no solo realizar predicciones personalizadas, sino también obtener respuestas a consultas complejas directamente sobre los datos procesados. En este caso, la información utilizada se enfoca en datos deportivos correspondientes a la liga española de fútbol. Sin embargo, la arquitectura desarrollada puede ser generalizada. De hecho, bastaría con simplemente proporcionar un nuevo conjunto de datos y modificar el flujo e instrucciones del agente para que actúe como experto en ese nuevo contexto. Esto es posible gracias a que las funciones implementadas externas que recibe el agente son generales a cualquier archivo de datos.

1.3 Objetivos

En esta sección se describen los objetivos propuestos de este trabajo para tratar de satisfacer las necesidades actuales y conseguir una herramienta eficiente e interactiva que lidere en el ámbito deportivo y ser de gran utilidad para los equipos de *scouting*, analistas de datos e incluso aficionados, concretamente en el fútbol profesional nacional.

1.3.1. Objetivos principales

Para lograr una solución y plataforma óptima y útil en el análisis de datos de equipos, partidos y jugadores, este trabajo debe cumplir dos objetivos principales:

- Facilitar la predicción de rendimientos de jugadores, estados de forma e incluso titularidades en un partido dependiendo de las condiciones y características asociadas a cada evento mediante el **uso de técnicas de aprendizaje automático avanzadas**.
- Permitir la consulta inteligente de información basada en los datos proporcionados de una manera sencilla e interactiva para su adaptación a usuarios con distintos niveles de conocimiento técnico mediante uso de LLMs y *frameworks* de agentes de **inteligencia artificial generativa**. Así, los usuarios pueden preguntar características más específicas de los partidos, jugadores y equipos.

1.3.2. Objetivos específicos

- Extraer, limpiar y procesar los datos relacionados con la liga española de fútbol en la temporada 2023/24.
- Obtener hiperparámetros óptimos tras una división de los datos en entrenamiento, validación y *test* para predecir variables clave relacionadas con el rendimiento de jugadores y alineaciones probables. Así, se trata de conseguir los mejores resultados posibles dentro de las limitaciones de los modelos de aprendizaje automático ya entrenados con sus hiperparámetros óptimos.
- Evaluar mediante diferentes métricas de regresión y clasificación de los modelos utilizados de aprendizaje automático.
- Implementar un agente de inteligencia artificial generativa capaz de responder preguntas y consultas sobre los datos procesados mediante el uso de *prompt engineering*, LLMs y herramientas auxiliares *tools*. Debe tener la capacidad de decisión para elegir las herramientas que necesita en función de la tarea o consulta que requiera el usuario y continuar con el flujo hasta la generación de la respuesta.
- Diseñar una interfaz web interactiva mediante el uso de Streamlit que permita a cualquier usuario realizar predicciones o consultar información de forma sencilla
- Validar y probar el sistema desarrollado mediante ejemplos de uso prácticos y análisis de resultados obtenidos.

Capítulo 2. Estado del arte

En este capítulo se comentan algunos estudios que han contribuido e impulsado el crecimiento de la tecnología y especialmente, la inteligencia artificial, en el ámbito deportivo y enfocándose en el fútbol.

Actualmente, la inteligencia artificial ha adquirido un gran protagonismo en el mundo del deporte debido a su utilidad y la capacidad de potenciar las cualidades de los deportistas, identificar patrones, predecir resultados, rendimiento e incluso lesiones. Y es ahí el punto que se tratará de maximizar en este trabajo para conseguir una plataforma eficiente y útil para predecir el rendimiento de los jugadores. Además, una vez se logra recopilar una gran cantidad de información sobre los equipos y jugadores, es imprescindible que la plataforma pueda resolver consultas y estadísticas concretas que conlleven cálculos sobre los equipos o jugadores. De esta forma, se convierte en una plataforma mucho más potente e interactiva.

2.1 Predicción del rendimiento

En primer lugar, un estudio interesante enfocado en la predicción del rendimiento de los jugadores es el trabajo realizado por Íñigo Gómez Carvajal en 2021 [15], donde, al igual que en este trabajo, se tiene en cuenta el nivel de rendimiento tanto del propio jugador a predecir como del nivel del equipo global al que pertenece y el rival que se enfrenta. Además, Bransen y Van Haaren [3] muestran la importancia de las acciones tanto ofensivas como defensivas en el rendimiento de los jugadores y no solo el enfoque en goles o tiros a portería. De hecho, utilizan XGBoost para estudiar las contribuciones de los pases de un jugador a goles para sus equipos y ofrecer reemplazos de jugadores en base a cómo de similares son sus contribuciones.

Otro estudio reciente es el de Pappalardo et. al [4], quienes utilizan un *dataset* de eventos espacio-temporales en partidos a lo largo de una temporada y en base a las acciones ocurridas y el resultado del partido en victoria o derrota realizan una medición del rendimiento de un jugador en función de un sistema propio que se basa en el peso de sus acciones de cara a ayudar a sus equipos a conseguir victorias. Los pesos de las acciones se diferencian en base a la posición y se han determinado mediante la generación de un modelo de clasificación lineal mediante vectores de soporte (LSVC) por posición.

Un último estudio interesante es la tesis realizada por Shen Chen [6]. En ella recoge diferentes estrategias para predecir resultados de partidos de la liga alemana durante las temporadas 2017-18 hasta 2021-22 incluyendo la utilización de goles esperados, o en inglés *Expected Goals* (xG), una herramienta estadística de Opta con gran relevancia en los últimos años.

2.2 Consulta inteligente de la información

Un trabajo de gran interés es el de Nils Blomgrem en su tesis sobre la implementación de un chatbot para analíticas de fútbol [7]. En ella, profundiza temas como la orquestación de *LLMs*, arquitectura RAG anteriormente mencionada y *Function Calling*, que permite la llamada a APIs y recuperación de datos en tiempo real y generación de objetos JSON para su integración directa con la aplicación.

Seguidamente, un trabajo reciente en el área de la consulta de información a bases de datos mediante *queries* en lenguaje natural es el desarrollo de SoccerRAG [5]. Se trata de una plataforma para potenciar el uso de los *LLMs* y arquitecturas *RAG* que extraen información multimodal de fútbol a partir de consultas con lenguaje natural, facilitando la interacción con el usuario y accesibilidad a datos futbolísticos mediante peticiones o *queries* gracias a la interfaz implementada. En este trabajo, no hizo falta la implementación de una arquitectura *RAG* debido a que la información almacenada en las diferentes tablas *SQLite* y la transformación de las peticiones en lenguaje natural a lenguaje SQL para poder obtener la recuperación de la información, o bien llamado en inglés la parte del *retrieval* en un *RAG*, mostraban resultados satisfactorios, efectivos y rápidos. Aún así, el desarrollo del *framework* de SoccerRAG [5] constituye un gran avance en la aplicación práctica de la inteligencia artificial en el área del deporte.

Una vez introducido el concepto de arquitectura *RAG*, en este trabajo se valoró la implementación de esta opción y la posibilidad de posibles mejores resultados. Sin embargo, como ya se ha comentado anteriormente, los resultados tan positivos y efectivos del almacenamiento de los datos en diferentes tablas *SQLite* dependiendo del *dataset* y sus respectivas temporadas por jugadores o bien por equipos, permite ser una plataforma escalable y con rápidos resultados en cuanto a la búsqueda de información. Aún así, tras el estudio de todos estos trabajos anteriores del estado del arte, la solución como propuesta de mejora para este trabajo en un futuro es la implementación de un *RAG* que tiene la arquitectura mostrada en la Figura 2.1, dividida en 7 grandes pasos.

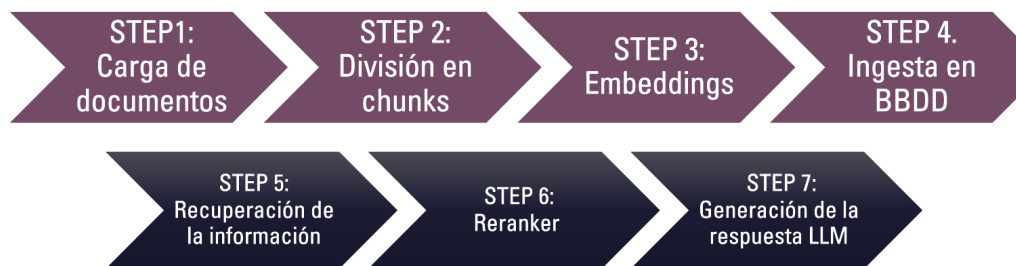


Figura 2.1. Arquitectura *RAG* diseñada. Fuente: Elaboración propia

La primera fila de color morado de la Figura 2.1 pertenece a la parte de la ingesta en una base de datos vectorial. Este concepto también lo trató Nils Blomgren en su tesis [7]. Estas bases de datos vectoriales almacenan los vectores o *embeddings* y permiten indexarlos para poder acceder a ellos en la menor cantidad de tiempo posible. Algunos ejemplos de ellos son *Pinecone*, como la que usa Nils Blomgren, aunque existen muchos más ejemplos, como *ChromaDB*. Por tanto, la fase de ingesta se resume en cuatro grandes pasos:

- Paso o *Step* 1: Carga de documentos, o en este caso, carga de los *datasets* de cada temporada con la información recopilada de la API incluyendo los jugadores y equipos.
- Paso o *Step* 2: División en *chunks*. Se tratan de trozos o fragmentos de texto en los que se divide la fuente de información o documentos.
- Paso o *Step* 3: Vectorización o *embeddings*. Se trata del proceso de transformación de los

chunks en vectores para poder ser indexados y aumentar la rapidez en la búsqueda de recuperación de la información en una base de datos vectorial.

- Paso o *Step 4*: Ingesta en la base de datos vectorial. Se trata del proceso de incorporar a la base de datos los nodos con el texto, vectores e incluso metadatos que puedan ser útiles para realizar filtros previos en la búsqueda de la información en la base de datos.

Por otro lado, la segunda fila de color negro de la Figura 2.1 corresponde a la segunda fase del RAG que se había diseñado como opción alternativa a la implementación final de las tablas *SQLite*. Esta se trata de la fase de recuperación de la información una vez ingestados los nodos en la base de datos. Se compone de los siguientes tres pasos.

- Paso o *Step 5*: Recuperación de la información de la base de datos a partir de la petición, *query* o pregunta que realiza el usuario. Se trata de devolver los nodos que contengan la información a la respuesta de la petición del usuario. Se utilizan distancias matemáticas para calcular las proximidades con respecto a la *query*.
- Paso o *Step 6*: Reranker. Se trata de reordenar los nodos devueltos en un orden en función del *score* o puntuación de los nodos con respecto a la pregunta que el usuario ha consultado. Es un paso opcional y no es imprescindible en un RAG, pero ayuda a los modelos a tener jerárquicamente los nodos más importantes y con más posibilidades de contener la información de la respuesta.
- Paso o *Step 7*: Generación de la respuesta. Se trata de llamar a los modelos *LLMs* para generar la respuesta en base a los nodos recibidos con el texto que contienen. Finalmente, la interfaz mostraría la respuesta generada por el modelo al usuario, indicando incluso la página del documento donde se basa la respuesta. Esto sería posible gracias a la incorporación de un metadato llamado 'número de página' en cada nodo e ingestarlo en la base de datos junto con el texto, vectores y resto de metadatos.

Capítulo 3. Marco teórico

El marco teórico de este proyecto de aplicación de aprendizaje automático e inteligencia artificial generativa para la predicción y consulta inteligente de datos incluye las siguientes secciones, principalmente enfocado en toda la teoría que se necesita para el primer bloque del trabajo.

- Aprendizaje automático: conceptos generales
- Modelos de aprendizaje automático
- Métricas de aprendizaje automático

3.1 Aprendizaje automático: conceptos generales

En esta sección se define el aprendizaje automático y sus principales fundamentos. El aprendizaje automático, también conocido en inglés como *machine learning*, es la disciplina dentro de la inteligencia artificial, que a través de algoritmos, dota a los ordenadores o un sistema de la capacidad de aprender y resolver problemas complejos para mejorar el rendimiento de los modelos.

Cabe destacar la figura de Arthur Samuel, uno de los pioneros en el campo del aprendizaje automático, que lo definió como 'el campo de estudio que proporciona a los ordenadores la capacidad de aprender sin estar explícitamente programados [59].

A su vez, el aprendizaje automático se divide en tres grandes tipos de aprendizaje: **supervisado**, **no supervisado** y **por refuerzo**, que se explican en la Sección 3.1.1, Sección 3.1.2 y Sección 3.1.3, respectivamente.

3.1.1. Aprendizaje supervisado

El aprendizaje supervisado consiste en un tipo de aprendizaje donde se utiliza un conjunto de datos que incluye entradas y salidas para que los modelos generen la salida deseada mediante un aprendizaje previo, pues se trata de explicar la relación de la variable dependiente y en función de las n variables independientes con el objetivo de predecir el comportamiento y valor de los datos desconocidos. El conjunto de datos total se representa de la siguiente manera:

$$D = \{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}, \quad y^i \in \mathbb{R}, \quad x^i \in \mathbb{R}^n, \quad \forall i = 1, \dots, m$$

donde m es el número total de muestras, n el número total de variables, y la variable x representa las características de entrada o variables independientes, cumpliendo que $x^i \in \mathbb{R}^n$ para todo $i = 1, \dots, m$. mientras que la variable y representa la salida o variable dependiente. Esta variable objetivo o *target* es la que se intenta predecir, donde sin pérdida de generalidad, tanto si es numérica como nominal, se tiene $y \in \mathbb{R}$. Estas variables definidas x e y presentes en el conjunto D están relacionadas mediante un patrón f , que solo consiste en un constructo teórico útil del que se asume su existencia para explicar la relación entre las variables x e y .

El aprendizaje supervisado se puede separar en dos tipos de tareas: clasificación y regresión.

- La **regresión** es la tarea donde la variable objetivo es de tipo **numérico**. Algunos ejemplos de problemas serían predicciones de salarios o precios [52]. Algunas técnicas usadas en la tarea de regresión que se utilizan para resolver estos problemas serían la regresión lineal o polinómica.
- Por otro lado, la **clasificación** consiste en la tarea que tiene como variable objetivo de tipo **categorico**, es decir, trata de predecir a qué clase o categoría pertenecen los datos. Una técnica usada en la tarea de clasificación sería la regresión logística, explicada en la Sección 3.2.3.

3.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado trata de examinar y agrupar conjuntos de datos no etiquetados y tiene como objetivo encontrar una estructura en los datos.

Para la realización de *datasets* donde se utiliza el aprendizaje supervisado, los seres humanos tienen que intervenir para etiquetar los conjuntos de datos a fin de entrenar al modelo, es decir, un modelo supervisado depende de la capacidad de conocer el resultado para concluir que una característica es un factor significativo en la variable objetivo. El conjunto de datos en el **aprendizaje no supervisado** [27] se representa de distinta manera:

$$D = \{x^1, x^2, \dots, x^m\} \quad x^i \in \mathbb{R}^n \quad \forall i = 1, \dots, m$$

Existen diferentes algoritmos que pueden descubrir patrones desconocidos en los datos sin dicha supervisión humana. Existen tres categorías principales de estos algoritmos:

- **Agrupación de clústeres:** se centran en la identificación de grupos de registros similares y en el etiquetado de registros según el grupo al que pertenecen. Los métodos de agrupación en clústeres se basan en la medición de distancias entre registros y entre clústeres. Los registros se asignan a los clústeres de un modo que tiende a minimizar la distancia entre los registros pertenecientes al mismo clúster.
- **Reglas de asociación:** Método basado en reglas para detectar relaciones entre variables en un conjunto de datos determinado. Un ejemplo podría ser la lista "Descubrimiento semanal" de Spotify[28], surgido tras un gran análisis y entendimiento del comportamiento y consumo de los clientes.
- **Reducción de dimensionalidad:** La reducción de la dimensionalidad es una técnica utilizada cuando el número de características, o dimensiones, de un determinado conjunto de datos es demasiado elevado. Esta consiste en una reducción del número de observaciones iniciales a un tamaño gestionable, además de preservar la integridad del conjunto de datos lo máximo posible. Se usa comúnmente en la fase de preprocesamiento de datos. Existen varios métodos de reducción de dimensionalidad:
 - 1) Análisis de componentes principales (PCA)** [20]: Este método utiliza una transformación lineal para crear una nueva representación de datos, dando como resultado un conjunto de componentes principales. El primer componente principal es la dirección que maximiza la varianza del conjunto de datos. Si bien el segundo componente principal también encuentra la varianza máxima en los datos, no tiene ninguna correlación con el primer componente principal, lo que genera una dirección que es perpendicular

u ortogonal al primer componente. Este proceso se repite en función del número de dimensiones, donde un siguiente componente principal es la dirección ortogonal a los componentes anteriores con mayor varianza.

2) Descomposición en valores singulares(SVD): SVD se denota mediante la fórmula: $A = USV^T$ donde U y V son matrices ortogonales. S es una matriz diagonal y los valores de S se consideran valores singulares de la matriz A . Similar a PCA, se suele utilizar para reducir el ruido y comprimir datos.

3) Codificadores automáticos: Utilizan las redes neuronales para comprimir los datos y luego volver a crear una nueva representación de la entrada de datos original. La principal diferencia entre los codificadores automáticos y el análisis de componentes principales (PCA) es que mientras PCA encuentra las direcciones a lo largo de las cuales puede proyectar los datos con la variación máxima, los codificadores automáticos reconstruyen la entrada original con solo una versión comprimida de la misma.

3.1.3. Aprendizaje por refuerzo

Finalmente, esta última parte de la sección se centra en el **aprendizaje por refuerzo**[8]. También conocido como **Reinforcement Learning (RL)**, consiste en un tipo de aprendizaje dentro del aprendizaje automático cuyo objetivo para resolver los problemas se basa en prueba y error. Se toman una serie de decisiones y en función de las acciones realizadas recibe recompensas o penalizaciones con el objetivo de maximizar dicha recompensa. Destaca la existencia de un agente y un entorno o ambiente, seguido de la acción, estado, recompensa y recompensa acumulada.

El agente es el algoritmo que se quiere entrenar y que aprenda a tomar decisiones. Otro concepto clave es el entorno, que es el espacio donde interactúa el agente con atributos como variables, valores límite, reglas y acciones válidas. La acción consiste en el paso que el agente realiza para navegar por el entorno, seguido del estado, que es el indicador del ambiente de cómo están los diversos elementos que lo componen en un momento dado. También, se tiene la recompensa, que es el premio, penalización o valor nulo que orientarán al agente según si la acción era positiva o negativa. Por último, se tiene la recompensa acumulada, que se trata de la suma de todas las recompensas o valor final. A continuación, se muestra la estructura que se sigue en el aprendizaje por refuerzo en la Figura 3.1.

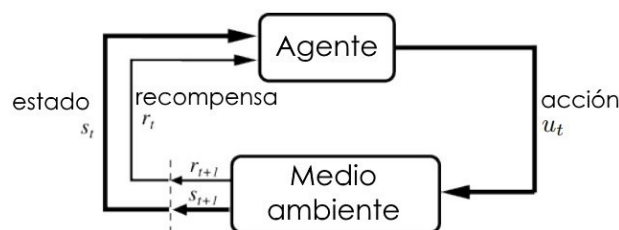


Figura 3.1. Estructura en el aprendizaje por refuerzo. Fuente:[31]

Un caso de uso con gran presencia en la actualidad del aprendizaje por refuerzo es en el área del marketing [42]. En aplicaciones como los sistemas de recomendación, el RL puede

personalizar las sugerencias para los usuarios individuales en función de sus interacciones realizando ofertas en tiempo real siempre con el objetivo de maximizar las ganancias y el crecimiento de los clientes. Gracias a los datos recopilados, se analiza el comportamiento del cliente y adapta sus recomendaciones de forma dinámica.

En cuanto al funcionamiento, el aprendizaje por refuerzo se basa en el proceso de decisión de Markov, modelo matemático de la toma de decisiones que utiliza intervalos de tiempo discretos. En cada paso, el agente lleva a cabo una nueva acción que da como resultado un nuevo estado del entorno. Del mismo modo, el estado actual se atribuye a la secuencia de acciones anteriores. Mediante prueba y error y su movimiento por el entorno, el agente crea un conjunto de reglas o políticas “condicionales”. Dichas políticas ayudan a decidir qué acción tomar a continuación para obtener una recompensa acumulada óptima. El agente también debe elegir entre seguir explorando el entorno para obtener nuevas recompensas de estado-acción o seleccionar acciones conocidas con altas recompensas de un estado determinado. Esto se denomina compensación entre exploración y explotación.

3.1.4. Generalización, división de los datos y complejidad

Volviendo al aprendizaje supervisado, se introduce el concepto de generalización. Una buena **generalización** consiste en la capacidad de un modelo de lograr predecir con acierto datos nuevos a partir de la información contenida en las observaciones empleadas para el aprendizaje o ajuste de un modelo. Para realizar una evaluación justa y objetiva del rendimiento del modelo que permita encontrar el equilibrio adecuado entre la flexibilidad y la capacidad de generalización, este será evaluado con un conjunto de datos independiente a los datos de entrenamiento. Esto permitirá evaluar la capacidad de generalizar a nuevos datos y que no se ajuste excesivamente a las muestras de ese entrenamiento. Por esta razón, surge la necesidad de dividir el conjunto de datos en un subconjunto de datos de **entrenamiento** y en un subconjunto de datos de **test**, donde dichos datos de entrenamiento se utilizan para ajustar los modelos al problema planteado, mientras que los datos de test sirven para comprobar el rendimiento o comportamiento de los modelos y así poder establecer una evaluación de cada modelo que se utilice. Para llevar a cabo dicha división de datos, se utiliza un procedimiento de muestreo: muestreo aleatorio simple o muestreo estratificado.

Por otro lado, destaca el concepto de **complejidad**. Un modelo demasiado complejo va a tener demasiada flexibilidad y va a ajustar demasiado, mientras que si tiene poca complejidad, no va a capturar bien el patrón de relación entre *inputs* y *outputs*. El **sobreajuste u overfitting** ocurre cuando el modelo se ajusta tan exactamente a los datos de entrenamiento que el modelo falla en la generalización, problema que puede ser causado por un excesivo sobreentrenamiento, como ya se explicó anteriormente. En este caso, se tiene: $J(\theta) \rightarrow 0$ donde J es la función de coste, que tenderá a 0 debido a ese sobreentrenamiento, pues ajusta los datos de entrenamiento con la mayor precisión posible. En la última imagen de la Figura 3.2 se observa una situación de sobreajuste.

Por otra parte, existe el problema de **subajuste o underfitting**, que consiste en el caso contrario al **overfitting** o sobreajuste. Esto ocurre cuando un modelo demasiado simple no es

capaz de hacer predicciones precisas ni encontrar patrones importantes debido a esa falta de entrenamiento, entre otras causas. Esa incapacidad de capturar la complejidad de los datos puede causar consecuencias significativas en la práctica. En la primera imagen de la Figura 3.2 se puede observar un ejemplo de una situación de *underfitting*, mientras que la segunda imagen corresponde a una situación intermedia con un correcto ajuste y buen equilibrio de complejidad.

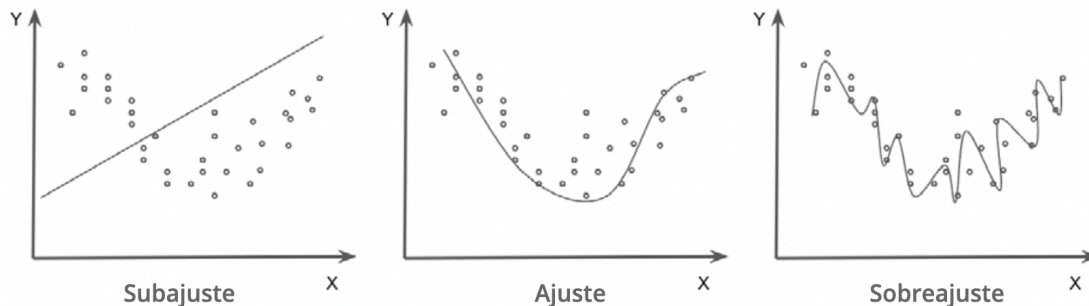


Figura 3.2. Ejemplo de subajuste, buen ajuste y sobreajuste. Fuente:[54]

Volviendo al concepto de complejidad, existen parámetros externos al modelo que afectan su comportamiento o rendimiento, pero que no se ajustan durante el proceso de entrenamiento mediante el algoritmo de aprendizaje. Estos parámetros externos son los hiperparámetros. Los hiperparámetros deben ser configurados por el diseñador del modelo antes de comenzar el proceso de entrenamiento. Estos son críticos para el rendimiento y la eficacia de los modelos de aprendizaje automático. El ajuste de estos hiperparámetros influye de manera importante en la complejidad del modelo, buscando un equilibrio de complejidad óptimo entre el subajuste y el sobreajuste. También se puede incluir otro subconjunto conocido como datos de **validación**. Este subconjunto resulta útil para el ajuste de dichos hiperparámetros que se emplean con cada técnica. Una buena elección de los valores de los hiperparámetros permitirá un mejor rendimiento del modelo que se verá reflejado en los datos de test mencionados anteriormente.

En la división de un *dataset*, una práctica común suele ser asignar el 70 % de las instancias para el subconjunto de datos de entrenamiento y lo restante para test. Otro ejemplo, caracterizado por la existencia de un gran número de observaciones, puede ser asignar un 50 % para datos de entrenamiento, junto con un 25 % para datos de validación y el 25 % restante para el subconjunto de test. Una alternativa para evitar la división de los datos en estos tres conjuntos es la **validación cruzada** [55], también conocido en inglés como *Cross Validation*. Esta técnica tiene como idea principal dividir el conjunto de datos en subconjuntos de entrenamiento y prueba iterativamente, ajustando el modelo en el conjunto de entrenamiento y evaluándolo en el conjunto de prueba en cada iteración. El método más usado es el '**k-fold cross-validation**', donde k es un número entero mayor que 1 que define el número de particiones del *dataset* así como el número de iteraciones de entrenamiento y validación que se usarán al construir el modelo. Se trata de un proceso iterativo que se repite k veces que consiste en la toma de una de las k particiones y se mantiene "oculta" al modelo. Posteriormente,

se utiliza cada uno de los k *folds* como conjunto de prueba una vez, y los $k-1$ *folds* restantes como conjunto de entrenamiento. Una vez completadas las k iteraciones, se promedian los resultados de las métricas de evaluación obtenidas en cada *fold*.

Por último, una solución para encontrar el equilibrio de complejidad o incluso prevenir el problema del sobreajuste puede ser reducir el número de características/variables. Otra, más común, es la **regularización**. Esta técnica consiste en la construcción de una nueva función de coste $J'(\theta)$ que incluye la introducción de una penalización adicional en la función de coste inicial $J(\theta)$ para promover soluciones más simples y estables.

$$J'(\theta) = J(\theta) + \lambda \cdot \text{Regularización}$$

donde $J(\theta)$ es la función de coste, λ es el parámetro de regularización y *Regularización* es el término adicional que penaliza los parámetros del modelo, evitando el sobreajuste. Existen varias técnicas de regularización [16] que asocian coeficientes a las variables de entrada, como ocurre en la regresión logística de la Sección 3.2.3.

- **Ridge**: Muy similar al enfoque de mínimos cuadrados, en esta técnica de regularización se penaliza la suma de los coeficientes elevados al cuadrado.

$$\text{Regularización} = \sum_{j=1}^n \theta_j^2$$

A esta penalización se le conoce como L_2 y tiene el efecto de reducir de forma proporcional el valor de todos los coeficientes del modelo pero sin que estos lleguen a cero. Ridge es especialmente útil cuando existe colinealidad entre las variables de entrada debido a la alta correlación entre ellas. Esta penalización permite reducir los efectos de la colinealidad al distribuir de una forma más uniforme el peso entre las variables correlacionadas.

- **Lasso**: La penalización L_1 tiene el efecto de forzar a algunas de las estimaciones de coeficientes ser exactamente iguales a cero cuando el parámetro λ de ajuste es suficientemente grande. Esta técnica de regularización Lasso realiza selección de variables. Como resultado, los modelos generados a partir de lasso son generalmente mucho más fáciles de interpretar que los producidos por regresión ridge.

$$\text{Regularización} = \sum_{j=1}^n |\theta_j|$$

- **Elastic Net**: Es una combinación de L_1 y L_2 donde el grado de influencia de ambas está controlado por un parámetro $\alpha \in [0, 1]$. El caso de $\alpha = 0$ corresponde a la penalización Ridge, mientras que el caso $\alpha = 1$ corresponde a la penalización Lasso. Una estrategia frecuentemente utilizada es asignarle casi todo el peso a la penalización L_1 (α muy próximo a 1) para conseguir seleccionar predictores y un poco a la L_2 para dar cierta estabilidad en el caso de que algunos predictores estén correlacionados.

$$\text{Regularización} = \alpha L_1 + (1 - \alpha) L_2$$

Teniendo en cuenta que L_1 y L_2 son las expresiones de regularización de Lasso y Ridge respectivamente.

3.1.5. Ajuste de hiperparámetros

Tras la definición de los hiperparámetros en el apartado 3.1.4, en esta sección se explicará la importancia y las técnicas de ajuste de los hiperparámetros. Este consiste en el proceso de encontrar la mejor combinación de valores para los hiperparámetros de un algoritmo de aprendizaje automático mediante la maximización de su rendimiento en un conjunto de datos de prueba o validación. Los hiperparámetros controlan de forma directa la estructura, funciones y rendimiento de los modelos y dicho ajuste permite modificar el rendimiento del modelo para lograr resultados óptimos.

A continuación se explicarán algunas técnicas para el ajuste de hiperparámetros.

- **Optimización bayesiana:** Técnica basada en el teorema de Bayes, que describe la probabilidad de que ocurra un evento de acuerdo con el conocimiento actual. Cuando esto se aplica a la optimización de hiperparámetros, el algoritmo crea un modelo probabilístico a partir de un conjunto de hiperparámetros que optimiza una métrica específica. También utiliza información sobre el rendimiento del modelo en iteraciones anteriores para guiar la búsqueda hacia regiones prometedoras del espacio de hiperparámetros.
- **Búsqueda por cuadrícula (*Grid Search*):** Esta técnica consiste en definir un conjunto de valores posibles para cada hiperparámetro y evaluar todas las posibles combinaciones de valores en un conjunto de validación. Se selecciona la combinación de hiperparámetros que maximiza el rendimiento del modelo en el conjunto de validación. La búsqueda por cuadrícula suele ser relativamente tediosa e intensiva a nivel computacional, sobre todo con números elevados de hiperparámetros.
- **Búsqueda aleatoria (*Random Search*):** En contraposición a la búsqueda por cuadrícula, la búsqueda aleatoria selecciona aleatoriamente valores para cada hiperparámetro dentro de un rango especificado y evalúa un número predeterminado de combinaciones de hiperparámetros.

3.2 Modelos de aprendizaje automático

3.2.1. Redes neuronales

Las redes neuronales artificiales (ANN) [9] es un algoritmo inspirado en el cerebro humano que intenta replicar el funcionamiento de las neuronas biológicas mediante un sistema de neuronas artificiales interconectadas. Estas redes están formadas por capas de nodos. Se caracterizan por tener una capa de entrada, una o varias capas ocultas en caso de que existan (*hidden layer*) y una capa de salida. Cada neurona artificial está conectada a otras a través de un peso, que se asocia a la conexión entre pares de neuronas, y un umbral específicos. Las conexiones pueden darse no solo entre neuronas de capas contiguas, sino también dentro de una misma capa o incluso saltándose capas. Si la salida de un nodo individual supera el valor de umbral asociado, este se activa y transmite datos a aquellos de la siguiente capa. Si no se supera el umbral, la información no se transmite. Un elemento

esencial de las redes neuronales artificiales son las funciones de activación [17], que se tratan de unas funciones que determinan si una neurona debe activarse o no. Estas funciones transforman la señal de entrada de un nodo de una red neuronal a una señal de salida que pasa a la siguiente capa. Las funciones de activación introducen el concepto de no linealidad y permiten que las redes neuronales aprendan y modelen relaciones no lineales complejas. A continuación, se hará una breve explicación de las capas principales que pueden existir en una red neuronal:

- **Capa de entrada:** Los nodos de entrada reciben los datos y añaden una neurona de sesgo mandando un 1.
- **Capa oculta:** Cada capa oculta analiza la salida de la capa anterior, la procesa aún más, es decir, se realizan cálculos intermedios y la pasa a la siguiente capa. Es importante resaltar que puede haber más de una capa oculta.
- **Capa de salida:** Proporciona todo el resultado final del procesamiento de datos. Las neuronas de esta capa poseen funciones de activación específicas. En un ejemplo de clasificación binaria, únicamente habrá un nodo de salida. En una situación de clasificación multiclase, existirá más de un nodo de salida en esta última capa.

A continuación, se muestra la Figura 3.3, donde se observa un ejemplo de la estructura de una red neuronal con las tres capas que se acaban de explicar, concretamente con dos capas ocultas.

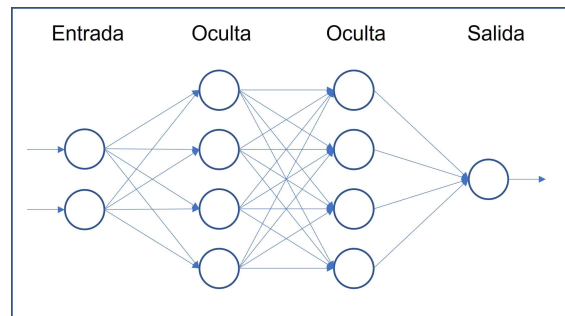


Figura 3.3. Ejemplo de red neuronal con 2 capas ocultas. Fuente:[37]

La dirección en la que se mueven los datos es otro aspecto a tener en cuenta, como es el caso de una red neuronal **alimentada hacia delante** o *feedforward*, donde los datos de la capa de entrada pasan a la capa o capas ocultas si estas existen y por último, finalizan su recorrido en la capa de salida sin haber retroalimentación ni bucles entre capas. En cada neurona de la red, se calcula una combinación lineal de las entradas y los pesos más un término de sesgo. Este valor se denota como z y se define como: $z = \sum_{i=1}^n w_i \cdot x_i + b$, siendo w_i los pesos mencionados anteriormente, x_i las entradas y b el sesgo. Posteriormente, se aplica una función de activación para obtener la salida de la neurona, denotada como a . El valor de a se obtiene como: $a = g(z)$, siendo g dicha función de activación. La función de activación transforma la combinación lineal de las entradas en una salida no lineal que se pasa a la siguiente capa de la red. En la Figura 3.4 se puede comprobar el comportamiento de cualquier neurona y se utilizan los términos z y a como se acaban de explicar.

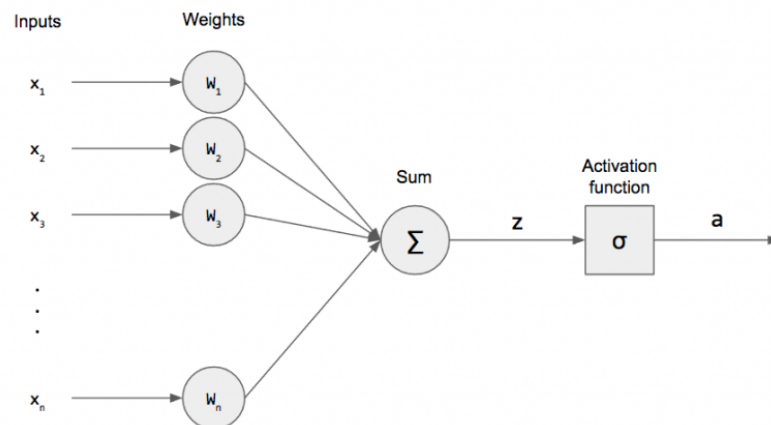


Figura 3.4. Comportamiento y funcionamiento de una neurona. Fuente:[35]

Arquitecturas de redes neuronales

En la década de 1950, el psicólogo Frank Rosenblatt desarrolló el perceptrón, conocido como el **perceptrón de Rosenblatt**, un tipo de red neuronal artificial con una única neurona como la expuesta en la Figura 3.4. Se trata del caso más sencillo de red *feedforward* que puede ser entrenada para reconocer patrones en los datos. También conocido como **Single Layer Perceptron** y caracterizado por su sencillez, no incluye capas ocultas debido a la existencia de una sola capa que se conecta a la capa inicial de entrada. Un gran inconveniente del perceptrón de Rosenblatt es la incapacidad de resolver problemas con datos **no linealmente separables** como el problema *XOR* [22].

En segundo lugar, otro tipo de arquitectura de una red neuronal es el **Multi-Layer Perceptron (MLP)**. A diferencia del perceptrón simple o de Rosenblatt, un MLP tiene múltiples capas de neuronas, incluyendo una o más capas ocultas. Durante muchos años, los MLP en realidad eran redes neuronales artificiales con una única capa oculta, o en algunos casos, unas pocas, pero potencialmente con muchas neuronas en esta capa. En la Figura 3.5, se puede ver un ejemplo de una arquitectura perceptrón multicapa con tres capas ocultas.

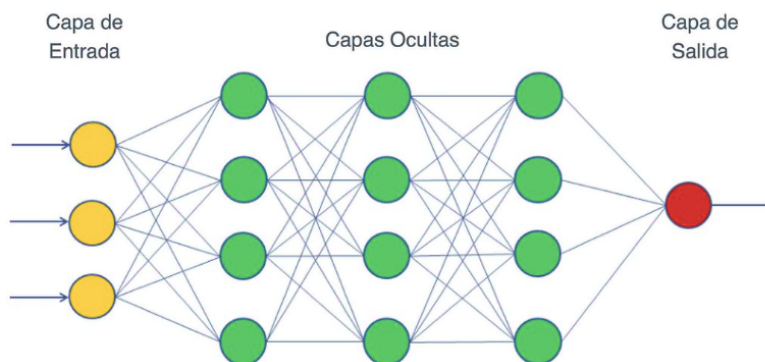


Figura 3.5. Ejemplo de arquitectura de perceptrón multicapa. Fuente: [53]

El algoritmo de entrenamiento utilizado por el MLP se basa en la propagación hacia atrás

(*backpropagation*), que como ahora se explicará a continuación, ajusta los pesos de las conexiones de la red para minimizar el error de predicción entre las salidas producidas por la red y las salidas deseadas.

Respecto a su importancia, el MLP es uno de los modelos de redes neuronales más utilizados en el aprendizaje automático debido a su capacidad **para modelar relaciones no lineales y su capacidad de generalización**. Existen algunos inconvenientes, como el gran coste computacional del entrenamiento de dicha red neuronal e incluso problemas de *overfitting*. Como solución a este problema, siguiendo la página 398 del libro "*Manual Statistical Learning*"[21], se establece que uno de los métodos más comunes es el *weighting decay* o decadencia de los pesos. Muy parecido a la regularización *Ridge*, se define la función de pérdida regularizada como: $J(\theta) + \lambda R(\theta)$, donde, en el caso de clasificación: $J(\theta) = -\sum_{i=1}^N \sum_{k=1}^K y_{ik} \log f_k(x_i)$ y $R(\theta) = \sum_{k,s} \beta_{ks}^2 + \sum_{s,l} \alpha_{sl}^2$, siendo β_{km} pesos de conexiones entre capa de entrada y capa oculta y α_{sl} pesos de conexiones entre capa oculta y de salida, seguido del parámetro de regularización λ .

Por otra parte, se tienen las **redes neuronales recurrentes**[10], que se caracterizan por utilizar datos secuenciales o series temporales. Una gran diferencia con respecto a la arquitectura *feed forward* es que obtienen información de entradas anteriores, es decir, permiten mantener una memoria o información interna. Otra diferencia consiste en la compartición de parámetros en cada capa de la red. También, el algoritmo que utilizan este tipo de redes es el de retropropagación a través del tiempo (BPTT) para determinar los gradientes. Existen tres tipos de redes neuronales recurrentes:

- **Bidireccionales(BRNN)**: Procesan secuencias de datos con capas de nodos ocultos hacia delante y hacia atrás.
- **Memoria a corto-largo plazo**: Permite al modelo aumentar la capacidad de memoria para adaptarse a una línea de tiempo más larga.
- **Unidades recurrentes cerradas**: Trata de conseguir una retención selectiva de la memoria. Destacan la puerta de restablecimiento y puerta de actualización para controlar la cantidad de información que se retiene.

Por último, se tiene las **redes neuronales convolucionales**. Estas redes potencian las tareas de reconocimiento de imágenes y *computer vision*, campo de la inteligencia artificial que permite a los ordenadores y los sistemas obtener información significativa de imágenes digitales, vídeos y otras entradas visuales. Son muy útiles en la **detección de objetos, reconocimiento de imágenes, de voz e incluso reconocimiento facial** para la identificación de las personas.

Se caracterizan por su composición de capas:

- **Capa convolucional**, donde se aplican filtros o kernels en el detector de características.
- **Capa de agrupación o submuestreo**, que se encarga de reducir la dimensión mediante la reducción del número de parámetros de la entrada, la complejidad y el sobreajuste.
- **Capa totalmente conectada**, que se encarga de realizar las tareas de clasificación. Además, en esta capa, cada nodo de la capa de salida sí está conectado directamente

a un nodo de la capa anterior.

Árboles de decisión

Antes de comprender el algoritmo de bosques aleatorios, dado que se componen de árboles de decisión, es necesario explicar este concepto previamente para poder entender dicha técnica. Los árboles de decisión [11], que se pueden utilizar tanto para problemas de regresión como de clasificación, buscan encontrar la mejor división para crear subconjuntos de datos. Aunque pueden ser propensos a problemas, como sesgos y sobreajuste, cuando varios árboles de decisión forman un conjunto en el algoritmo de *random forest*, se logran predicciones más precisas.

Todos los algoritmos basados en árboles de decisión básicamente generan una estructura de 'árbol' que utilizan condiciones que constituyen los nodos de decisión en el árbol, actuando como un medio para dividir los datos y permitir alcanzar el valor buscado. En la primera imagen de la figura 3.6, se representa de una forma visual la estructura de un árbol de decisión, mientras que en la segunda imagen de la misma figura 3.6, se puede observar cómo se divide el espacio de características en regiones distintas y no superpuestas. Estas regiones se construyen a partir de los cortes que se realizan de manera recursiva en dicho espacio, dividiéndose este en regiones más pequeñas con el objetivo de maximizar la pureza de las clases en los nodos resultantes, es decir, separar en conjuntos homogéneos basados en la variable de entrada más significativa. Estas divisiones que se realizan se basan en umbrales o condiciones según cada característica. El proceso de división comienza por un nodo raíz que equivale al espacio de *inputs*, que se va subdividiendo en nodos intermedios. Cuando un nodo no se subdivide más, se convierte en una hoja. Este proceso de ramificación utiliza medidas de impureza para maximizar la homogeneidad de los nodos resultantes. A continuación se muestra la Figura 3.6 mencionada anteriormente con un ejemplo de estructura de árbol de decisión y su correspondiente partición en regiones del espacio de características explicado.

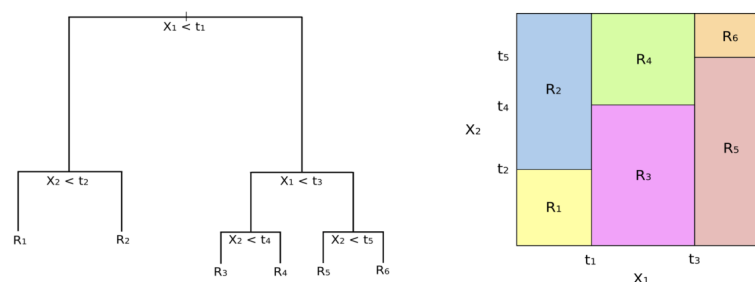


Figura 3.6. Cortes en un árbol de decisión y partición en regiones del espacio de características. Fuente: [11]

Más específicamente, otro aspecto importante y a tener en cuenta en el proceso de construcción de los árboles de decisión es la ramificación eficiente del árbol, es decir, cómo seleccionar las divisiones que mejor separen los datos en distintas clases y de la manera más óptima. Surgen algunos conceptos que proporcionan información en dicha construcción de los árboles de decisión como son los siguientes:

En primer lugar, cabe destacar el concepto definido por Shanon en 1948 conocido como

entropía cruzada, o en inglés *cross entropy*. Como ya establece el libro *Manual Statistic Learning*[21], se trata de una medida de evaluar la impureza de los nodos, es decir, la calidad de las divisiones de dichos nodos. La fórmula correspondiente a la entropía se puede ver en la Ecuación 3.1, donde \hat{p}_{sk} representa la proporción de observaciones del nodo s que pertenecen a la clase k , siendo K el número total de clases, fórmula que se puede encontrar en el mismo libro *Manual Statistic Learning*[21]. Si un nodo es puro, es decir, contiene únicamente observaciones de una clase, su entropía será cero.

$$Entropy = - \sum_{k=1}^K \hat{p}_{sk} \log(\hat{p}_{sk}) \quad (3.1)$$

Otro concepto es el conocido como **ganancia de información**, o más comúnmente conocido en inglés como *information gain*, que consiste en otra medida para calcular qué atributo proporciona la mejor división de los datos. Su fórmula matemática se muestra en la Ecuación 3.2, donde $H(S)$ es la entropía mencionada anteriormente del conjunto de datos S , $|S_v|$ es el número de instancias en S con valor v para el atributo A , $|S|$ número de instancias de S y $H(S_v)$ es la entropía del subconjunto de datos S_v .

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot H(S_v) \quad (3.2)$$

Una vez construido el árbol a partir del conjunto de datos de entrenamiento, es posible realizar predicciones recorriendo el árbol desde la raíz hasta una hoja, que es donde se aplica el criterio para la predicción, como la clase mayoritaria en dicha hoja. Evaluando las condiciones en cada nodo y siguiendo las ramas correspondientes se consigue una predicción final.

Se puede obtener el orden la importancia de las características[13] y así poder determinar cuál de ellas generan las mejores particiones o cortes y cuáles no son tan útiles.

Otro concepto a tener en cuenta es la importancia de los hiperparámetros para poder ser utilizados como criterios de parada para detener el crecimiento del árbol. Algunos de los **hiperparámetros** más importantes en árboles de decisión son los siguientes:

- **Mínimo número de muestras:** Este hiperparámetro se trata de un criterio de parada definido como la cantidad mínima de observaciones que debe tener un nodo para poder realizar su división. Un valor bajo del mínimo número de muestras puede resultar en una gran complejidad de los árboles debido a las pocas muestras por nodo, mientras que un valor alto puede resultar en árboles más simples que pueden tener el riesgo de subajuste.
- **Profundidad máxima:** La profundidad máxima es otro criterio de parada del crecimiento de un árbol que consiste en el número de niveles que existen en el árbol desde el inicio hasta el nodo más profundo. Además, si no se especifica, puede llegar a las hojas que no contienen nada o un mínimo número de muestras como criterio de parada. Es un factor muy importante que puede prevenir el sobreajuste.

3.2.2. Algoritmo de random forest

Este algoritmo de bosque aleatorio es una extensión del método de **bagging** [12], que se trata de una técnica para mejorar la precisión de los árboles de decisión. Para la explicación de **bagging**, es necesario explicar el **bootstrap aggregating**, que consiste en promediar modelos similares creados sobre datos ligeramente diferentes obtenidos mediante **bootstrapping**. Este **muestreo con reemplazo** o **bootstrapping** consiste en una técnica en la que se toman observaciones al azar del *dataset* original y una misma fila de esas observaciones del conjunto de datos podrá aparecer varias veces en el subconjunto que se está creando, de ahí el término de reemplazo. Las muestras *bootstrap* que se generan al hacer **bagging** introducen un elemento de aleatoriedad que en la práctica provoca que todos los árboles sean distintos, pero en ocasiones no son lo suficientemente distintos debido a la existencia de esa correlación entre árboles. Una vez dividido el conjunto de datos en estos subconjuntos aleatorios o muestras *bootstrap*, se entrena el modelo en cada subconjunto y se combinan todos los resultados de los modelos.

Random forest es a su vez la combinación de **bagging** y la selección aleatoria de características. El objetivo es que los árboles que se promedian estén poco correlados, ya que así se reduce la varianza de dicho promedio. En la introducción de la aleatoriedad en la selección de características, cada corte en un árbol de decisión se realiza utilizando un conjunto de características aleatorias diferente. Dada esa selección aleatoria de variables, se evalúa cuál de ellas es la mejor opción para hacer la división en cada nodo del árbol y se selecciona la mejor característica o variable igual que se explicó en los árboles de decisión con el objetivo de reducir la correlación de los árboles y mejorar la generalización. La Figura 3.7 muestra un ejemplo de la arquitectura y funcionamiento de este algoritmo.

En *random forest* también se deben tener en cuenta los hiperparámetros, más allá de los explicados en los árboles de decisión que se identificaban como criterios de parada en el crecimiento de la construcción de los árboles. Un hiperparámetro importante en bosques aleatorios es el **número o cantidad de árboles** que se utilizarán. Un mayor número de árboles puede aumentar la robustez del modelo, pero también puede aumentar el riesgo de sobreajuste. Otro de ellos es la **cantidad de características muestradas**. El número de características que se tienen en cuenta para buscar la mejor división es otro hiperparámetro importante que puede ayudar a la ya mencionada decorrelación de árboles y mejora de la generalización del modelo.

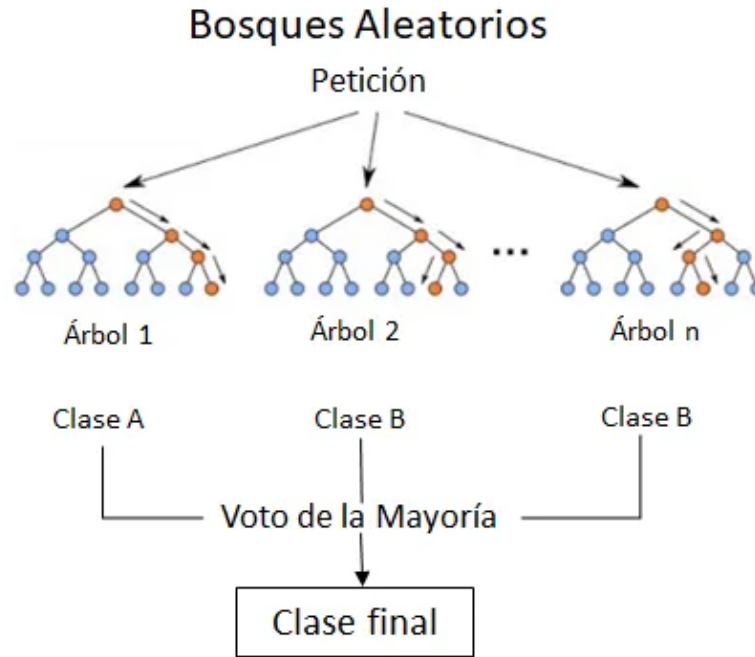


Figura 3.7. Ejemplo de arquitectura y funcionamiento de *random forest* con n árboles de decisión. Fuente: [24]

3.2.3. Regresión logística

La regresión logística es un ejemplo de modelo lineal generalizado que tiene como función de enlace la función logit, que se muestra en la Ecuación 3.3, siendo p la probabilidad de que pertenezca a la clase positiva.

$$\text{logit} : (0, 1) \rightarrow \mathbb{R}, \quad \text{logit}(p) = \log \left(\frac{p}{1-p} \right) \quad (3.3)$$

Se cumple la igualdad mostrada en la siguiente Ecuación 3.4.

$$\log \left(\frac{p}{1-p} \right) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_k x_k \quad (3.4)$$

siendo $\theta_0, \theta_1, \dots, \theta_k$ los coeficientes de regresión. La **regresión logística** es un modelo que tiene como objetivo predecir la probabilidad de que una observación pertenezca a una de las clases existentes. Se divide en regresión logística binaria, donde: $y \in \{0, 1\}$ y regresión logística multinomial, donde la variable dependiente y puede tomar más clases.

La regresión logística utiliza la distribución de **Bernoulli** para modelar la probabilidad de éxito en un evento binario. En dicha distribución, la variable aleatoria toma el valor de 1 con probabilidad p (éxito) y el valor de 0 con probabilidad $1 - p$ (fracaso), que se expresa como viene a continuación en la Ecuación 3.5.

$$P(y_i = 1) = p_i, P(y_i = 0) = 1 - p_i, \quad i = 1, \dots, m \quad (3.5)$$

La función de masa de probabilidad de cada y_i viene dada como muestra la Ecuación 3.6.

$$p_i^{y_i} (1 - p_i)^{1-y_i} \quad i = 1, \dots, m \quad (3.6)$$

El valor esperado de esta respuesta Bernoulli y_i se expresa en la Ecuación 3.7, siendo y_i , $i = 1, \dots, n$, la variable aleatoria Bernoulli.

$$E[y_i] = 1 \cdot p_i + 0 \cdot (1 - p_i) = p_i \quad (3.7)$$

La hipótesis de la regresión logística se expresa como viene en la Ecuación 3.8. Esta se trata de una función conocida como **función sigmoide**, donde x es el vector de características, θ el vector de parámetros y $\sigma(z)$ la probabilidad de pertenecer a la clase, como se ha mencionado anteriormente.

$$\sigma(z) : \mathbb{R} \rightarrow [0, 1], \quad \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = \theta^T x \quad (3.8)$$

Como apunte, en regresión logística el objetivo es predecir la probabilidad de que una observación pertenezca a cada clase. Al tratarse de una probabilidad, el conjunto de llegada de la función sigmoide es el intervalo $[0,1]$. Se cumple la siguiente expresión en la Desigualdad 3.9.

$$0 \leq \sigma(z) \leq 1 \quad (3.9)$$

En la Figura 3.8, se vuelve al concepto del umbral decisión explicado anteriormente, donde la primera flecha naranja representa la situación en la cual el modelo predice $y = 0 \Leftrightarrow \sigma(z) < 0.5$, es decir si $P(\theta^T x) < 0.5$.

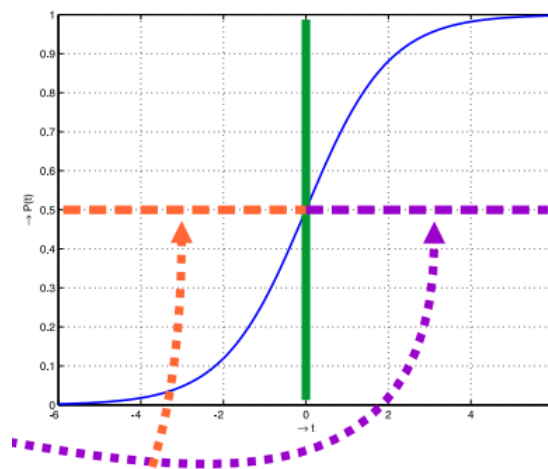


Figura 3.8. Función Sigmoide

El modelo predice $y = 0 \Leftrightarrow \sigma(z) < 0.5$, es decir si $P(\theta^T x) < 0.5$.

La segunda flecha (color morado) ocurre cuando el modelo predice $y = 1 \Leftrightarrow \sigma(z) \geq 0.5$, es decir si $P(\theta^T x) \geq 0.5$.

El modelo predice $y = 1 \Leftrightarrow \sigma(z) \geq 0.5$, es decir si $P(\theta^T x) \geq 0.5$.

También, destaca la importancia del **método de máxima verosimilitud** que estima los parámetros θ . La función de verosimilitud es la siguiente expresión definida en la Ecuación 3.10.

$$L(\theta) = \prod_{i=1}^m \frac{1}{1 + e^{-\theta^T x_i}} \quad (3.10)$$

donde x_i representa un vector de características y θ el vector de parámetros mencionado anteriormente. El siguiente paso consiste en aplicar logaritmos a ambos miembros de la ecuación 3.10. De esta manera, se obtiene la función log-verosimilitud negativa(NLL) definida en la Ecuación 3.11.

$$NLL(\theta) = -\frac{1}{m} \sum_{i=1}^m [y_i \log(\sigma(\theta^T x_i)) + (1 - y_i) \log(1 - (\sigma(\theta^T x_i)))] \quad (3.11)$$

siendo $\sigma(\theta^T x_i)$ la probabilidad predicha de la i -ésima observación y m el número de muestras de entrenamiento. A partir de este paso, se utilizarían algoritmos de optimización para encontrar los hiperparámetros óptimos. La función de coste se expresa de manera similar como en la Definición 3.12, siendo σ la función sigmoide antes definida en la Ecuación 3.8.

$$J(\theta) = \begin{cases} -\log(\sigma(\theta^T x)) & \text{si } y = 1, \\ -\log(1 - \sigma(\theta^T x)) & \text{si } y = 0. \end{cases} \quad (3.12)$$

Como $y \in \{0; 1\}$, esta expresión se puede simplificar como se muestra en la Ecuación 3.13, siendo m el número de muestras de entrenamiento, $y^{(i)}$ la etiqueta real de la instancia i , y $\sigma(\theta^T x_i)$ la predicción del modelo para la instancia i .

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\sigma(\theta^T x_i)) + (1 - y^{(i)}) \log(1 - \sigma(\theta^T x_i))] \quad (3.13)$$

Además, se puede utilizar la **regularización** añadiendo un término/parámetro a la función de coste $J(\theta)$ para que el modelo no tenga errores en la generalización y evitar el sobreajuste. Si se trata de la regularización *Ridge* o L_2 , se tendría la Ecuación 3.14, donde α es ese parámetro de regularización adicional.

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\sigma(\theta^T x_i)) + (1 - y^{(i)}) \log(1 - \sigma(\theta^T x_i))] + \frac{\alpha}{2m} \sum_{j=1}^n \theta_j^2 \quad (3.14)$$

Si se trata de la regularización *Lasso* o L_1 , se tendría la Ecuación 3.15:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(\sigma(\theta^T x_i)) + (1 - y^{(i)}) \log(1 - \sigma(\theta^T x_i)) \right] + \frac{\alpha}{m} \sum_{j=1}^n |\theta_j| \quad (3.15)$$

3.3 Métricas

Las métricas son herramientas fundamentales en el campo del aprendizaje automático que nos permiten evaluar el rendimiento y la eficacia de nuestros modelos predictivos. Proporcionan una medida cuantitativa del desempeño del modelo, permitiendo comprender su comportamiento en términos de precisión, error y otras dimensiones relevantes para el problema en cuestión. La selección adecuada de las métricas es esencial para entender cómo se está desempeñando el modelo en relación con los objetivos del trabajo. Dependiendo del tipo de problema que se esté abordando, existen métricas específicas diseñadas para evaluar diferentes aspectos del rendimiento del modelo. Por ello, se van a enumerar y explicar las principales métricas[14] según la tarea de aprendizaje supervisado, es decir, se agruparán las métricas correspondientes a la regresión y por otro lado, se comentarán análogamente las métricas correspondientes a la clasificación.

3.3.1. Métricas de regresión

En primer lugar, para algoritmos de regresión se suelen utilizar métricas como:

- **Error Absoluto Medio (MAE):** Mide la diferencia media absoluta entre las predicciones del modelo y los valores reales. A continuación se muestra su fórmula correspondiente:

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

donde \hat{y}_i es el valor predicho por el modelo, mientras que y_i es el valor real de la variable de respuesta. Es útil cuando se desea evaluar el rendimiento del modelo en términos de la magnitud del error y no se desea penalizar en exceso los valores atípicos (*outliers*) o errores extremos.

- **Error Cuadrático Medio (MSE):** Mide el promedio de las diferencias al cuadrado entre las predicciones del modelo y los valores reales.

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

Al igual que el error absoluto medio, esta métrica de regresión es sensible a valores atípicos.

3.3.2. Métricas de clasificación

En problemas de clasificación, existen diferentes métricas para evaluar el rendimiento de los modelos. Cada métrica evalúa una dimensión o criterio diferente de la calidad de un clasificador. Más específicamente, en la clasificación binaria, suele haber una clase que representa el evento de interés, la positiva. Algunas métricas son las siguientes:

- **Matriz de confusión:** Se trata de una matriz de dimensión 2x2 que describe el rendimiento de un clasificador mostrando los resultados de las predicciones del modelo en

comparación con los valores reales. Las entradas de la matriz son:

- **Verdaderos Positivos (TP):** Cuando la clase real de una observación era positiva y la predicha también lo es.
- **Verdaderos Negativos (TN):** Cuando la clase real de la observación era negativa y el predicho también es negativo.
- **Falsos Positivos (FP) o Error tipo I:** Un falso positivo ocurre cuando el modelo predice incorrectamente que una muestra pertenece a una clase positiva cuando en realidad pertenece a la clase negativa.
- **Falsos Negativos (FN) o Error de tipo II:** Cuando la clase real del punto de datos era positiva y el valor predicho pertenece a la clase negativa.

En la Figura 3.9 se muestra un ejemplo de matriz de confusión:

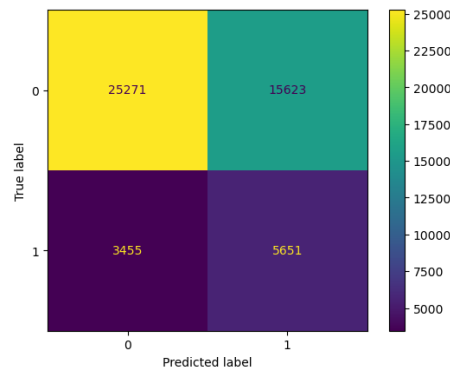


Figura 3.9. Ejemplo de matriz de confusión.

En la Figura 3.9, la diagonal de la matriz representa TP y FN, respectivamente, mientras que la segunda columna de la primera fila corresponde a FN y la primera columna de la segunda fila a FP. Una vez explicados estos cuatro términos, ahora se utilizarán para dar lugar a las diferentes métricas o conceptos relacionados con dicha matriz de confusión:

- **Precisión:** Mide la proporción de instancias clasificadas como positivas que son verdaderamente positivas. Se trata de estimar $P(\text{Clasificación correcta} \mid \text{Ejemplo predicho positivo})$, es decir, estimar la probabilidad de la clasificación correcta de una instancia condicionado a que el ejemplo predicho es positivo. Su fórmula es la siguiente:

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

- **Recall (Sensibilidad):** También conocido como tasa de verdaderos positivos. Mide la proporción de instancias positivas reales que fueron correctamente identificadas por el modelo. Esto estima la probabilidad $P(\text{Clasificación correcta} \mid \text{Ejemplo es positivo})$. Se calcula como:

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

- **Especificidad:** Es la tasa de verdaderos negativos, es decir, es una medida del rendimiento de un modelo de clasificación que mide la proporción de instancias negativas reales que fueron correctamente identificadas por el modelo como negativas. Se trata

de estimar la probabilidad condicionada $P(\text{Clasificación correcta} \mid \text{Ejemplo es negativo})$. La fórmula de la especificidad es la siguiente:

$$\text{Especificidad} = \frac{TN}{TN + FP}$$

- **Valor Predictivo Negativo (True Negative Value):** Mide la proporción de los datos verdaderamente negativos clasificados como instancias negativas. Esto estima la probabilidad $P(\text{Clasificación correcta} \mid \text{Ejemplo predicho es negativo})$. Su calcula de la siguiente manera:

$$VPN = \frac{TN}{TN + FN}$$

- **Exactitud:** Otra medida que consiste en medir la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones realizadas. Su fórmula es la siguiente:

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

- **Puntuación F1 (F-Score):** La puntuación F1 es la media armónica de precisión y *recall*, y se calcula de la siguiente manera:

$$F1 = 2 \times \frac{\text{precisión} \times \text{recall}}{\text{precisión} + \text{recall}}$$

Además, como consecuencia de utilizar la media armónica en lugar de la media aritmética, se penalizan más severamente los valores bajos.

- **Curvas ROC y AUC:** La curva ROC (Característica Operativa del Receptor) es una representación gráfica que muestra la relación entre la tasa de verdaderos positivos (*recall*) y la tasa de falsos positivos (1 - especificidad) para diferentes umbrales de decisión, como ya se comentó en la Sección 3.1.1. En el eje X se representa la tasa de falsos positivos y en el eje Y se representa la tasa de verdaderos positivos. El valor de AUC ("Área bajo la curva") varía entre 0 y 1, donde un valor de 1 indica un modelo perfecto que puede distinguir perfectamente entre clases positivas y negativas. En la Figura 3.10 se muestra un ejemplo:

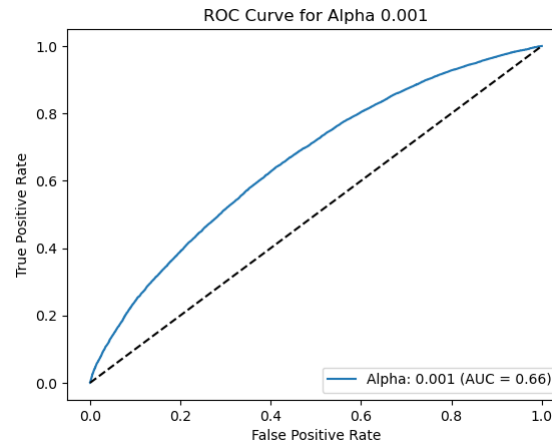


Figura 3.10. Ejemplo de curva ROC y su correspondiente valor AUC.

Por último, en un problema de clasificación desbalanceado[56], como puede ocurrir entre las clases positivas y negativas de la predicción de titularidades, donde la clase positiva corresponde a predecir que el jugador es titular y negativa suplente, algunas métricas pueden dar una información incorrecta y dar la sensación de un clasificador de alta calidad cuando en realidad la clase minoritaria no se está prediciendo correctamente. Por ejemplo, si se mide la efectividad de un modelo por la cantidad de aciertos solo teniendo en cuenta la clase mayoritaria, se obtendrá un valor alto. Sin embargo, es simplemente una falsa sensación de que el modelo funciona bien. Debido a ese desequilibrio en las clases, métricas como la precisión serán afectadas porque darán valores muy bajos para la clase minoritaria y altos para la mayoritaria. Esto indica que no se ha conseguido un correcto aprendizaje debido a esa insuficiencia de muestras pertenecientes a la clase minoritaria. En un caso de clasificación binaria, existen cuatro posibles opciones para cada clase: Si las métricas de precisión y *recall* son altas, el modelo maneja bien esa clase, mientras que si la precisión es alta y *recall* bajo, el modelo no detecta bien la clase pero es confiable cuando lo logra. Si la precisión es baja y el *recall* alto, la clase se detecta bien pero incluye observaciones de otras clases. Finalmente, si ambos valores son bajos, el modelo no logra identificar la clase correctamente. Algunas métricas son especialmente útiles cuando hay un desequilibrio entre las clases en el conjunto de datos, como la puntuación F1. Esta métrica encuentra ese equilibrio entre la precisión y el *recall*. Una métrica que no se ve afectada por cambios en la distribución de clases o en la escala de los valores de predicción es el AUC-ROC (área bajo la curva ROC). Por lo tanto, es una métrica de evaluación útil cuando se trabaja con conjuntos de datos desequilibrados.

Capítulo 4. Metodología

En este capítulo, se detallan las etapas de la metodología que se ha utilizado en el proyecto. Estas se dividen en las siguientes secciones:

- **Extracción de los datos**
- **Análisis exploratorio de los datos**
- **Diseño de la arquitectura del sistema**
- **Desarrollo e implementación**

4.1 Extracción de los datos.

En primer lugar, para la obtención de los datos y estadísticas de todos los jugadores de todos los partidos de la temporada 2023/24 de la liga española de fútbol, requirió un gran esfuerzo para la extracción de toda esta información. Para ello, se empleó la conexión a una API que recolecta datos de todos los deportes llamada *API-Football*[45]. Esto implicó una gran cantidad de tiempo para la extracción de todos los datos y se realizó un análisis con información de los partidos ya jugados para verificar que se trataban de datos fiables y reales. Esto se completó mediante la comparación de estadísticas de jugadores en determinados partidos como el número de minutos, goles, asistencias, entre otros, y así concluir que eran válidos para trabajar.

La obtención de los datos se guardó en varios *datasets*. El primero trata de toda la información relacionada con cada partido, incluyendo la fecha del evento, los equipos participantes, el resultado del partido y el estadio donde se disputó. El segundo *dataset* contiene toda la información posible sobre los jugadores de la competición a lo largo de todos los partidos de la temporada. De esta manera, cada instancia o fila de datos corresponde a las estadísticas pertenecientes a un jugador específico en un partido concreto, incluyendo goles, asistencias, minutos jugados y muchas otras variables que se analizan para enriquecer esta base de datos y posteriormente a los modelos de predicción.

4.2 Análisis exploratorio de los datos.

Para profundizar en la exploración de los datos extraídos y conocer todos los factores que influyen en el rendimiento de un jugador o si será titular o no, en la Tabla 4.1 y Tabla 4.2 se muestra la descripción de las variables explicativas y variables objetivo de ambos *datasets*.

Tabla 4.1. Variables del dataset de partidos temporada 2023/24

VARIABLE	DESCRIPCIÓN
ID Partido	Identificador del partido
Fecha	Fecha del partido
Equipo local	Equipo que jugó en su estadio (local)
Equipo visitante	Equipo que jugó fuera de su estadio (visitante)
Goles local	Nº de goles del equipo local en el partido
Goles visitante	Nº de goles del equipo visitante en el partido
Estadio	Nº de goles del equipo visitante en el partido

La Tabla 4.2 muestra el *dataset* de partidos describiendo cada variable.

Tabla 4.2. Variables del dataset de jugadores temporada 2023/24

VARIABLE	DESCRIPCIÓN
ID jugador	Identificador único de cada jugador
ID partido	Identificador único de cada partido
Nombre	Nombre completo del jugador
Posición	Posición habitual del jugador en el campo
Goles	Número total de goles anotados por jugador en el partido
Asistencias	Número total de asistencias realizadas por jugador en el partido
Minutos	Total de minutos disputados por jugador en el partido
Equipo	Nombre del equipo en el que juega el jugador
Titularidad	Variable binaria si el jugador es titular o no en el partido
Tarjetas amarillas	Cantidad total de tarjetas amarillas recibidas por jugador en el partido
Tarjetas rojas	Cantidad total de tarjetas rojas recibidas por el jugador.
Fuera de juegos	Número de veces que el jugador cometió posición de fuera de juego.
Tiros totales	Total de disparos realizados por el jugador
Tiros a portería	Número de disparos a puerta
Duelos totales	Cantidad de enfrentamientos físicos o técnicos entre el jugador y oponentes.
Duelos ganados	Número de duelos en los que el jugador salió con el control del balón
Intentos de regates	Número de veces que el jugador intentó regatear a un oponente
Regates exitosos	Número de regates completados con éxito
Faltas cometidas	Número total de faltas cometidas por el jugador
Faltas provocadas	Número de faltas que recibió el jugador
Pases precisos	Número de pases completados correctamente
Pases totales	Número total de pases realizados por el jugador
Entradas totales	Número de acciones defensivas ('tackles').
Entradas bloqueadas	Entradas que interrumpieron el avance del rival
Intercepciones	Número de veces que el jugador interceptó con éxito un pase del equipo rival.
Penalti fallado	Número de penaltis lanzados que no acabaron en gol.
Penalti marcado	Número de penaltis convertidos exitosamente en gol.
Penalti parado	Número de penaltis detenidos por el jugador (aplicable a porteros).
Rendimiento del jugador	Puntuación que refleja el desempeño del jugador (escala 1-10)

Como ya se comentó en la Sección 1.3, donde se explicaron los principales objetivos del trabajo, se utilizaron modelos de aprendizaje automático para obtener predicciones del rendimiento de jugadores, estados de forma y titularidades de un jugador. Para su predicción, en

el primer caso la variable objetivo es la variable 'rendimiento'. Esta se trata de una variable numérica con escala del 1 al 10, donde 1 representa la puntuación más baja del rendimiento de un jugador en un partido y 10 en la máxima puntuación que se puede obtener. Esta se basa en todo el resto de estadísticas de un jugador para poder calcular una nota sobre 10 de cada jugador. También, para la predicción del estado de forma de un jugador la variable objetivo sigue siendo la variable 'rendimiento'. En este caso, el cálculo del estado de forma en porcentaje se consigue dividiendo el rendimiento del jugador en el partido entre su máximo rendimiento en la temporada (estado *prime* del jugador) y multiplicado por 100, como se muestra en la Ecuación 4.1.

$$\text{Estado de forma (\%)} = \frac{\text{Rendimiento del jugador}}{\text{Máxima puntuación del jugador en la temporada}} \times 100 \quad (4.1)$$

Por último, la predicción de la variable 'Titularidad' consiste en un problema de clasificación binario. El valor de 1 indica que el jugador es titular en ese partido y 0 suplente. Para ello, se incluyeron en los modelos todas las variables explicativas, incluyendo algunas variables extras como el rendimiento de los jugadores en los últimos 5 partidos recientes a ese partido actual, número de goles en los últimos 5 partidos recientes y otras análogas. Así, se obtuvieron mejores resultados en las métricas.

4.3 Diseño de la arquitectura del sistema

Como ya se explicó anteriormente en la Sección 1.3, este trabajo se divide en dos grandes bloques: la utilización de modelos de aprendizaje automático para facilitar la predicción del rendimiento de los jugadores y titularidades y la construcción de un agente de IA generativa para la consulta inteligente de la información.

Estos dos principales bloques están reflejados en una interfaz web interactiva mediante el uso de Streamlit permitiendo a los usuarios elegir si desean realizar predicciones o consultar información específica sobre detalles de los jugadores, partidos o estadísticas concretas.

Por un lado, el primer bloque conlleva el uso de conexión vía API para la extracción de los datos, explicada en la Sección 4.1. Posteriormente, se requiere un análisis exploratorio profundo y procesamiento previo de la información, que se explican detalladamente en la Sección 4.4. Por último, la elección del uso de los modelos de aprendizaje automático y sus evaluaciones mediante métricas conforma este primer gran bloque del proyecto. A continuación, en la Figura 4.1 se muestra el flujo de arquitectura de este primer gran bloque:

Por otro lado, para la implementación del segundo bloque es necesario un análisis de distintos *frameworks* de agentes, como *langchain* o el recién publicado por Google, ADK (*Agent Development Kit*). Finalmente se optó por la utilización del *framework de langchain*. Una vez definida la plataforma de trabajo, se estudió las distintas opciones de arquitecturas y se eligió la más sencilla. Esta consiste en un único agente que en función de la petición del usuario, tiene la capacidad de elegir las herramientas que debe utilizar dependiendo de la tarea requerida. El agente se compone de tres componentes principales, como se muestra en la Figura 4.2: un **prompt específico**, el modelo de lenguaje (**LLM**) y las herramientas (**tools**)

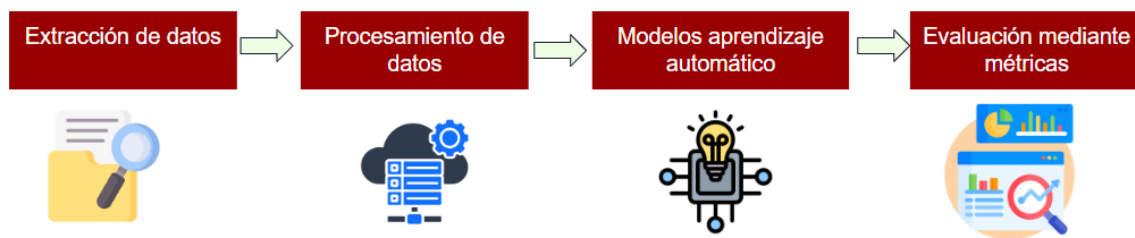


Figura 4.1. Arquitectura y flujo de predicción del rendimiento de jugadores. Fuente: Elaboración propia

específicas asignadas al agente.

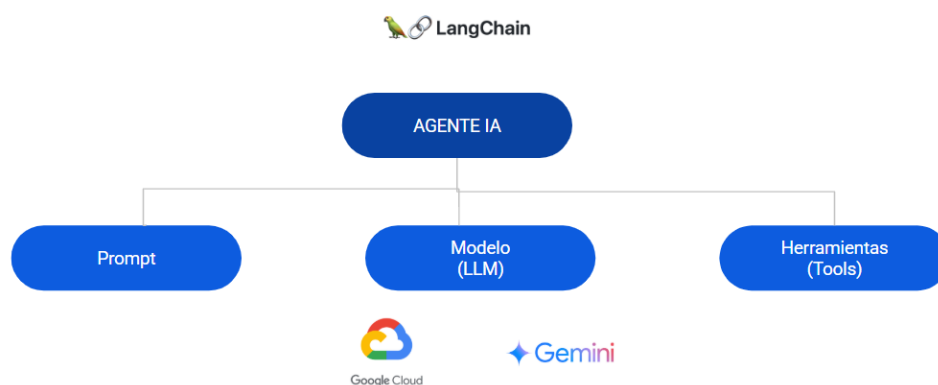


Figura 4.2. Componentes principales de la arquitectura del segundo gran bloque. Fuente: Elaboración propia.

Para el primer componente se requieren técnicas de ingeniería de prompt (**prompt engineering**). Algunas técnicas que se utilizaron son las llamadas 'One hot' o 'Few shot', que consiste en definir uno o algunos ejemplos de tareas que debe realizar el agente y cómo debe llevarla a cabo siguiendo el flujo proporcionado e incluso incluyendo el formato de salida que debe mostrar por pantalla. A continuación se muestra el prompt sin incluir los ejemplos:

'Eres un analista experto en fútbol y un maestro de bases de datos especializado en sqlite3. Los datos con los que trabajas se extraen de archivos CSV ubicados en la carpeta «csv». Basándote en los nombres de las columnas, comprende la estructura de cada tabla e identifica las posibles relaciones entre ellas. Cuando obtengas la información solicitada, configura solo las tablas que sean necesarias, no vuelvas a crear tablas que ya existen. Utiliza la siguiente instrucción SQL para comprobar qué tablas están configuradas: SELECT name FROM sqlite_master WHERE type='table'; Una vez que hayas recopilado la información necesaria utilizando una herramienta, responde con claridad y no invoques otras herramientas innecesariamente. Utiliza herramientas solo si no puedes determinar la respuesta con la información de la que dispones.'

Para el segundo componente del agente, el LLM, se estudiaron las distintas opciones de

modelos eficientes, económicos y que tengan la capacidad de realizar este tipo de tareas. Finalmente, se decidió utilizar la infraestructura *cloud* de *Google Cloud* y se analizaron las ventajas y desventajas de los modelos en el *Model Garden* del servicio de *Vertex AI* de *Google Cloud*. En concreto, el modelo utilizado es '**gemini-2.0-flash**' debido a su gran eficiencia y costes bajos.

Para el tercer componente y completar la implementación de esta arquitectura es necesario también definir la base de datos donde se almacena y consulta el agente toda la información y proporcionar al agente una serie de herramientas que se explican a continuación.

En primer lugar, todos los CSV de partidos y de jugadores de las temporadas disponibles se almacenan en una base de datos **SQLite**. EL agente recibe cuatro herramientas específicas que utiliza cuando procese la petición del usuario y decida cuál es necesaria para la realización de la tarea. La primera función que se ha implementado consiste en listar todos los archivos que se tienen en una carpeta donde se almacenan todos los CSV. De hecho, esta función devuelve todos los nombres de los cvs disponibles donde el agente va a poder buscar información o un mensaje de error en caso de no existir ningún archivo disponible. La segunda función lista todas las columnas de un CSV específico de la carpeta de archivos disponibles que devuelve la primera función. Análogamente, devuelve todos los nombres de las columnas o variables que se tienen en un CSV específico o un mensaje de error. La tercera función importa los archivos CSV a la base de datos SQLite como tablas. Esta función devuelve un resumen de las tablas exitosas que se han podido crear o un mensaje de error si ha ocurrido algún problema en cargar algunas de las tablas que se necesitan. La cuarta y última función ejecuta una petición en SQL a la base de datos que ya contienen las tablas. Esta petición debe comenzar siempre con la palabra 'SELECT' y devuelve un diccionario con los resultados de la petición o un mensaje de error. De esta forma, se consigue traducir la petición del usuario en mensaje natural a una petición en lenguaje SQL y poder realizar consultas a las tablas de la base de datos. Una vez se devuelve el resultado de la consulta, el agente lo procesa y vuelve a construir la respuesta en lenguaje natural con el resultado de la consulta realizada. Todas estas funciones o también conocidas en inglés, *tools* del agente, se muestran en el esquema de la Figura 4.3.

4.4 Desarrollo e implementación

Por un lado, en esta sección se explican los pasos y técnicas empleadas para la implementación de los modelos de aprendizaje automático correspondientes al primer gran bloque del proyecto. Como ya se comentó en la Sección 4.1, primero es necesaria la extracción de los datos de la competición española de fútbol, que se obtuvieron gracias a la conexión a una API. Posteriormente, estos datos necesitaban ser analizados y procesados.

4.4.1. Procesamiento de datos

En primer lugar, se comprobó la existencia de valores faltantes. Las variables de los jugadores como minutos, asistencias, goles, tarjetas, pases clave y muchas otras más columnas presentaban un gran número de valores faltantes. Tras una exploración con las estadísticas de los partidos reales de cada temporada, se concluyó que estos valores nulos correspon-

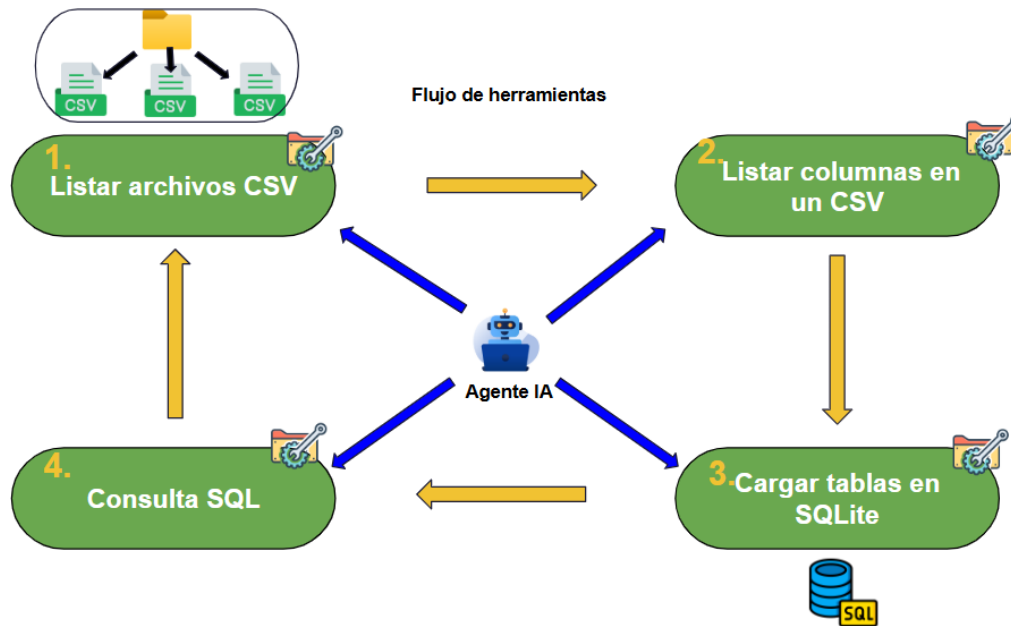


Figura 4.3. Flujo de herramientas proporcionadas al agente. Fuente: Elaboración propia.

dían a un valor de 0 de ese jugador en esa variable en ese determinado partido. En otras palabras, los jugadores que no habían disputado minutos, o no habían marcado ningún gol, o no habían realizado ningún pase clave en el partido, presentaban valores nulos en lugar de un valor de 0. Por esta razón, todas estas variables que presentaban valores faltantes en esas columnas específicas se rellenaron por valores de 0. Sin embargo, se analizó que la variable 'rendimiento' presentaba valores nulos en jugadores que sí habían disputado minutos en el partido. Tras una gran exploración, se comprobó que los jugadores que habían disputado menos de 7 minutos en el partido no tenían valoración de la actuación en ese partido. Esto sugirió múltiples opciones, como sustituir los valores faltantes por otros valores o eliminar estas instancias porque no aportaban información. Finalmente, se optó por la eliminación de las filas del *dataset* que correspondían a jugadores con menos de 7 minutos en el partido debido a la escasa aportación útil que proporcionaban, pues un jugador no podía ser evaluado con tan pocos minutos disputados en el partido.

Por otra parte, se creó una variable 'is_home' construida a partir de las variables de equipo local del *dataset* de partidos y la variable de equipo del *dataset* de jugadores. Esta variable creada presenta valores de 1 si el equipo juega como local y 0 si lo hace como visitante. Esto implicó la eliminación de la variable 'Estadio', 'Equipo local' y 'Equipo visitante' al disponer de esa información mediante la nueva variable creada. También se estudió la presencia de valores atípicos mediante la construcción de gráficos de diagramas de cajas, o en inglés, *boxplots*. Para la utilización de modelos como redes neuronales (MLP), se normalizaron las variables mediante la estandarización. Otras técnicas de procesamiento de datos que se utilizaron fue el uso de *One Hot Encoding* en las variables categóricas, concretamente en las variables de equipo, equipo rival y posición del jugador. Esta última variable generó las siguientes variables *dummies*:

- position_G: Futbolista que juega como portero (*G = Goalkeeper*)

- position_D: Futbolista que juega como defensa ($D = Defender$)
- position_M: Futbolista que juega como mediocentro ($M = Midfielder$)
- position_F: Futbolista que juega como delantero ($F = Forward$)

De esta manera, se tiene el valor de 1 si el jugador correspondiente pertenece a un equipo y 0 si no pertenece a ese equipo, y análogamente con la variable del equipo rival. Para la posición, se obtiene un 1 si la posición del jugador corresponde a portero, defensa, mediocentro o delantero y 0 para el resto de posiciones que no tenga asignado. En la Tabla 4.3 se muestra un ejemplo del jugador Djené, defensor del Getafe en un partido disputado contra el Elche.

Tabla 4.3. Ejemplo de un jugador tras técnica de One Hot Encoding

Jugador	team_getafe	opponent_elche	position_G	position_D	position_M	position_F
Djené	1	1	0	0	0	1

También se ha aplicado *Label Encoder* en la variable de '*player_name*', el nombre del jugador, para obtener un número asignado a cada jugador. Esto supone la eliminación de variables como el nombre del jugador y el ID del jugador. En la Tabla 4.4 se muestra un ejemplo de la codificación del nombre de los jugadores.

Tabla 4.4. Ejemplo de Label Encoder de varios jugadores

Player_name_encoded	Player_name
145	Vinícius Júnior
307	Robert Lewandowski

Otra técnica que se ha aplicado es la extracción del día de la semana y mes de la variable '*dateEvent*', correspondiente a la fecha del partido. Esto se ha realizado para estudiar si tenía influencia significativa en la predicción del rendimiento de los jugadores o titularidades. Así, como la fecha no corresponde a un dato numérico y no podía ser utilizado en el entrenamiento de los modelos, se aplicó la anterior técnica mencionada *Label Encoding* para asignar valores numéricos en orden en función del día de la semana, siendo Lunes con valor 1 y Domingo con valor 7 y análogamente con los meses, siendo Enero con valor 1 y Diciembre con valor 12. En la Tabla 4.5, se muestra un ejemplo del jugador Djené, defensor del Getafe, correspondiente a un partido del sábado 10 de diciembre.

Tabla 4.5. Ejemplo de Label Encoder para fecha de un partido

Player_name_encoded	Player_name	Day_of_week	Month
211	Djené	6	12

Por último, en la Figura 7.3 del capítulo 7, correspondiente al Apéndice del trabajo, se muestra la matriz de correlación. Esta matriz incluye todas las variables numéricas para el entrenamiento de los modelos. Es importante destacar que en la matriz de correlación no se han incluido las variables *dummies* correspondientes a los equipos a los que pertenece el jugador y al rival al que se enfrenta debido a que aumentaría la dificultad de presenciar los patrones en dicha matriz, pero sí han sido incluidas en el entrenamiento de los modelos. En dicha Figura 7.3 se puede identificar la correlación entre algunas variables, principalmente entre

las nuevas creadas para las estadísticas de los últimos cinco partidos previos hasta la fecha. Otras como las variables *dummies* de la posición de un jugador, tienen una correlación con la variable *starter* de titularidad, pues dependiendo de la posición que desempeñe el jugador, es un indicador que claramente influye en su titularidad.

4.4.2. Fuga de datos: Solución

Otro problema que se trató es la fuga de datos, o en inglés *data leakage* [60]. La fuga de datos, resumidamente, consiste en incluir datos a los modelos que no estarán disponibles cuando se realizan las predicciones. En este caso de uso, las variables de los jugadores en un partido como minutos, goles, asistencias y todas las demás estadísticas disponibles almacenadas de un jugador en un partido no pueden ser usadas en un partido que se quiere predecir el rendimiento del jugador o si será titular, pues se tratan de variables de un partido a predecir del cual aún no se sabe la actuación de ese jugador. Esto puede dar a un caso de sobreajuste, pues las métricas daban evaluaciones con valores muy altos pero no correspondía con una predicción real. Para resolver este problema, se optó por la siguiente solución. Esta consiste en el cálculo de estadísticas del jugador previas al partido correspondiente. Es decir, se calculó la media de minutos, goles, asistencias, duelos ganados, pases totales y todas las demás estadísticas de los últimos 5 partidos más recientes a la fecha de ese partido. Además, para las instancias que pertenecían a las 5 primeras jornadas, se calculó la media de estas variables con las jornadas anteriores disponibles hasta la fecha. De esta manera, se solucionó la fuga de datos y ya se obtuvieron variables que sí se podían incluir a los modelos para su predicción y que aportaban información útil y real al tratarse de las 5 últimas jornadas recientes a la fecha del partido.

Análogamente, no solo se calculó para los jugadores, sino también para los equipos. Para obtener el estado de forma previo de cada equipo en cada jornada, se calculó el estado de forma previo de todos sus jugadores en los últimos 5 partidos a la fecha del partido y la media móvil resultante correspondía a la media del equipo. Esto se tuvo que calcular para todas las jornadas, de ahí que se tratara de una media móvil dependiendo de la fecha del partido. Por último, se generó un conjunto de datos general para el conjunto de entrenamiento de los modelos que constaba del jugador correspondiente, el equipo, la fecha, el estado de forma previo del jugador a esa fecha, el rival, el estado de forma previo del rival a esa fecha y la variable objetivo, el rendimiento del jugador o la titularidad, dependiendo el caso. A continuación, en la Tabla 4.6, Tabla 4.7 y Tabla 4.8, se muestran los tres subconjuntos que se realizaron para poder solucionar la fuga de datos y dotar de todas las variables posibles a los modelos.

Tabla 4.6. Ejemplo conjunto de datos a nivel jugador

Jugador	Equipo	Fecha	Estado de forma previo*	Rival
Vinícius Júnior	Real Madrid	15/09/2023	92.5 %	Barcelona
Robert Lewandowski	Barcelona	14/11/2023	89.4 %	Sevilla

Tabla 4.7. Ejemplo conjunto de datos a nivel equipo

Equipo	Fecha	Estado de forma previo*
Real Madrid	08/05/2024	92.5 %
Valencia	05/01/2024	89.4 %

Tabla 4.8. Conjunto de datos general para los modelos.

Jugador	Equipo	Fecha	Estado de forma*	Rival	Estado de forma rival	Rendimiento
Jude Bellingham	Real Madrid	08/05/2024	92.5 %	Barcelona	88.2 %	8.4
Hugo Duro	Valencia	05/01/2024	89.4 %	Atlético Madrid	86.7 %	7.9

*La columna 'Estado de forma previo' no solo se trata de una única columna del rendimiento medio del jugador en los últimos partidos hasta la fecha, sino también más columnas como el número de goles en los últimos partidos recientes hasta ese partido, el número de asistencias en los últimos partidos más recientes hasta esa fecha y otras variables creadas en función de la fecha de cada partido.

4.4.3. División de datos: entrenamiento, validación y test

Una vez obtenido el conjunto de datos general preparados para poder ser utilizados en los modelos y así obtener las predicciones, se ha seguido una estrategia para realizar una evaluación justa y objetiva del rendimiento de los modelos de aprendizaje automático. Como ya se explicó en la Sección 3.1.4, esto permite encontrar el equilibrio adecuado entre la flexibilidad y la capacidad de generalización. Mediante la evaluación en un conjunto de datos independiente a los datos de entrenamiento, esto permite evaluar la capacidad de generalizar a nuevos datos y que no se ajuste excesivamente a las muestras de ese entrenamiento. Por esta razón, en este caso de uso, el conjunto de datos obtenido se ha dividido en **conjuntos de entrenamiento, validación y test**. En la siguiente Figura 4.4, se muestra la estrategia que se ha utilizado para realizar la división, siendo el conjunto de entrenamiento para entrenar los modelos de aprendizaje automático, el conjunto de validación para la optimización de los hiperparámetros de los modelos y el de *test* para evaluarlos en un conjunto de datos real. Tras numerosas pruebas previas, esta estrategia elegida es la que mejor rendimiento ha dado, resultando en un 60 % el conjunto de datos de entrenamiento, 20 % subconjunto de datos de validación y un 20 % el subconjunto de datos de test.

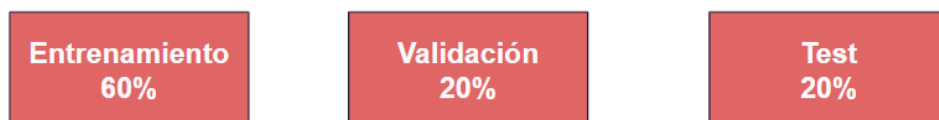


Figura 4.4. División de datos en entrenamiento, validación y test.

Siguiendo esta estrategia, el resultado final que se muestra en el código del proyecto con el *dataset* de las temporadas 2022/23 y 2023/24 y todas las variables utilizadas preparado

para utilizarse en el entrenamiento de los modelos de aprendizaje automático resulta de la siguiente forma, respetando el 60 % para los datos de entrenamiento, 20 % para validación y 20 % para el conjunto de *test*:

Train shape: (14013, 84)

Validation shape: (4671, 84)

Test shape: (4671, 84)

4.4.4. Herramientas

Las herramientas utilizadas para el desarrollo del proyecto son las siguientes:

- **Draw.io** [44]: Aplicación web gratuita y de código abierto utilizada para la creación y el diseño de una gran variedad de esquemas, diagramas y figuras de elaboración propia.
- **VS Code** [47]: Editor de código fuente, desarrollado por Microsoft. Ofrece soporte para múltiples lenguajes de programación, y resulta altamente configurable. Su compatibilidad con Python, posibilidad de depuración y control de versiones lo convierten en una herramienta ideal para el desarrollo de scripts y proyectos de análisis de datos. Durante el trabajo, Visual Studio Code fue utilizado como entorno de desarrollo principal para programar los procesos ETL, manipular datos y ejecutar consultas.
- **Python** [49]: Lenguaje de programación de alto nivel ampliamente utilizado en ciencia de datos, análisis estadístico, desarrollo web y automatización de procesos. En el proyecto, Python se emplea para desarrollar los procesos Extracción, Transformación y Carga (ETL), desarrollo de modelos de aprendizaje automático y desarrollo de la arquitectura del bloque del agente de inteligencia artificial, entre otros.
- **SQLite** [43]: Sistema de gestión de bases de datos relacional que se caracteriza por ser ligero y de código abierto. A diferencia de otros sistemas, no requiere un servidor, lo que lo hace ideal para aplicaciones de escritorio, prototipos y pequeños proyectos analíticos. Su integración con Python ha permitido automatizar la creación de tablas y la realización de consultas SQL directamente desde el editor. Se ha utilizado para la *tool* del agente de cargar las tablas y poder realizar consultas sobre ellas.
- **Streamlit** [46]: Librería de Python o *Framework* de código abierto para el desarrollo rápido de aplicaciones web interactivas para análisis de datos y aprendizaje automático. Se ha utilizado para la interfaz gráfica del proyecto.
- **Langchain** [48]: *Framework* de Python orientado al desarrollo de aplicaciones que integran modelos de lenguaje. Facilita la creación de agentes inteligentes y sistemas conversacionales. Se ha empleado para el bloque del agente conversacional para las consultas sobre datos de fútbol.

4.5 Cronograma

Antes de planificar el trabajo del proyecto, es necesario identificar las tareas a realizar y delimitar con precisión el alcance de este trabajo. El alcance del proyecto se descompone hasta el más mínimo detalle por medio de la EDT (Estructura de Desglose de Trabajo) o en inglés, *WBS (Work Breakdown Structure)*, que consiste en una descomposición jerárquica orientada a los entregables del trabajo para cumplir con los objetivos del proyecto que se definieron en

la Sección 1.3 y para crear los entregables requeridos. La realización de la EDT ha permitido organizar y definir el alcance total del proyecto. Además, la EDT se divide por niveles de trabajo, que contienen los componentes de la EDT/WBS, que se llaman paquetes de trabajo. En la Figura 4.5 se muestra la EDT realizada con los diferentes niveles que contienen los respectivos paquetes de trabajo.

- **Nivel 0:** Contiene la tarea general del proyecto, que consiste en la implementación de una plataforma de predicción y de consultas a un agente sobre datos de fútbol. Constituye el primer nivel y es el más general.
- **Nivel 1:** Incluye los principales paquetes de trabajo, como son las tareas de planificación y diseño (paquete de trabajo con código 1 en la Figura 4.5), ETL (Extracción(tarea con código 2.1), transformación(con código 2.2) y carga de datos(con código 2.3)), modelos de aprendizaje automático (3), utilización de un *framework* de agentes (4), desarrollo de una interfaz web (5) y pruebas y validación (6).
- **Nivel 2:** Este nivel contiene las tareas más específicas. A pesar de que se han seleccionado las tareas más importantes, se podría haber considerado un nivel 3 extra que incluyan tareas con un mayor detalle, pues por ejemplo, dentro de la tarea de procesamiento de datos (paquete de trabajo con código 2.2 en la Figura 4.5), se han llevado a cabo tareas como sustitución de valores nulos, creación de nuevas variables y normalización de variables, entre otras. Debido a la complejidad de incorporar tareas aún más pequeñas y específicas, se ha optado por limitar la profundidad de la EDT al Nivel 2. En la Figura 4.5 se muestra la estructura de desglose de trabajo clasificada en los tres grandes niveles:

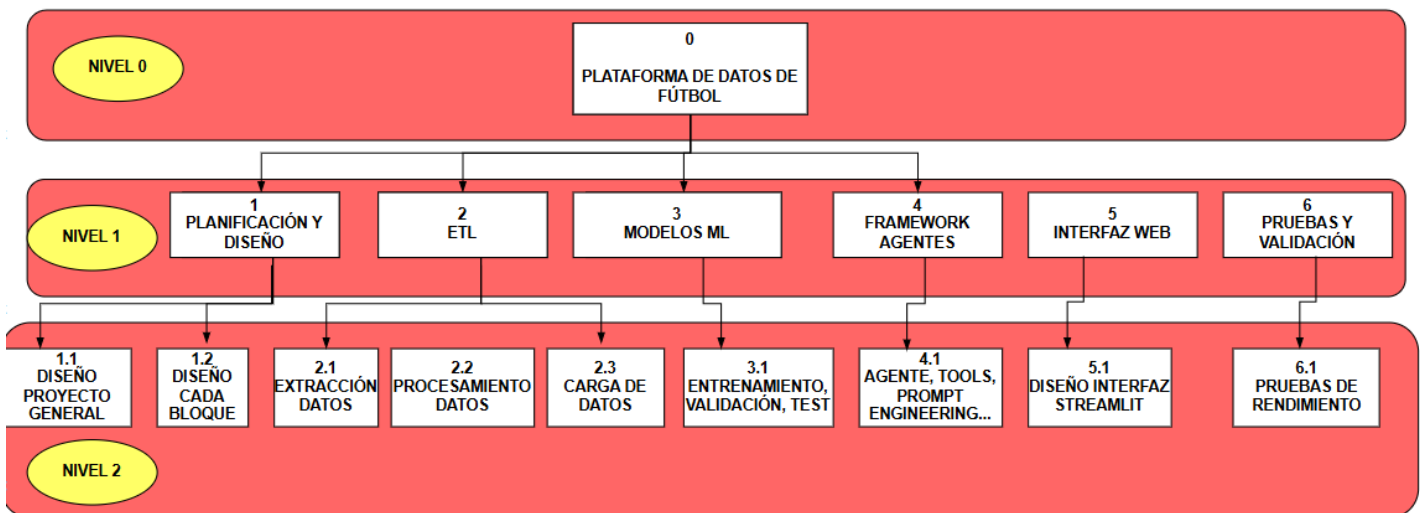


Figura 4.5. Estructura de desglose de trabajo o EDT. Fuente: Elaboración propia realizada con herramienta Draw.io

Por otra parte, también es necesario definir algunos límites del alcance del proyecto. En este caso, debido al coste computacional que requiere el procesamiento y entrenamiento de los modelos con un gran volumen de datos, esta plataforma se ha implementado enfocándose

en dos únicas temporadas de la liga española, concretamente 2022/23 y 2023/24. Otro de los límites de alcance de este proyecto es la arquitectura diseñada con el *framework* de agentes. Debido a la complejidad que supone una arquitectura con un gran número de agentes o herramientas, se ha optado por una arquitectura sencilla y asignando tareas específicas al agente y sus determinadas *tools*. Esto supone un límite de alcance del proyecto porque podría tratarse de una arquitectura más compleja y con más funcionalidades a pesar del gran funcionamiento de la actual plataforma.

Respecto a la planificación que se ha seguido en este proyecto, esta se basa en una metodología ágil. En este caso, se ha optado por una alternativa donde se realizará una combinación entre la **planificación Scrum** y la **planificación en ola**.

4.5.1. Planificación Scrum

El marco de **Scrum** se centra en las fases iterativas y técnicas, donde se necesitan entregas funcionales iterativas, como los paquetes de trabajo de la fase de experimentación, el entrenamiento de modelos de aprendizaje automático, optimización de hiperparámetros y pruebas mediante la evaluación de métricas de los modelos de predicción en el conjunto de *test* (paquete de trabajo 3.1 de la EDT de la Figura 4.5), entre otras. Aunque la planificación se ha enfocado más en una metodología de planificación en ola, en la Figura 4.6 se representa un diagrama de ciclo Scrum sencillo que contiene los elementos esenciales del proceso Scrum de carácter general: *Product Backlog*, *Sprint Backlog*, el ciclo del *sprint*, las reuniones diarias y el *Product Increment*. Además, se debe mencionar que los *sprints* se han asignado con una duración de 2 semanas de duración, manteniendo el *feedback* constante con la tutora del proyecto.

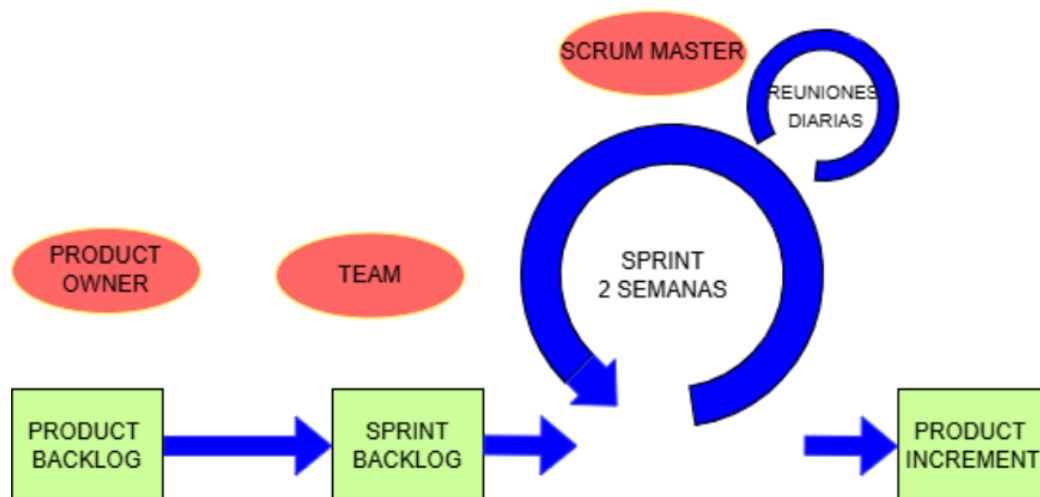


Figura 4.6. Diagrama ciclo Scrum del proyecto. Fuente: Elaboración propia con herramienta Draw.io

Como explicación de este diagrama, que se trata de un diagrama general para un proyecto común, se van a explicar los componentes o figuras que sí se han aplicado en la planificación de este proyecto. El *Product Backlog*, en este caso, es la lista priorizada de requisitos, funcio-

nalidades o mejoras. Por ejemplo, la creación de la interfaz gráfica para el usuario mediante la herramienta *Streamlit*, correspondiente al paquete de trabajo con código 5.1 del Nivel 2 de la EDT de la Figura 4.5, considerada además de requisito un objetivo de la Sección 1.3, podría pertenecer a esta lista. Al tratarse de un diagrama de carácter general, se ha considerado añadir la figura de *Product Owner*, que se define como el responsable de gestionar las prioridades basándose en los objetivos del proyecto y el feedback de los *stakeholders*, en este caso, los usuarios de la plataforma. Respecto al *Sprint Backlog*, el equipo (en el diagrama representado como *Team*) selecciona unas tareas específicas del *Product Backlog* y se incluyen en el *Sprint Backlog*, dependiendo del *sprint* correspondiente. En tercer lugar, se trabajan las tareas asignadas en el *Sprint Backlog*, en este caso, cada sprint corresponde a un entregable del Nivel 2 de la EDT. Además, el *Scrum Master*, perfil para un proyecto general que en este caso no se aplica, supervisa que el equipo siga las prácticas ágiles y resuelve obstáculos. En cuarto lugar, en el diagrama se reflejan las reuniones diarias rápidas que se realizan durante el Sprint, donde el equipo discute los avances, los desafíos y las tareas pendientes. En este caso, para este proyecto solo ha sido necesario reunirse con la tutora para cada *sprint*, pero suele ser una práctica común en los proyectos, de ahí que aparezca en el diagrama general de la Figura 4.6.

4.5.2. Planificación en ola

Por otro lado, la **planificación en ola** se basa en definir el cronograma general y detallar las primeras tareas más inmediatas, como el diseño de la arquitectura del proyecto y el proceso de ETL, con códigos 1 y 2 en la EDT de la Figura 4.5, respectivamente.

En la Figura 7.2 del capítulo 7, correspondiente al Apéndice del trabajo, se muestra el diagrama de Gantt. Este diagrama permite visualizar la secuenciación de tareas y la relación entre ellas, facilitando el manejo de dependencias. Además, refleja de manera clara la transición de tareas detalladas a bloques generales, mostrando cómo la planificación se ajusta conforme avanza el proyecto. Para la realización de este diagrama, se ha requerido de la EDT para desglosar el trabajo realizado en los distintos paquetes de trabajo, que a su vez se dividen en tareas más específicas. Los plazos de fechas establecidos en el diagrama son aproximados y se ha optado por dividir el eje Y de la Figura 7.2 en meses, y estos a su vez en semanas. A través del diagrama se refleja que los paquetes de trabajo 2.2, correspondiente al procesamiento de datos, y el 3.1, correspondiente al entrenamiento de los modelos y la optimización de los hiperparámetros, han resultado los paquetes con mayor duración del proyecto, principalmente debido al coste computacional que requiere procesar y entrenar un volumen de esta cantidad de datos.

4.6 Presupuesto

Esta sección muestra el coste estimado del proyecto con un cálculo de días y horas aproximado. Estos costes podrían aumentar, si por ejemplo, se usaran distintas licencias. En particular, el acceso a fuentes de datos sobre ligas deportivas, específicamente la liga española de fútbol, podría suponer un aumento significativo en el presupuesto debido a que numerosas *APIs* tienen cuotas elevadas para la extracción de sus datos.

Como ya se comentó anteriormente, la API utilizada para la extracción de datos es *API Football* [45] y se eligió el plan 'Pro' que ofrecen para poder realizar la extracción de la manera más rápida y eficiente posible. Esta opción tiene un coste de 19 euros al mes siguiendo los datos de su página web oficial [41].

En cuanto al almacenamiento de datos, como las tablas que se levantan en la *tool* que utiliza el agente de inteligencia artificial se basan en SQLite, esta tarea evitó costes adicionales y fue totalmente gratuita.

Por otro lado, también se debe destacar que durante toda la fase de experimentación en los meses previos a la fecha de entrega de este trabajo, el bloque desarrollado de las consultas al agente sobre datos de la liga española se implementó utilizando una *API_key* gratuita de Google Cloud Platform, concretamente de Google Studio, para evitar costes adicionales al tratarse de la fase de experimentación del proyecto. Para ello, la llamada al *LLM*, en este caso "*gemini-2.0-flash*", se realizó de la siguiente manera:

```
llm = ChatGoogleGenerativeAI(model="gemini-2.0-flash", google_api_key="...")
```

Una vez se terminó esta fase de pruebas, se consiguió obtener la licencia de 3 meses gratuita para un proyecto en la nube de Google Cloud Platform equivalente a 300 euros. Esta licencia permite a los estudiantes desarrollar cualquier proyecto en su plataforma en la nube hasta superar o bien esa cantidad de dinero, o si no se ha llegado a sobrepasar, superar la cantidad de tiempo de 3 meses desde la fecha de inicio del proyecto. En este caso, esta oportunidad ha permitido hacer uso de los *LLM* dentro del límite del presupuesto, enriqueciendo al agente y dotándole de la capacidad de poder generar respuestas. Se probaron diferentes modelos y finalmente, tras un análisis de los precios por token de cada *LLM* y ventajas de cada uno, se optó por "*gemini-2.0-flash*" debido a las respuestas claras, concisas y sin alucinaciones por parte del *LLM*.

Por último, para la redacción de la memoria en *látex* se ha utilizado la plataforma *Overleaf* [50]. Esta herramienta cambió su política de precios durante el desarrollo del proyecto y de ahí que los primeros cinco meses de redacción de memoria resultaran gratuitos. Finalmente, los últimos dos meses se aprovechó el plan de estudiante para reducir los costes de nuevo, que se trataba de un abono de 10 euros al mes. En la Tabla 4.9 se muestran los costes principales de las tareas no gratuitas realizadas en el proyecto.

Tabla 4.9. Coste de tareas realizadas en el proyecto

Tarea	Tiempo	Coste total
Extracción de datos	1 mes (primer mes)	19€
Proyecto Google	1 mes (último mes)	300€
Redacción memoria	5 meses gratis + pago 2 meses pago	10 x 2= 20€
Proyecto total	7 meses	339€*

*Este coste total es aproximado. El coste ha sido mucho más bajo de lo esperado debido a la obtención de licencias como estudiante, tanto para la redacción de la memoria como para la llamada a los *LLMs*, que fue posible gracias a la licencia de estudiante de Google Cloud.

Resumidamente, la Tabla 4.10 expone los costes totales clasificando el proyecto en las dis-

tintos tipos de coste, mostrando que el valor puede ser cuantificado en tiempo o en precio. En esta Tabla 4.10 también se resalta que durante la fase de investigación, la búsqueda del estado del arte se realizó mediante plataformas de investigación gratuitas, libros y el repositorio TITULA de la Universidad Europea. Por el contrario, algunas de las tareas de la fase de experimentación que necesitan de o bien un *software* específico, o algunas herramientas u otras plataformas, sí que elevaron el coste total del proyecto, como se puede ver en la Tabla 4.9 anteriormente mencionada. Sin embargo, no todas las tareas del desarrollo del proyecto requirieron de plataformas o herramientas de pago, pues otras tareas como la elaboración de figuras propias para la redacción de la memoria, realizadas con la herramienta Draw.io, o la implementación de la interfaz con la librería Streamlit, entre otras muchas tareas, resultaron gratuitas.

Tabla 4.10. Costes totales del proyecto

Tipo de coste	Valor (tiempo o precio)	Comentarios
Horas de trabajo en el proyecto	315 horas (valor en tiempo)	Tiempo medio diario: 1 hora y media. Al tratarse de un proyecto de 7 meses aproximadamente, 1.5 horas x 210 días = 315 horas, incluyendo reuniones con tutores, implementación técnica de la plataforma y redacción de memoria.
Equipo técnico utilizado	0€	Ordenador personal.
Software utilizado	339 €	Herramientas o <i>frameworks</i> : Google Cloud Platform (300€), <i>Langchain</i> (0€), API Football (19 €), Overleaf (20 €), librería Streamlit para interfaz (0 €).
Estudios e informes	0€	Búsqueda del estado del arte: Plataformas de investigación, libros, plataforma Google Scholar y repositorio TITULA de la Universidad Europea.

Capítulo 5. Resultados y discusión

En este capítulo se mostrarán los resultados obtenidos del trabajo realizado de ambos bloques principales.

5.1 Resultados y discusión bloque de IA Generativa

En primer lugar, esta sección incluye los resultados, junto con su discusión, del bloque de la consulta de datos de fútbol al agente de inteligencia artificial.

Por un lado, el primer bloque de la consulta inteligente de datos de fútbol ha destacado en varios aspectos importantes. Al no existir métricas deterministas como sí las hay para el bloque de predicciones de los modelos de aprendizaje automático, explicadas en la sección 3.3, como el objetivo es transformar preguntas en lenguaje natural en consultas sobre los datos y devolver respuestas comprensibles y precisas, los resultados se deben orientar a evidencias cualitativas y ventajas funcionales, valorando los resultados en eficiencia, accesibilidad y usabilidad. Un ejemplo ilustrativo es la consulta realizada sobre la valoración media del rendimiento de Vinícius Júnior en los partidos de local. La *query* o consulta del usuario es la siguiente:

*'Calcula la valoración media de 'rating' que tiene el jugador Vinícius Júnior
en los partidos jugando de local en casa.'*

Mediante el parámetro *verbose=True*, esto permite ver lo que realiza el agente por detrás y así comprobar el funcionamiento y razonamiento interno, incluyendo las llamadas a las herramientas o *tools* que disponía el agente, como ya se explicó en la Figura 4.3.

Ejemplo 1: Consulta sobre rendimiento medio de un jugador.

Paso 1: Lectura de columnas del CSV.

En este primer paso, el agente utiliza la herramienta 'List CSV Columns' para comprobar las columnas que dispone en este *dataset*.

Action: List CSV Columns

Action Input: data_jugadores_y_partidos_limpio.csv

Observation: ['fixture_id', 'player_id', 'player_name', 'minutes',
'starter', 'rating', 'captain', 'goals', 'assists', 'yellow_cards',
'red_cards', 'offsides', 'shots_total', 'shots_on_target',
'duels_total', 'duels_won', 'dribbles_attempts', 'dribbles_success',
'fouls_drawn', 'fouls_committed', 'passes_key', 'passes_accuracy',
'passes_total', 'tackles_total', 'tackles_blocks', 'interceptions',
'penalty_scored', 'penalty_missed', 'penalty_saved', 'idEvent',
'dateEvent', 'homeScore', 'awayScore', 'venue', 'status', 'is_home',
'rival_mean_rating', 'rival_avg_rating', 'goal_difference',
'player_name_encoded', 'team_alaves', 'team_almeria',

```
'team_athletic club', 'team_atletico madrid', 'team_barcelona',  
'team_cadiz', 'team_celta vigo', 'team_getafe', 'team_girona',  
'team_granada cf', 'team_las palmas', 'team_mallorca', 'team_osasuna',  
'team_rayo vallecano', 'team_real betis', 'team_real madrid',  
'team_real sociedad', 'team_sevilla', 'team_valencia', 'team_villarreal',  
'rival_team_alaves', 'rival_team_almeria', 'rival_team_athletic club',  
'rival_team_atletico madrid', 'rival_team_barcelona', 'rival_team_cadiz',  
'rival_team_celta vigo', 'rival_team_getafe', 'rival_team_girona',  
'rival_team_granada cf', 'rival_team_las palmas', 'rival_team_mallorca',  
'rival_team_osasuna', 'rival_team_rayo vallecano', 'rival_team_real betis',  
'rival_team_real madrid', 'rival_team_real sociedad', 'rival_team_sevilla',  
'rival_team_valencia', 'rival_team_villarreal', 'position_D', 'position_F',  
'position_G', 'position_M', 'month', 'day_of_week']
```

Paso 2: Creación de la tabla SQL

En segundo lugar, una vez el agente ya conoce las columnas que dispone para analizar cuáles pueden aportar más información para la generación de la respuesta a la petición lanzada por el usuario, el agente continúa el flujo utilizando la herramienta 'Setup SQL Tables'. Esto le permitirá crear la tabla SQL a la que posteriormente lanzará consultas.

Action: Setup SQL Tables

Action Input: data_jugadores_y_partidos_limpio.csv

Observation: Successful tables set up: ['data_jugadores_y_partidos_limpio']

Paso 3: Consulta SQL

En tercer lugar, la petición en lenguaje natural que ha recibido de la consulta del usuario la transforma en lenguaje SQL y así poder realizar la *query* a la tabla creada. A continuación se muestra la consulta, donde el agente decide satisfactoriamente que la columna más interesante es 'player_name' para poder acceder al nombre del jugador Vinícius Júnior y la columna 'is_home' con valor 1, donde ya se explicó su significado en las variables *dummies* de la Sección 4.4.1, correspondientes a los partidos jugados como local. Esto le permitirá conocer la información de los partidos jugados como local del jugador. Además, cabe destacar que gracias a la utilización del filtro *WHERE*, esto le permite realizar un filtro a todo el resto del *dataset* y enfocarse únicamente en la información correspondiente al jugador Vinícius Júnior en los partidos únicamente jugados en casa. Este resultado supone una gran ventaja porque agiliza la búsqueda para encontrar la información disponible y así posteriormente calcular lo que necesita con esos datos ya filtrados. Además, el agente ha entendido exitosamente que necesita calcular la media del rendimiento, correspondiente a la variable 'rating' de ese filtro realizado con el método *WHERE*. Para ello, utiliza el operador *AVG* sobre esa variable de rendimiento.

Action: Query SQL

```
Action Input:
SELECT AVG(rating)
FROM data_jugadores_y_partidos_limpio
WHERE player_name = 'Vinícius Júnior'
AND is_home = 1;
Observation: {'AVG(rating)': {0: 7.699999999999999}}
```

Generación de la respuesta

Por último, el resultado final tras la consulta SQL realizada a la tabla es de 7.6999. El agente muestra por pantalla la solución en lenguaje natural para que lo entienda el usuario. De esta manera, se cumple con los objetivos anteriormente mencionados en la Sección 1.3, destacando la interacción con el usuario y su fácil interpretabilidad, permitiendo que los usuarios no necesiten niveles técnicos de programación al recibir la respuesta en lenguaje natural, de ahí la gran ventaja de esta plataforma. A continuación se muestra la respuesta final del agente:

La valoración media de Vinícius Júnior en partidos de local es 7.7.

Resumidamente, el agente ha sido capaz de resolver las siguientes tareas gracias a las *tools* proporcionadas:

- **Identificar correctamente el archivo correspondiente:** En este ejemplo, el agente dispone de una serie de archivos csv en una carpeta que se le indica en el *prompt* y es capaz de seleccionar el archivo que necesita para responder a la pregunta.
- **Explorar las columnas relevantes del CSV:** El agente lista las columnas existentes en el CSV seleccionado y analiza cuáles de ellas son mas probables a contener la información que se busca.
- **Crear la tabla SQL en memoria:** El agente crea la tabla SQL para poder realizar consultas sobre ella.
- **Formular la consulta SQL necesaria:** El agente entiende y procesa lo que busca el usuario en su pregunta y transforma a lenguaje SQL para poder extraer los cálculos de la tabla creada.
- **Devolver el resultado final en lenguaje natural:** El agente, una vez calculado el resultado, genera la respuesta al usuario con la información necesaria ya recopilada. En este caso, la respuesta es: 'La valoración media de Vinícius Júnior en partidos de local es 7.7'.

Por tanto, tras la finalización del trabajo del agente para dar respuesta a esta consulta del usuario mediante este ejemplo ilustrativo, se pueden concluir las siguientes ventajas:

- **Eficiencia en la consulta de datos:** El agente automatiza pasos que un usuario debería realizar manualmente, como cargar el archivo, inspeccionar columnas, generar la consulta SQL, ejecutar y devolver el resultado.
- **Rapidez de respuesta:** La transformación desde la pregunta en lenguaje natural hasta la respuesta numérica se realiza en cuestión de segundos.

- **Organización en el flujo de trabajo:** Mediante técnicas como el *prompt engineering*, el agente sigue una secuencia lógica (listar, explorar, preparar tabla, consultar) que asegura consistencia y minimiza errores.
- **Accesibilidad para usuarios no técnicos:** Se elimina la necesidad de conocimientos en SQL o programación, pues basta con formular preguntas en lenguaje natural.
- **Flexibilidad del sistema:** Aunque en este caso se usan datos de La Liga española, la arquitectura puede reutilizarse en cualquier dominio simplemente cambiando los datos de entrada y ajustando el prompt.
- **Mejora en la experiencia de usuario:** Al mantener un historial conversacional, el agente permite encadenar consultas progresivas, enriqueciendo el análisis interactivo de los datos.

Para finalizar la discusión y resultados del primer bloque del trabajo sobre la consulta inteligente de los datos de jugadores de fútbol, a continuación se muestra otro ejemplo distinto pero también necesario para comprender la capacidad del agente, con sus herramientas, de dar respuesta a todo lo que el usuario necesite preguntar. En este caso, la respuesta no será un valor numérico como la media de goles, asistencias e incluso regates, sino un conjunto de nombres de jugadores que deberá descubrir y que cumplan una serie de condiciones. La pregunta que el usuario realiza consiste en averiguar los jugadores de la temporada 2023-2024 que, jugando como defensas, sean los máximos goleadores entre su posición. Para ello, se debe tener en cuenta que los jugadores defensivos carecen de estadísticas goleadoras muy altas. De hecho, el agente deberá analizar todos los goleadores de todos los partidos, clasificar a aquellos que sean defensas, agruparlos y calcular los goles que lleve cada uno, y por último, calcular quiénes de ellos son los que más goles anotaron. La consulta que realiza el usuario es la siguiente:

'Dime los jugadores defensas que marcaron más goles en la temporada 2023-2024'

Ejemplo 2: Consulta sobre defensas goleadores

Paso 1: Razonamiento de la consulta del usuario:

Obtener los jugadores defensas que marcaron más goles en la temporada, mostrando el top 10 ordenado de mayor a menor número de goles.

Razonamiento inicial (Thought):

El agente identifica que debe acceder al archivo CSV, consultar sus columnas, crear la tabla SQL correspondiente y, posteriormente, ejecutar una consulta que filtre a los jugadores cuya posición sea defensa (`position_D = 1`) y calcule el número total de goles.

Paso 2: Lectura de columnas del CSV:

El agente utiliza la herramienta 'List CSV Columns' para comprobar las columnas que dispone en este dataset. El agente identifica que las columnas del nombre del jugador, goles y

posición de defensa (variable *dummy* son las que aportarán información a la respuesta.

Acción: Listar columnas del CSV

Input: data_jugadores_y_partidos_limpio.csv

Observación: Se obtiene la lista completa de columnas, entre las que destaca player_name, goals y position_D.

Paso 3: Creación de la tabla SQL:

El agente continúa el flujo utilizando la herramienta 'Setup SQL Tables'. Esto le permitirá crear la tabla SQL a la que posteriormente lanzará consultas.

Acción: Crear tabla SQL a partir del CSV

Input: data_jugadores_y_partidos_limpio.csv

Observación: Tabla creada correctamente: data_jugadores_y_partidos_limpio.

Paso 4: Consulta SQL:

El agente convierte la petición del usuario en lenguaje natural a una consulta SQL. Para ello, en primer lugar utiliza el filtro *WHERE* para seleccionar a los jugadores que actúan como defensas. Posteriormente, utiliza el operador *GROUP BY* para agrupar a los jugadores. Además, utiliza *ORDER BY* para ordenar a los futbolistas de mayor a menor en función del número de goles anotados en la temporada y *LIMIT* para limitar la búsqueda a los 10 máximos goleadores. Por último, selecciona el nombre del jugador y la suma de goles totales anotados, gracias al operador *SUM()*. A continuación se muestra la consulta SQL:

Acción: Ejecutar consulta SQL

Input:

```
SELECT player_name, SUM(goals) AS total_goals
FROM data_jugadores_y_partidos_limpio
WHERE position_D = 1
GROUP BY player_name
ORDER BY total_goals DESC
LIMIT 10;
```

Observación: Se obtiene un listado de defensas con sus respectivos goles.

Paso 5: Generación de la respuesta:

Por último, el agente muestra por pantalla la respuesta en lenguaje natural con la información recopilada. La respuesta obtenida es la siguiente:

'Los defensas que marcaron más goles son: Eric García (5 goles), Daniel Carvajal (4 goles), Yuri Berchiche (3 goles), Sergio Ramos (3 goles), Ricard Sánchez (3 goles), Lucas Vázquez

(3 goles), Kike Salas (3 goles), Jorge Cuenca (3 goles), Florian Lejeune (3 goles), Antonio Raíllo (3 goles).'

Finalmente, se vuelve a concluir que gracias a una respuesta clara y concisa en lenguaje natural, esta plataforma permite una gran accesibilidad para usuarios no técnicos sin necesidad de tener conocimientos en bases de datos o programación, pues mediante una petición en lenguaje natural, el agente tiene la capacidad de resolver cálculos complejos internamente.

5.2 Resultados y discusión del bloque de predicciones.

Por otra parte, el segundo bloque sí es posible evaluarse en base a las métricas explicadas en la Sección 3.3. Los modelos de aprendizaje automático, explicados en la Sección 3.2, han sido utilizados para predecir el rendimiento de los jugadores y sus titularidades en base a todas las características y variables explicativas que influyen en él. Por tanto, la tarea de predecir el rendimiento de los jugadores se trata de un problema de regresión, mientras que la predicción de si un jugador es titular o no, de clasificación binario. Para la tarea de predicción del rendimiento, los modelos que se han utilizado son: redes neuronales, concretamente *MultiLayer Perceptron Regressor* o *MLPRegressor*, explicado en la Sección 3.2.1, el algoritmo de *Random Forest Regressor*, explicado en la Sección 3.2.2 y el algoritmo XGBoost. Estos han sido los principales modelos utilizados para obtener las predicciones del rendimiento de los jugadores. Por otra parte, para la tarea de predicción de titularidad de un jugador en un partido, se han utilizado modelos de clasificación como: redes neuronales, concretamente *MultiLayer Perceptron Classifier* o *MLPClassifier* y Regresión Logística, explicado en la Sección 3.2.3.

Estas métricas de la Sección 3.3 permiten evaluar a los modelos y decidir cuáles devuelven resultados más fiables y mejores con respecto a los otros.

Respecto a la predicción del rendimiento, al tratarse de un problema de regresión, donde la variable objetivo '*rating*' es una variable numérica de escala de 1 a 10, las principales métricas usadas son *MSE* o error cuadrático medio y R^2 .

Respecto a la predicción de la titularidad, al tratarse de un problema de clasificación binario, con la clase positiva indicando que el jugador será titular en el partido y clase negativa su-plente, se han utilizado las métricas de la matriz de confusión, precisión, *recall*, exactitud y F-1 Score, entre otras.

5.2.1. Resultados predicción titularidad

Antes de obtener los valores de las métricas, es necesario definir los valores de los hiperpa-rámetros óptimos con los que han entrenado los modelos. Por ello, en primer lugar se definen en la Tabla 5.1 todos los posibles valores con los que se ha realizado esta búsqueda de hiper-parámetros óptimos, destacando que el coste computacional ha impedido establecer valores demasiado altos debido a la gran cantidad de datos.

Notación*: Para la representación del número de neuronas y de capas ocultas, (50,) repre-senta 1 sola capa oculta con 50 neuronas, (100,50) 2 capas ocultas con 100 neuronas en la primera capa oculta y 50 neuronas en la segunda capa oculta, y así sucesivamente. En re-

Tabla 5.1. Valores de hiperparámetros posibles tras búsqueda en cuadrícula en modelos de clasificación

MODELO	HIPERPARÁMETROS	VALORES
REGRESIÓN LOGÍSTICA	C	0.01, 0.1, 1, 10, 100
RANDOM FOREST CLASSIFIER	Nº de árboles	100,200,300
	Profundidad máxima	10,20
	Min. nº de muestras	2,5,10
REDES NEURONALES	Nº de neuronas	(50,),(100,),(100,50),(100,100)
	α	0.0001,0.001
	η	0.001,0.01
	Función de activación	'relu','tanh'

des neuronales α representa el parámetro de regularización *ridge* y η la tasa de aprendizaje inicial.

Respecto a la predicción de la titularidad de los jugadores en un partido, en la Tabla 5.2 se muestran los hiperparámetros de los modelos utilizados con los valores óptimos obtenidos. Además, se mostrarán los resultados de los valores de las métricas obtenidas en posteriores tablas, tanto para el conjunto de validación como el conjunto de test.

Tabla 5.2. Hiperparámetros óptimos tras búsqueda en cuadrícula en modelos de clasificación

MODELO	HIPERPARÁMETROS	VALOR
REGRESIÓN LOGÍSTICA	C	100
RANDOM FOREST CLASSIFIER	Nº de árboles	100
	Profundidad máxima	20
	Min. nº de muestras	10
REDES NEURONALES	Nº de capas ocultas	2
	Nº de neuronas	(100,100)
	α	0.001
	η	0.001
	Función de activación	relu

En la técnica de regresión logística, el valor óptimo obtenido para el parámetro C es de 100, siendo, $C = \frac{1}{\alpha}$ y α el parámetro de regularización.

En la técnica de bosques aleatorios para clasificación, concretamente el modelo utilizado *Random Forest Classifier*, el número óptimo de árboles óptimo es 100, la profundidad máxima de 20 y el mínimo número de muestras 10. Debido al coste computacional que requiere trabajar con valores más altos, estos son los que han dado mejores valores en las métricas.

Por último, el número óptimo de capas ocultas en el modelo de redes neuronales, concretamente *MLPClassifier*, es de 2, siendo 100 neuronas en la primera capa y 100 en la segunda. Otros hiperparámetros que se utilizaron son la tasa de aprendizaje inicial, con valor óptimo de 0.001 y el parámetro de regularización, con valor óptimo de 0.001.

Tras la revisión de los hiperparámetros óptimos de los modelos, los resultados obtenidos de las métricas con estos modelos en el conjunto de validación se muestran en la Tabla 5.3, mientras que los resultados obtenidos de las métricas en el conjunto de test se muestran en

la Tabla 5.4.

Tabla 5.3. Resultados en conjunto de validación de cada técnica con sus métricas

TÉCNICAS	SENSIBILIDAD	AUC	EXACTITUD	ESPECIFICIDAD
REGRESIÓN LOGÍSTICA	0.56	0.667	0.590	0.68
RANDOM FOREST	0.87	0.74	0.74	0.41
REDES NEURONALES	0.88	0.783	0.774	0.50

Tabla 5.4. Resultados en conjunto de test de cada técnica con sus métricas

TÉCNICAS	SENSIBILIDAD	AUC	EXACTITUD	ESPECIFICIDAD
REGRESIÓN LOGÍSTICA	0.53	0.666	0.582	0.71
RANDOM FOREST	0.87	0.739	0.733	0.39
REDES NEURONALES	0.87	0.777	0.760	0.50

Como conclusión, la Tabla 5.3 muestra que en el conjunto de validación, se observa que la regresión logística obtiene la menor exactitud (0.590) y un AUC moderado (0.667). Aunque su especificidad es relativamente elevada (0.68), su sensibilidad (0.56) refleja una baja capacidad para identificar correctamente los jugadores titulares, correspondientes a la clase positiva.

El modelo de *random forest* muestra una mejora significativa en sensibilidad (0.87) y en exactitud (0.74), alcanzando además un AUC de 0.74. Sin embargo, su especificidad es baja (0.41), lo que indica que clasifica incorrectamente una gran cantidad de jugadores como suplentes como titulares. La Figura 7.1 del capítulo 7, correspondiente al Apéndice del trabajo, se trata de un gráfico de barras que muestra la importancia de las diez variables que más influyen en la predicción de la titularidad de un jugador. Analizando el gráfico, tiene sentido que la primera variable más importante sea la identificación asignada a cada jugador, seguido de la media del rendimiento en los últimos cinco partidos hasta la fecha del partido a predecir y la media del equipo en los últimos cinco partidos. Como curiosidad, entre estas diez variables más influyentes en la titularidad, destaca que la posición de portero influye notablemente. Esto tiene sentido porque una práctica común de los equipos suele ser asignar a un único portero titular para que juegue toda la temporada como titular. El gráfico de la Figura 7.1, por tanto, indica que si un jugador se trata de un portero titular de algún equipo, tiene más probabilidad de ser clasificado como titular en los modelos de predicción.

Las redes neuronales (MLP) logran el mejor rendimiento global en el conjunto de validación: una exactitud de 0.774, sensibilidad de 0.88 y un AUC de 0.783, superando tanto a la regresión logística como al *random forest*. Su especificidad (0.50) es superior a la del *random forest*, aunque sigue mostrando cierta dificultad para identificar los jugadores suplentes. Esto se debe al desbalanceo que existe entre los jugadores titulares y los jugadores suplentes en el *dataset*, indicando la dificultad en captar el patrón para la predicción de un jugador como suplente.

Respecto al análisis de resultados en el conjunto de *test*, la Tabla 5.4 muestra que en el conjunto de *test*, los valores de las métricas en general son un poco inferiores a los valores de las métricas en el conjunto de validación. Esto indica que los modelos están generalizando

bien y sin sobreajustar. En este caso, la regresión logística mantiene un rendimiento inferior, con exactitud de 0.582 y AUC de 0.666, destacando únicamente por una especificidad relativamente alta (0.71). El modelo de *random forest* conserva una alta sensibilidad (0.87) y un AUC de 0.739, pero vuelve a mostrar baja especificidad (0.39). Por su parte, las redes neuronales vuelven a ser la técnica con mejores resultados obtenidos, pues destacan la exactitud de 0.760 y el mayor AUC (0.777), junto con una sensibilidad de 0.87 y especificidad de 0.50, lo que evidencia un comportamiento más equilibrado que el *random forest*.

Como conclusión, el modelo de redes neuronales (MLP) se consolida como la mejor alternativa, pues alcanza los valores más altos de exactitud y AUC tanto en validación como en test, junto con una sensibilidad muy elevada y una especificidad más equilibrada que la del Random Forest.

5.2.2. Resultados predicción rendimiento

Respecto a la predicción del rendimiento de jugadores en un partido, en primer lugar se muestran los valores de hiperparámetros posibles en la Tabla 5.5 antes de obtener los hiperparámetros óptimos de los modelos.

Tabla 5.5. Espacio de búsqueda de hiperparámetros considerado en la optimización de cada modelo

Modelo	Hiperparámetro	Valores considerados
RANDOM FOREST	Nº estimadores	[200, 300, 400]
	Profundidad máxima	[5, 8]
	Min. muestras por nodo	[2, 5]
	Min. muestras por hoja	[1, 3]
XGBOOST	Nº estimadores	[300, 400, 500]
	Tasa de aprendizaje	[0.01, 0.05, 0.1]
	Profundidad máxima	[3, 5, 7]
REDES NEURONALES (MLP)	Nº de neuronas	[(50,), (100,), (100, 50)]
	α	[0.0001, 0.001, 0.01]
	η	[0.001, 0.01]
	Función de activación	[relu, tanh]

Posteriormente, en la Tabla 5.6 se muestran los valores óptimos de los hiperparámetros de cada técnica. Además, en esta sección se van a presentar los resultados de los modelos de predicción del rendimiento de los jugadores con los hiperparámetros óptimos ya seleccionados tanto en el conjunto de validación como en el de test para comprobar el rendimiento del modelo con datos nuevos y no vistos previamente.

En la técnica de XGBoost, donde se utiliza *XGBoost Regressor*, el número de estimadores óptimo es de 300, mientras que la tasa de aprendizaje un valor de 0.05 y una profundidad máxima de 3.

En la técnica de bosques aleatorios, concretamente *Random Forest Regressor* el número de árboles óptimo, también llamado en inglés *n_estimators* es de 200. Aunque se ha intentado probar con valores más altos, no se pudo comprobar debido al gran coste computacional que

Tabla 5.6. Hiperparámetros óptimos tras búsqueda en cuadrícula en modelos de regresión

MODELO	HIPERPARÁMETROS	VALOR
RANDOM FOREST	Nº de árboles	200
	Profundidad máxima	5
	Min. nº de muestras	5
XGBOOST	Nº estimadores	300
	Tasa de aprendizaje	0.05
	Profundidad máxima	3
REDES NEURONALES	Nº de capas ocultas	1
	Nº de neuronas	(50,0)
	α	0.0001
	η	0.01
	Función de activación	relu

requería. También ocurre con la profundidad máxima, pues aunque el valor óptimo es de 5 niveles, tampoco se pudo comparar con más de 10 por el excesivo coste computacional, al igual que con la tasa de aprendizaje, que se obtiene un valor óptimo de 0.05.

Por último, el número de capas ocultas en redes neuronales, concretamente *MLP Regresor*, seleccionadas tras la búsqueda en cuadrícula es de 1, con 50 neuronas en la primera capa oculta. También se han tenido en cuenta otros hiperparámetros como la tasa de aprendizaje inicial y el parámetro de regularización. Los valores óptimos son de 0.001 y 0.0001, respectivamente.

Por último, tras obtener los hiperparámetros óptimos de los modelos, la Tabla 5.7 muestra directamente los resultados de las métricas en el conjunto de *test*, pues al tratarse de valores muy similares a los obtenidos en el conjunto de validación, se ha optado por mostrar directamente los resultados en el conjunto de datos no vistos previamente, correspondientes al conjunto de *test*.

Tabla 5.7. Resultados de los modelos de regresión en el conjunto de test

TÉCNICAS	MSE	R ²
Random Forest	0.2256	0.2450
XGBoost	0.2236	0.2518
Redes Neuronales (MLP)	0.2843	0.0480

Tras el análisis de los resultados, aunque en la predicción del rendimiento de los jugadores se han obtenido peores métricas que en la predicción de si un jugador será titular o no en un partido, se tratan de valores aceptables teniendo en cuenta el coste computacional que requiere para que los modelos capten mejor los patrones en la predicción del rendimiento. Si bien el valor de R² obtenido (0.25) refleja una capacidad explicativa limitada, es importante destacar que este rendimiento es superior al de un modelo que se limitara a predecir la media de la variable objetivo, de ahí que los resultados sean aceptables a pesar de poder mejorarse con una gran cantidad de datos mayor, pues se ha intentado realizar todo tipo de técnicas con las variables, conjunto de datos e incluso la solución a la fuga de datos y estas han sido las mejores métricas obtenidas teniendo en cuenta las limitaciones.

5.3 Interfaz gráfica: Resultado final

La interfaz gráfica resultante presenta múltiples funcionalidades. En primer lugar, esta permite al usuario tanto realizar predicciones de o bien titularidades o bien rendimiento de jugadores como de realizar consultas al agente conversacional. La Figura 5.1 muestra la portada principal de la interfaz.

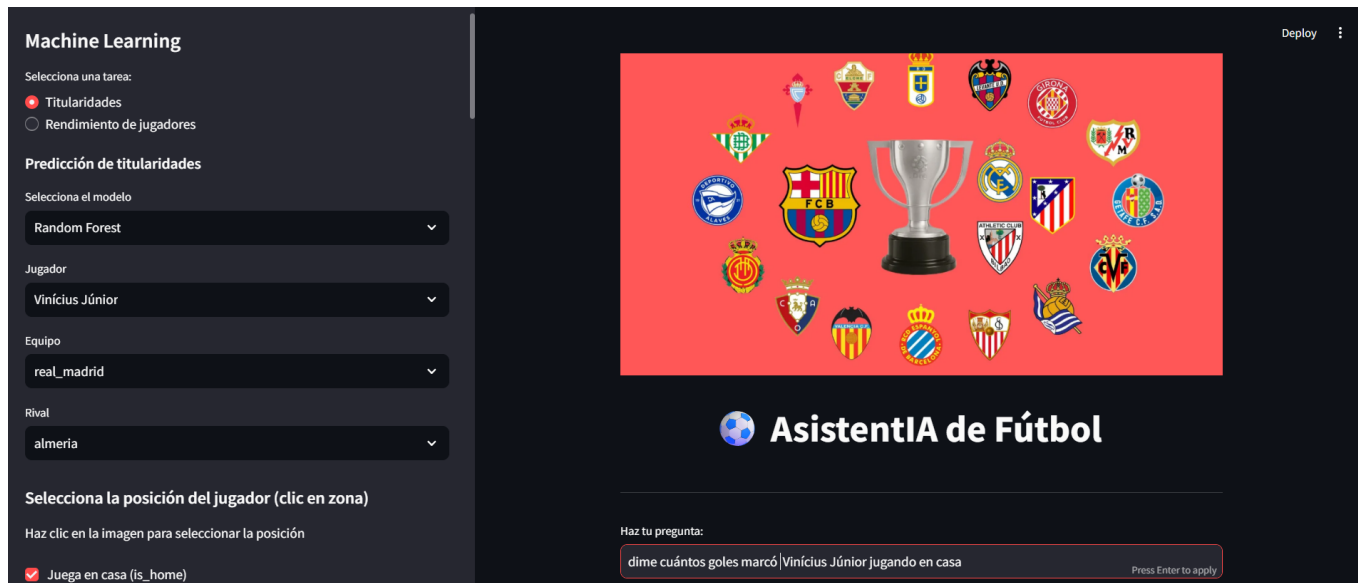


Figura 5.1. Interfaz portada principal

En ella, se pueden observar tanto el bloque del agente, situado a la derecha, como el bloque de predicciones, situado a la izquierda. En este segundo bloque, el usuario puede elegir la predicción del rendimiento o de la titularidad mediante el botón de arriba. Una vez seleccionado, se debe elegir entre el modelo de predicción que utilizar y posteriormente todo el resto de variables. Entre ellas, destaca la variable de posición, la cual el usuario elegirá clicando en la zona de juego de una imagen de un campo de fútbol, produciendo así la capacidad de captación del interés del usuario y siendo mucho más interactiva. Algunas del resto de variables, principalmente las variables categóricas, se seleccionan o bien por un desplegable, como el nombre de todos los jugadores, el equipo al que pertenecen y rival de los equipos. Otras de ellas, principalmente las variables numéricas, se seleccionan con una barra interactiva con sus respectivos rangos por variable. De esta manera, se impulsa la participación del usuario y se facilita que no tenga que poner a mano los valores numéricos, sino directamente utilizando la barra. Todas estas funcionalidades se muestran gráficamente en la Figura 5.2.

Finalmente, mediante los botones de 'Enviar' y 'Predecir', se realizan los diferentes procesos, dependiendo del bloque. Un ejemplo de conversación con el agente donde el usuario realiza una consulta al agente es la mostrada en la Figura 5.3.

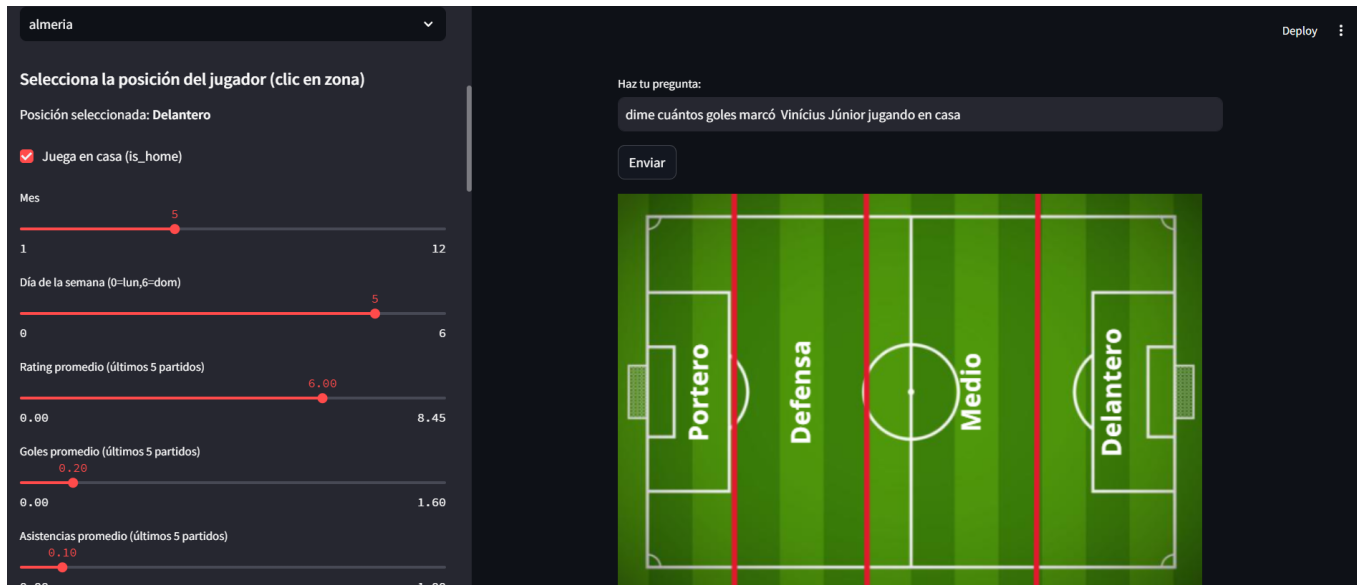


Figura 5.2. Funcionalidades de las variables en la interfaz

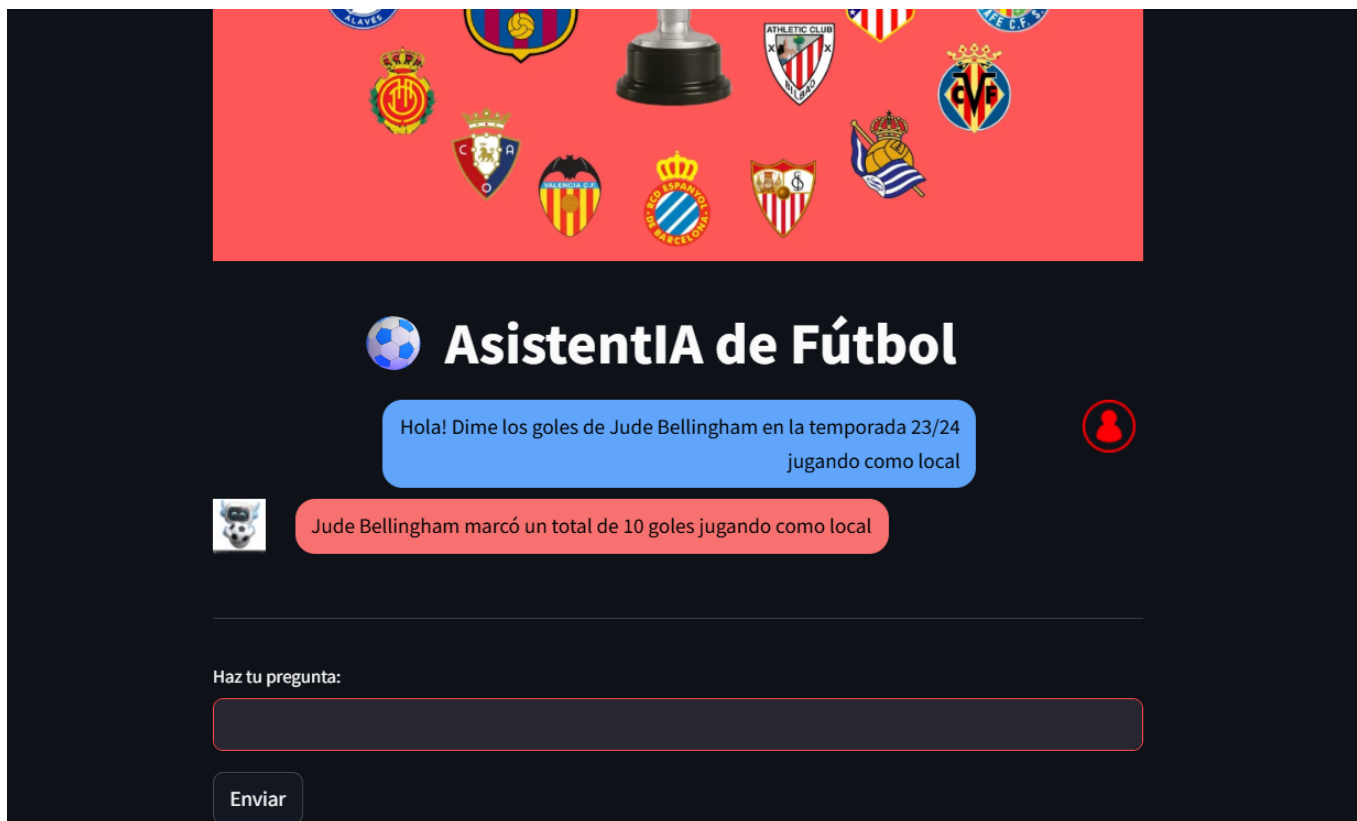


Figura 5.3. Consulta al agente conversacional

Capítulo 6. Conclusiones y trabajo futuro

En este último capítulo se toman en cuenta todos los objetivos iniciales propuestos y las funcionalidades alcanzadas de la plataforma para realizar una conclusión y análisis final del trabajo realizado. Además, se añade una sección de trabajo futuro como mejoras a la plataforma.

6.1 Conclusiones

Más allá de la utilización de las distintas herramientas, plataformas e incluso *frameworks* de agentes de inteligencia artificial, el uso de los lenguajes de programación Python y SQL ha sido esencial y el pilar fundamental para la construcción de este proyecto. Mediante el uso del lenguaje Python, ha sido posible completar el proceso *ETL* o *Extract, Transform and Load*, incluyendo la extracción de datos vía API, la transformación o procesamiento de ellos y la carga en la base de datos. Además, todo el bloque de los modelos de predicción de aprendizaje automático ha sido realizado mediante este lenguaje. Por otra parte, el lenguaje SQL ha resultado de gran importancia para la creación de las consultas del agente a las tablas de SQLite, haciendo uso de la herramienta implementada para transformar la petición del usuario en lenguaje natural a lenguaje SQL.

A continuación se realizará una revisión de las tareas y funcionalidades conseguidas en la plataforma para valorar el trabajo realizado.

- Se ha logrado obtener un gran conjunto de datos mediante conexión a una API, concretamente *API Football* [45], que incluye *datasets* de los partidos de los equipos, estadísticas de los jugadores en cada partido y de diferentes temporadas, enfocándose principalmente en las temporadas 2022-2023 y 2023-2024.
- Se han obtenido unos datos limpios tras la aplicación de varias técnicas de procesamiento de datos.
- Se han obtenido los mejores hiperparámetros posibles teniendo en cuenta las limitaciones del gran volumen de datos y consecuentemente métricas con resultados aceptables de los modelos de aprendizaje automático.
- Se ha implementado un agente de inteligencia artificial experto en los datos proporcionados capaz de responder a una gran velocidad con respuestas sencillas, claras y efectivas.
- Se ha diseñado una interfaz web interactiva y accesible para generar audiencia y facilitar a los usuarios el uso de la plataforma.

En balance, se ha conseguido cumplir con los objetivos propuestos inicialmente y obtener una plataforma útil para usuarios como equipos de scouting, analistas de datos e incluso aficionados de la liga española de fútbol. Sin embargo, lo realmente interesante es la flexibilidad de esta plataforma, principalmente el bloque del agente conversacional experto en los datos proporcionados, pues más allá de tratarse de datos de fútbol, se ha conseguido una plataforma que puede reutilizarse en cualquier ámbito simplemente sustituyendo los datos actuales por cualquier otro *dataset* de entrada y modificando algunas instrucciones específicas al agente de inteligencia artificial. En definitiva, la posibilidad de reutilizar este proyecto en diferentes

necesidades lo dota de un alto valor añadido y abre la puerta a nuevas aplicaciones en el futuro.

6.2 Trabajo futuro

La principal limitación de este proyecto ha sido el gran volumen de carga de datos. Esto ha implicado un elevado coste computacional y una mayor dificultad para conseguir valores altos en las métricas de los modelos de aprendizaje automático. Debido a esto, se adoptó una estrategia para realizar la división de los datos en subconjuntos de entrenamiento, validación y *test*. Este conjunto de observaciones de validación ha permitido el estudio de todas las combinaciones de hiperparámetros para conseguir maximizar el rendimiento de los modelos. Una vez realizado todo este proceso, la utilización de los distintos algoritmos ha sido esencial para poder comparar el comportamiento de cada uno. Por tanto, un aspecto de mejora en futuros trabajos que mejoraría los valores de las métricas de los modelos de aprendizaje automático es la incorporación de datos adicionales que podrían aumentar aún más el rendimiento de los modelos, pues el gran coste computacional que requiere el análisis de un gran volumen de observaciones y variables ha sido un gran desafío. En este caso de uso, la incorporación de datos de partidos y jugadores de más temporadas permitiría poder capturar mejor los patrones existentes y mejorar las predicciones.

Por otra parte, un aspecto a tener en cuenta en fases futuras es el diseño de la arquitectura implementada para el bloque del agente de inteligencia artificial. Hoy en día, con la tecnología evolucionando constantemente, surgen nuevas soluciones y diseños de arquitectura que optimizan el rendimiento, velocidad de generación de respuesta y otros muchos factores. Como ya se explicó en la Figura 2.1, se valoró previamente la arquitectura de un RAG para el desarrollo de este trabajo. Sin embargo, al obtener resultados rápidos, eficientes y correctos, se optó por la arquitectura actual diseñada. Aún así, a pesar de que el elevado coste computacional para el bloque de las predicciones ha impedido la incorporación de datos de más temporadas, el bloque de las consultas al agente experto en los datos proporcionados sí podría adoptar una gran mejora en cuanto a escalabilidad, pues la incorporación de más *datasets* para que el agente explore e identifique información y cálculos de estadísticas de otros años sería posible de realizar mediante la implementación de una arquitectura RAG que optimizaría aún más la plataforma, donde se tendría que vectorizar toda la información con modelos de *embeddings* y hacer uso de distancias matemáticas entre los nodos para la recuperación de información. Por último, es importante resaltar que la incorporación de esta arquitectura RAG no sería la sustitución a la arquitectura actual diseñada, pues la utilización de las herramientas implementadas que permite al agente realizar cálculos complejos con las estadísticas de los jugadores no sería posible sin hacer uso de estas *tools* desarrolladas.

Una última consideración futura a tener en cuenta es el *framework* de agentes utilizado. Como ya se explicó anteriormente, una de las grandes ventajas de este proyecto es la flexibilidad de la plataforma para adaptarse a cualquier otro ámbito mediante la sustitución de los *datasets* ingestados. Si en un futuro se necesita hacer uso de esta plataforma para consultar cuestiones sobre otros temas, la incorporación de otras *tools* para capacitar con otras características al agente e incluso aumentar el número de agentes para la realización de distintas tareas es una opción a tener en cuenta. Por esta razón, se debería volver a considerar el *framework* de

agentes a utilizar, pues dependiendo de las necesidades y objetivos de una futura plataforma, existen otros *frameworks* que permiten la capacidad de comunicación entre los distintos agentes, como es el *Agent Development Kit* [51] de Google Cloud, que incorpora protocolos de comunicación entre agentes como *A2A* o *Agent to Agent* y otros conceptos como el *MCP* o *Model Context Protocol*, conceptos que se podrían profundizar en trabajos futuros.

TÍTULO: Aplicación de Machine Learning e IA Generativa para la predicción y consulta inteligente sobre datos de jugadores de fútbol

AUTOR: SERGIO PÉREZ-BAQUERO ROBISCO

TITULACIÓN: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

DIRECTOR DEL PROYECTO: ALEJANDRO VICENTE GONZÁLEZ

YUDITH CARDINALE

FECHA: [05] de 2025

Bibliografía

- [1] *Portal Firma - Aplicaciones de firma*, Firmaelectronica.gob.es, 2023. dirección: <https://firmaelectronica.gob.es/Home/Ciudadanos/Aplicaciones-Firma.html> (visitado 11-03-2023).
- [2] *Adobe Acrobat Reader*, Adobe.com, 2023. dirección: <https://www.adobe.com/es/acrobat/pdf-reader.html> (visitado 12-03-2023).
- [3] L. Bransen y J. Van Haaren, «Measuring soccer players' performances,» *Journal of Sports Analytics*, vol. 6, n.º 2, págs. 123-135, 2020. DOI: 10.3233/JSA-200456.
- [4] L. Pappalardo, P. Cintia, P. Ferragina, E. Massucco, D. Pedreschi y F. Giannotti, «Playerank: data-driven performance evaluation and player ranking in soccer via a machine learning approach,» *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, n.º 5, págs. 1-27, 2019. DOI: 10.1145/3343172.
- [5] A. T. Strand, S. Gautam, C. Midoglu y P. Halvorsen, «Demo: Soccer Information Retrieval Via Natural Queries using SoccerRAG,» en *2024 International Conference on Content-Based Multimedia Indexing (CBMI)*, 2024, págs. 1-5. DOI: 10.1109/CBMI62980.2024.10859233.
- [6] S. Chen, «ProQuest Dissertations & Theses,» ProQuest Dissertations Publishing, 31327926, Tesis doct., University of California, Los Angeles, 2024.
- [7] N. Blomgren, «A Chatbot for Football Analytics: A Deep Dive into RAG, LLM Orchestration and Function Calling,» Dissertation, Tesis doct., KTH Royal Institute of Technology, 2025. dirección: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-368039>.
- [8] AWS, *What is Reinforcement Learning?* Fecha: Junio 2024, 2024. dirección: [https://aws.amazon.com/es/what-is/reinforcement-learning/#:~:text=El%20aprendizaje%20por%20refuerzo%20\(RL,utilizan%20para%20lograr%20sus%20objetivos..](https://aws.amazon.com/es/what-is/reinforcement-learning/#:~:text=El%20aprendizaje%20por%20refuerzo%20(RL,utilizan%20para%20lograr%20sus%20objetivos..)
- [9] AWS, *What is a Neural Network?* Fecha: Junio 2024, 2024. dirección: <https://aws.amazon.com/es/what-is/neural-network/#:~:text=Una%20red%20neuronal%20es%20un,lo%20hace%20el%20cerebro%20humano.>
- [10] AWS, *What is a Recurrent Neural Network?* Fecha: Junio 2024, 2024. dirección: <https://aws.amazon.com/es/what-is/recurrent-neural-network/>.
- [11] Bookdown, *Árboles de Decisión: Parte I*, Fecha: Junio 2024, 2024. dirección: <https://bookdown.org/content/2031/arboles-de-decision-parte-i.html>.
- [12] CienciaDeDatos, *Random Forest en Python*, Fecha: Junio 2024, 2024. dirección: https://cienciadedatos.net/documentos/py08_random_forest_python.html.
- [13] CodificandoBits, *Bosques Aleatorios*, Fecha: Junio 2024, 2024. dirección: <https://www.codificandobits.com/blog/bosques-aleatorios/>.
- [14] DatabitAI, *Métricas de Evaluación en Machine Learning*, Fecha: Junio 2024, 2024. dirección: <https://databitai.com/machine-learning/metricas-de-evaluacion-en-machine-learning/>.

- [15] Í. Gómez Carvajal, «Estudio de técnicas de data science para la predicción de rendimientos deportivos,» Tesis de máster, Tesis doct., UAM, 2021.
- [16] CienciaDeDatos, *Selección de Predictores: Subset Selection, Ridge, Lasso, Dimension Reduction*, Sitio web, Fecha: Junio 2024, 2024. dirección: https://cienciadedatos.net/documentos/31_seleccion_de_predictores_subset_selection_ridge_lasso_dimension_reduction.
- [17] Datacamp, *Funciones de activación en redes neuronales*, Fecha: Junio 2024, 2024. dirección: <https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks>.
- [18] Gamco, *Perceptrón Multicapa (MLP)*, Fecha: Junio 2024, 2024. dirección: [https://gamco.es/glosario/perceptron-multicapa-mlp/#:~:text=El%20Perceptr%C3%B3n%20Multicapa%20\(MLP%2C%20por,el%20campo%20del%20aprendizaje%20autom%C3%A1tico..](https://gamco.es/glosario/perceptron-multicapa-mlp/#:~:text=El%20Perceptr%C3%B3n%20Multicapa%20(MLP%2C%20por,el%20campo%20del%20aprendizaje%20autom%C3%A1tico..)
- [19] KeepCoding, *Importancia de los Kernels para los SVM*, Fecha: Junio 2024, 2024. dirección: <https://keepcoding.io/blog/importancia-de-los-kernels-para-los-svm/>.
- [20] M. Kubat, *An Introduction to Machine Learning*. Springer, 2017.
- [21] T. Hastie, R. Tibshirani y J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.
- [22] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning, Chapter 6: Deep Forward Networks*, Fecha: Junio 2024, 2024. dirección: <https://www.deeplearningbook.org/contents/mlp.html>.
- [23] Medium, *Un Lego a la Vez: Explicando la Matemática de Cómo las Redes Neuronales Aprenden*, Fecha: Junio 2024, 2024. dirección: <https://medium.com/latinxinai/un-lego-a-la-vez-explicando-la-matem%C3%A1tica-de-como-las-redes-neuronales-aprenden-ae582ab91da6>.
- [24] Medium, *Bosques aleatorios*, Fecha: Junio 2024, 2024. dirección: <https://medium.com/@hpumah/bosques-aleatorios-482163ace92e>.
- [25] R. F. Casal, *Aprendizaje Estadístico: GLM*, Fecha: Junio 2024, 2024. dirección: https://rubenfcasal.github.io/aprendizaje_estadistico/reg-glm.html.
- [26] Datarmony, *Aprendizaje supervisado. Algoritmos y ejemplos*, Fecha: Junio 2025, 2025. dirección: https://www.datarmony.com/blog/aprendizaje-supervisado-algoritmos-ejemplos/?_adin=11734293023.
- [27] IBM, *Unsupervised Learning*, Fecha: Junio 2025, 2025. dirección: <https://www.ibm.com/es-es/topics/unsupervised-learning#:~:text=El%20aprendizaje%20no%20supervisado%2C%20tambi%C3%A9n,necesidad%20de%20ninguna%20intervenci%C3%B3n%20humana..>
- [28] Telefonica, *Aprendizaje no supervisado*, Fecha: Junio 2024, 2024. dirección: <https://www.telefonica.com/es/sala-comunicacion/blog/que-es-aprendizaje-no-supervisado/>.

- [29] Gamco, *Ejemplos aprendizaje automático*, Fecha: Junio 2025, 2025. dirección: <https://gamco.es/6-ejemplos-real-de-aprendizaje-automatico/>.
- [30] InteractiveChaos, *Soft Margin SVM*, Fecha: Junio 2025, 2025. dirección: <https://interactivechaos.com/es/manual/tutorial-de-machine-learning/soft-margin-classifiers>.
- [31] Xataka, *Aprendizaje por refuerzo*, Fecha: Junio 2025, 2025. dirección: <https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-aprendizaje-refuerzo>.
- [32] APD, *Casos de uso aprendizaje por refuerzo*, Fecha: Junio 2025, 2025. dirección: <https://www.apd.es/que-es-reinforcement-learning/>.
- [33] KeepCoding, *Problemas linealmente y no linealmente separables*, Fecha: Junio 2025, 2025. dirección: <https://keepcoding.io/blog/problemas-lineal-y-no-linealmente-separables/>.
- [34] InteractiveChaos, *Sigmoid Kernel*, Fecha: Junio 2025, 2025. dirección: <https://interactivechaos.com/es/manual/tutorial-de-machine-learning/sigmoid-kernel>.
- [35] Knoldus, *Single Layer Perceptron*, 2025. dirección: <https://blog.knoldus.com/complete-guide-to-single-layer-perceptron-with-implementation/>.
- [36] CienciaDeDatos, *Kernels SVM (proviene de libro ISLR)*, Fecha: Junio 2025, 2025. dirección: https://cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines.
- [37] CienciasInSeso, *Redes neuronales*, Fecha: Junio 2025, 2025. dirección: <https://www.cienciasinseso.com/redes-neuronales/>.
- [38] R. Patil y V. Gudivada, «A review of current trends, techniques, and challenges in large language models (LLMs),» *Applied Sciences*, vol. 14, n.º 5, pág. 2074, 2024.
- [39] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun e Y. Zhang, «A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly,» *High-Confidence Computing*, pág. 100 211, 2024.
- [40] Z. Pu, Y. Pan, S. Wang et al., «Orientation and Decision-Making for Soccer Based on Sports Analytics and AI: A Systematic Review,» *IEEE/CAA Journal of Automatica Sinica*, vol. 11, n.º 1, págs. 37-57, 2024. DOI: 10.1109/JAS.2023.123807.
- [41] «API-Football Pricing Plans.» (), dirección: <https://www.api-football.com/pricing> (visitado 21-09-2025).
- [42] «¿Qué es Reinforcement Learning?» (), dirección: <https://www.apd.es/que-es-reinforcement-learning/> (visitado 04-10-2025).
- [43] «SQLite.» (), dirección: <https://sqlite.org/> (visitado 21-09-2025).
- [44] «Draw.io.» (), dirección: <http://drawio.com/> (visitado 21-09-2025).
- [45] «API-Football.» (), dirección: <https://www.api-football.com/> (visitado 21-09-2025).
- [46] «Streamlit.» (), dirección: <https://streamlit.io/> (visitado 01-10-2025).

- [47] «Visual Studio Code.» (), dirección: <https://code.visualstudio.com/> (visitado 01-10-2025).
- [48] «Langchain.» (), dirección: <https://www.langchain.com/> (visitado 01-10-2025).
- [49] «Python.» (), dirección: <https://www.python.org/> (visitado 01-10-2025).
- [50] «Overleaf.» (), dirección: <https://www.overleaf.com/about> (visitado 21-09-2025).
- [51] «Agent Development Kit.» (), dirección: <https://cloud.google.com/vertex-ai/generative-ai/docs/agent-development-kit/quickstart?hl=es-419> (visitado 27-09-2025).
- [52] H. Forum, *Regresión y clasificación*, Fecha: Junio 2025, 2025. dirección: <https://forum.huawei.com/enterprise/es/regresi%C3%B3n-vs-clasificaci%C3%B3n-en-machine-learning/thread/667232299315511296-667212895009779712>.
- [53] Medium, *Multi Layer Perceptron*, 2025. dirección: <https://medium.com/@jajp2203/clasificaci%C3%B3n-de-im%C3%A1genes-con-m%C3%A9todos-cl%C3%A1sicos-y-redes-neuronales-14332af4c49c>.
- [54] BID, *Sobreajuste y subajuste*, Fecha: Junio 2025, 2025. dirección: <https://el-bid.github.io/Manual-IA-Responsable/desarrollo-del-modelo-y-validaci%C3%B3n.html>.
- [55] CodificandoBits, *Validación Cruzada: K-Fold Cross Validation*, Sitio web, Fecha: Junio 2024, 2024. dirección: <https://www.codificandobits.com/blog/validacion-cruzada-k-fold-cross-validation/>.
- [56] A. M. Learning, *Métricas en clasificación desbalanceada*, Fecha: Junio 2025, 2025. dirección: <https://www.aprendemachinellearning.com/clasificacion-con-datos-desbalanceados/>.
- [57] B. of Transportation Statistics, *Fuente Dataset*, Fecha: Junio 2025, 2025. dirección: https://www.transtats.bts.gov/databases.asp?Z1qr_VQ=E&Z1qr_Qr5p=N8vn6v10&f7owrp6_VQF=D.
- [58] RedHat, *Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo*, Fecha: Junio 2024, 2024. dirección: <https://www.redhat.com/es/blog/what-aiml-and-why-does-it-matter-your-business>.
- [59] A. Samuel, *Aprendizaje automático*, Fecha: Junio 2025, 2025. dirección: <https://www.kyoceradocumentsolutions.es/es/smarter-workspaces/insights-hub/articles/que-es-el-aprendizaje-automatico.html#:~:text=Asimismo%2C%20el%20aprendizaje%20autom%C3%A1tico%2C%20incluido,aprender%20sin%20estar%20expl%C3%ADcitamente%20programados%E2%80%9D..>
- [60] IBM, *Data leakage*, Fecha: Junio 2025, 2025. dirección: <https://www.ibm.com/think/topics/data-leakage-machine-learning>.

Capítulo 7. Apéndice I: Gráficas complementarias.

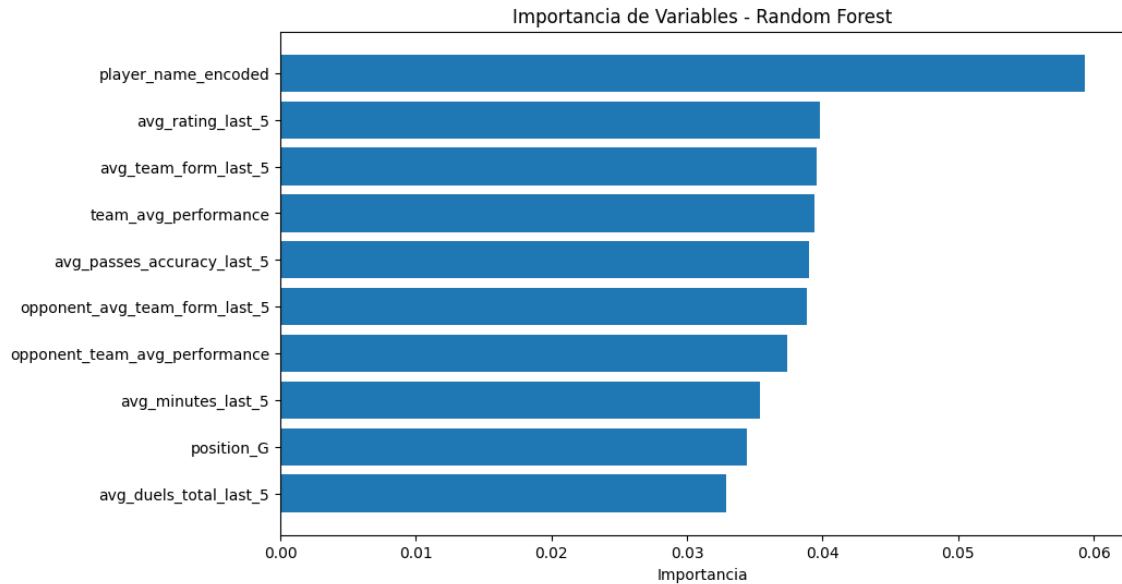


Figura 7.1. Gráfico de la importancia de variables en modelo de *random forest* en la predicción de titularidades.

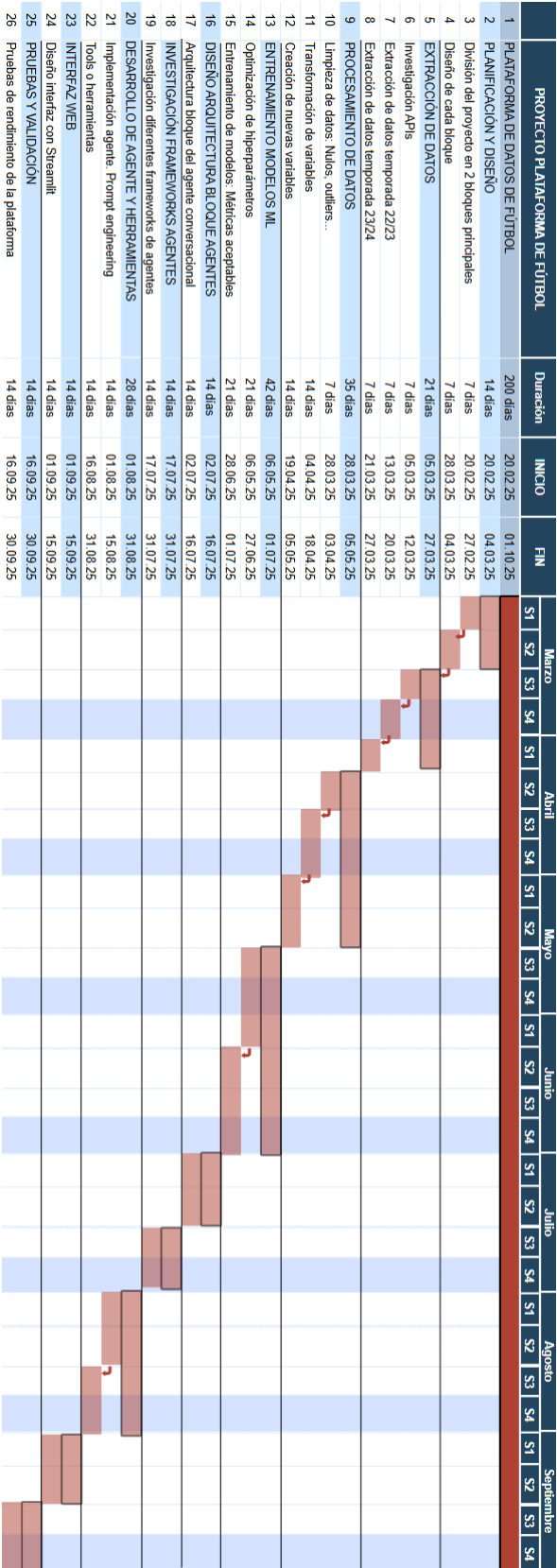


Figura 7.2. Diagrama de Gantt del proyecto. Fuente: Elaboración propia.

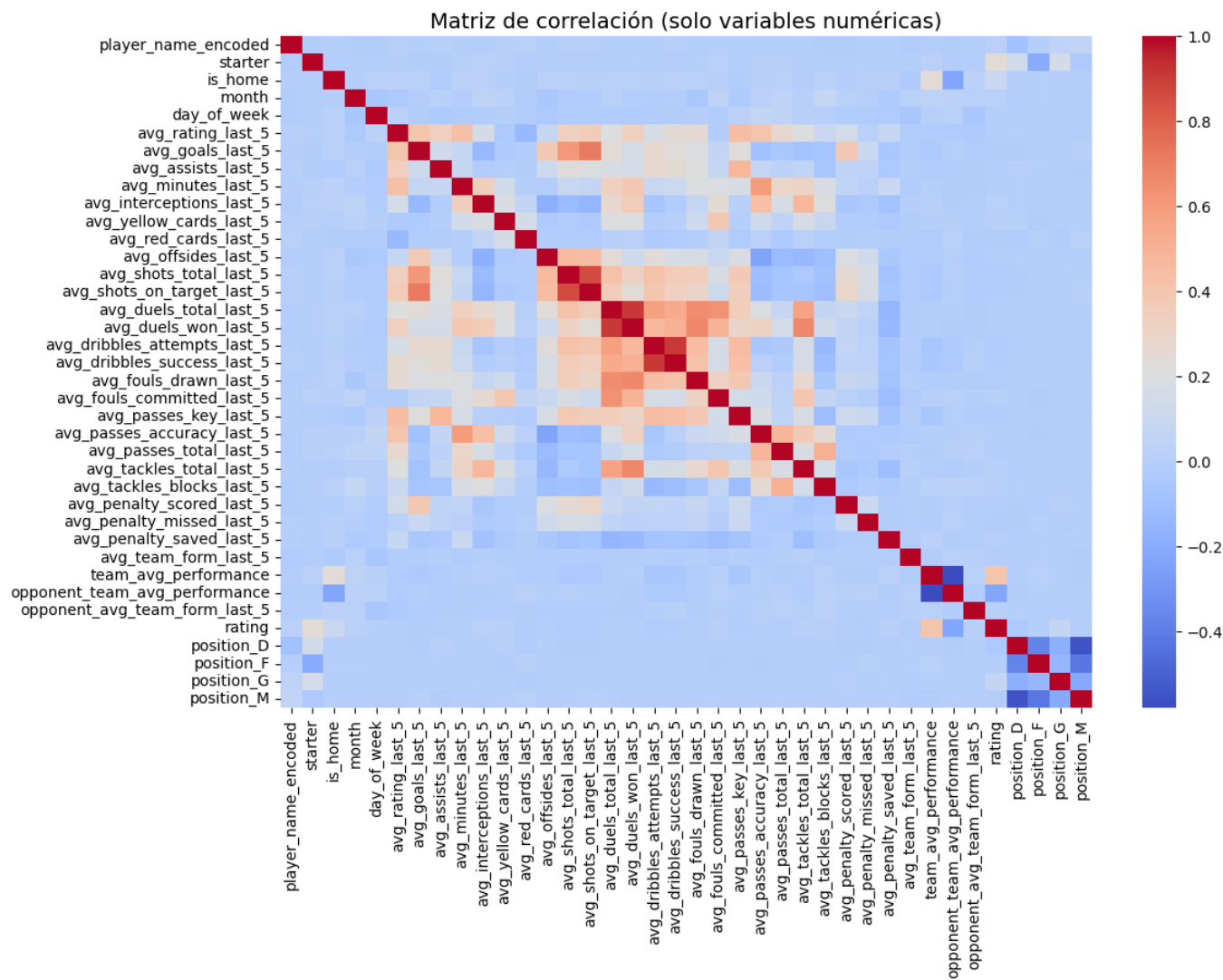


Figura 7.3. Matriz de correlación de variables numéricas