Universidad Autónoma de Madrid

Escuela Politécnica Superior

22|23

Trabajo fin de grado

Predicción del rendimiento de jugadores de la liga de fútbol profesional para Comunio

Alejandro Martín Herrera

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Predicción del rendimiento de jugadores de la liga de fútbol profesional para Comunio

Autor: Alejandro Martín Herrera
Tutor: Alejandro Bellogín Kouki

enero 2023

Alejandro Martín Herrera
Predicción del rendimiento de jugadores de la liga
de fútbol profesional para Comunio

Alejandro Martín Herrera
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A la UAM, por poner a mi disposición los conocimientos adquiridos.
A mi tutor, por su enorme disponibilidad y sus valiosas críticas.
A mis profesores, por todo lo aprendido a lo largo del grado.
A mi familia y amigos, por estar siempre ahí.

He fallado una y otra vez a lo largo de mi vida.
Es por eso por lo que he tenido éxito.

Michael Jordan

Resumen

El análisis de datos y su potencial aplicación en muchos campos y disciplinas es un hecho, y el deporte no es una excepción. En el caso del fútbol y, en concreto, en la gestión deportiva, se ha hecho imprescindible el uso de herramientas tecnológicas que ayuden a tomar mejores decisiones. Estas herramientas utilizadas por los clubes de fútbol en el mundo real han sido trasladadas al mundo virtual y sirven para confeccionar aplicaciones en juegos de fútbol *on-line*, como es el caso de *Comunio*, un juego donde el usuario adquiere el rol de gestor deportivo de un club virtual.

Este proyecto se basa en el desarrollo de un modelo predictivo de puntuación para jugadores en base a su rendimiento real. Esta puntuación será la base para la toma de decisiones en cuanto a fichajes y estrategias deportivas, garantizando los resultados más exitosos.

Para ello se ha diseñado e implementado, en el lenguaje de programación Python, un algoritmo que se ha entrenado con datos basados en estadísticas de jugadores. Tras la selección de páginas web como fuente, se han usado rastreadores y procesadores web (*crawling/scraping*) para la recopilación e indexación de los datos. Se han valorado sistemas de predicción que utilizan técnicas de aprendizaje automático en relación con modelos de regresión, tanto lineal como logística, también se han analizado otras posibles soluciones de regresión, como árboles de decisión, bosques aleatorios y redes neuronales. Se describirán todas estas técnicas junto con las tecnologías web utilizadas en el proyecto.

Tras todo lo anterior, se detallarán los requisitos del proyecto software, así como su diseño y desarrollo, para finalmente poder implementarlo en una plataforma web que se ha creado para tal fin, donde se podrán aplicar las predicciones obtenidas y se mostrará cómo los usuarios del juego *Comunio* pueden beneficiarse de esta herramienta.

Palabras clave

Comunio, aprendizaje automático, modelo de predicción, modelo de regresión, página web

Abstract

Data analysis and its potential application in many fields and disciplines is a fact, and sport is not an exception. In the case of football and, specifically, in sports management, it has become essential to use technological tools to help make better decisions. These tools used by football clubs in the real world have been transferred to the virtual world and are used to create applications in online football games, as is the case of *Comunio*, a game where the user takes on the role of sports manager of a virtual club.

This project is based on the development of a predictive scoring model for players based on their real performance. This score will be the basis for decision making in terms of transfers and sporting strategies, guaranteeing the most successful results.

For this purpose, an algorithm has been designed and implemented in the Python programming language, which has been trained with data based on player statistics. After the selection of web pages as sources, crawlers and web processors (crawling/scraping) have been used to collect and index the data. Prediction systems using machine learning techniques have been evaluated in relation to regression models, both linear and logistic, and other possible regression solutions, such as decision trees, random forests, and neural networks, have also been analysed. All these techniques will be described together with the web technologies used in the project.

After all the above, the requirements of the software project will be detailed, as well as its design and development, to finally implement it in a web platform that has been created for this purpose, where the obtained predictions can be applied, and it will be shown how the users of the *Comunio* game can benefit from this tool.

Keywords

Comunio, machine learning, prediction model, regression model, web site

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

El fútbol no es solo un deporte. Es también un negocio que mueve una enorme cantidad de dinero en todo el mundo, lo cual despierta el interés no solo de las disciplinas que tradicionalmente están vinculadas con este deporte (psicología, nutrición o entrenamientos, entre otras) sino también del mundo de la empresa, de la gestión económica y, en las últimas décadas, de la tecnología.

Esta tecnología permite la recogida y tratamiento de un sinfín de datos que se obtienen tanto de la actividad deportiva de los jugadores como de su rendimiento personal y sus características, de manera que se hace posible un análisis objetivo de los datos obtenidos a la hora de optimizar los recursos en un club deportivo, mediante una adecuada toma de decisiones a la hora de hacer fichajes o, incluso, permitiendo modificar estrategias de juego a la hora de preparar los partidos.

Esto permite una evaluación continua de las marcas personales de los jugadores con propuestas de mejora individuales, así como una dirección determinada en las estrategias, tanto deportivas como extradeportivas. De esta forma, podemos decir que los macrodatos han modificado la manera de gestionar los equipos de fútbol, incluso la filosofía misma del deporte, ya que las decisiones sobre los clubes ya no las toman “ojeadores” ni entrenadores, sino matemáticos, científicos y analistas de datos que son capaces de interpretar y predecir, en base a datos objetivos, resultados probables en cuanto a fichajes o estrategias de juego.

El deporte en general ha salido del mundo de las sensaciones e intuiciones y se ha adentrado en el mundo de los datos.

1.1. Motivación

Los juegos *on-line* son una de las actividades en expansión en la actualidad y una nueva posibilidad de negocio para los creadores de plataformas, aplicaciones y páginas web. Dado que las nuevas herramientas tecnológicas basadas en el análisis de datos y el uso de algoritmos ha demostrado su eficacia y eficiencia en la gestión deportiva de los clubes de fútbol en el mundo real, también se están empleando estos métodos para las competiciones que se producen en este tipo de juegos.

Dentro de los juegos on-line relacionados con el deporte y en concreto con el fútbol, se ha elegido un juego que destaca la parte estratégica y de toma de decisiones de este deporte: el fútbol desde la gestión. Este juego nos propone un perfil emergente en la actualidad del mundo del deporte: el gestor deportivo, que se configura como un nuevo perfil profesional, con el objetivo de gestionar las dos variantes más importantes de un negocio: el producto (en nuestro caso los futbolistas) y el mercado (en nuestro caso las competiciones).

Los resultados positivos en las competiciones generarán ingresos, por lo que el gestor deportivo deberá incorporar las herramientas tecnológicas necesarias para favorecer los mejores rendimientos. Hay que añadir un factor emergente, las redes sociales permiten “captar el pulso de los seguidores” de clubes y deportistas, lo que obliga también al gestor a dominar estos medios. Un club es una empresa.

Existen en la actualidad múltiples aplicaciones de juegos donde el usuario simula ser un gestor deportivo de un equipo de fútbol, como son Comunio, BeManager, Futmondo, Biwenger, o La Liga Fantasy, entre otros. Implementaremos una herramienta predictiva en uno de los juegos más exitosos: Comunio.

Figura 1.1: Logotipo de Comunio.

Comunio es un juego online donde el usuario adquiere el rol directivo de un equipo de fútbol formado por una serie de jugadores que se encuentran actualmente en activo en la liga de fútbol profesional española y que deben ser fichados por el usuario del juego con el objetivo de demostrar sus habilidades y estrategia como gestor-entrenador del club. En la Figura 1.1 puede verse el logotipo de Comunio.

Se trata de un juego de competición, que funciona mediante una clasificación por puntos, los cuales son atribuidos a los jugadores basándose en su rendimiento real en los partidos. Estas puntuaciones se calculan en base a dos parámetros: por un lado se valoran los resultados de un algoritmo llamado *SofaScore*, que pondera y evalúa las estadísticas de los jugadores en cada partido otorgándoles una puntuación, y por otro lado se aplican a dichas puntuaciones una serie de valoraciones numéricas realizadas por periodistas de los diarios deportivos más conocidos en este país (Marca, As y Mundo Deportivo) en base a su experiencia y conocimientos sobre este medio.

El juego tiene como objetivo final acumular el máximo número de puntos posibles, lo que determinará ser el ganador de la liga Comunio, lo que se traducirá en haber sido capaz de llevar la mejor gestión de un equipo deportivo en relación con sus competidores.

1.2. Objetivos

El objetivo general de este Trabajo de Fin de Grado será diseñar un modelo de predicción de la puntuación que obtendrán los jugadores en su próxima jornada, de forma que les sea de utilidad a usuarios de juegos como Comunio, para saber cuáles son los mejores jugadores que pueden adquirir para su equipo, así como para decidir cuál es la mejor alineación en base a la plantilla de la que disponen.

El objetivo específico consiste fundamentalmente en diseñar e implementar en el lenguaje de programación Python un algoritmo que sea capaz de realizar predicciones de la manera más precisa posible usando técnicas de aprendizaje automático. Dichas técnicas consisten en varios modelos de regresión que se aplican sobre un conjunto de datos obtenido, realizando técnicas de rastreo web (del inglés, *web scraping/crawling*).

Además, se mostrarán los resultados de este trabajo en una plataforma web creada desde cero, donde se podrán comprobar las predicciones obtenidas sobre los jugadores de la liga de fútbol profesional española, de forma que el usuario podrá usar esta herramienta desde una interfaz de usuario.

1.3. Estructura

Capítulo primero: Introducción. Se explica la motivación, los objetivos y la estructura del trabajo.

Capítulo segundo: Estado del arte. Se describe la información que se ha usado como pilar teórico necesario para comprender el proyecto. Analizando diferentes modelos de aprendizaje automático.

Capítulo tercero: Diseño e implementación. Se explican las decisiones tomadas en cuanto a las cuestiones de diseño y desarrollo del proyecto, además de los detalles sobre la implementación del mismo.

Capítulo cuarto: Pruebas y resultados. Se detallan tanto los experimentos llevados a cabo por medio de las técnicas estudiadas, como la manera en la que se han realizado.

Capítulo quinto: Conclusiones y trabajo futuro. Se muestran las conclusiones del trabajo realizado así como sus posibles mejoras o ampliaciones para el futuro.

2

Estado del arte

Este capítulo proporcionará una primera exploración acerca de los conceptos en los que este trabajo se sustenta. Se presentará información acerca de cómo y cuánto influyen los sistemas de predicción en el mundo del deporte, y más concretamente en el del fútbol. También se introducirán múltiples aspectos, desde cómo funciona la recopilación de los datos a través de programas llamados arañas (del inglés, *spiders*), pasando por el funcionamiento de los diferentes modelos de regresión que se llevarán a cabo, hasta las tecnologías que se utilizarán en la representación del resultado final en forma de aplicación web.

2.1. Predicción en el deporte

Los sistemas predictivos en nuestra vida cotidiana se encuentran mucho más presentes de lo que puede parecer, por ejemplo en la predicción meteorológica o en la predicción de la demanda eléctrica. Las predicciones en el deporte, más comúnmente conocidas como pronósticos deportivos, son aquellas que determina un especialista acerca de un resultado deportivo. Se trata de una cuestión que depende de una gran cantidad de variables, lo que lo convierte en un problema ciertamente complejo.

Al trasladar este problema a la tecnología, la solución radica en realizar un cálculo anticipado de un suceso que está por producirse, este tipo de resultados predictivos son hoy en día cada vez más comunes, de hecho, son muy útiles para tratar de mejorar el rendimiento deportivo y todo lo que lo rodea. Como ya hemos mencionado anteriormente, en este trabajo nos centraremos en una aplicación de ocio que cada vez es más frecuente en el mundo del fútbol profesional denominada Comunio.

Para poder llevar a cabo esta tarea, se van a utilizar diferentes técnicas de aprendizaje automático con el objetivo de lograr una predicción del rendimiento de los jugadores de fútbol profesional de la primera división de la liga española. Para ello, en la sección 2.1.1 se describirá de qué manera influye la ciencia de datos aplicada al mundo del deporte, y en la sección 2.1.2 se comentarán algunas herramientas de aprendizaje automático que hoy en día se utilizan en aplicaciones similares a Comunio.

2.1.1. Ciencia de datos aplicada al deporte

El aprendizaje automático, ya no solo en el fútbol, sino en el mundo del deporte en general, es muy utilizado. Esto ha supuesto un gran avance debido a la ventaja que puede proporcionar en múltiples casos de uso. Encontramos ejemplos en el análisis empírico del béisbol [1], los sabermetristas (que es como se llama a los expertos que analizan esye deporte a través de estadísticas, del inglés *sabermetrics*) reúnen los datos más relevantes del juego para dar respuesta a ciertas preguntas con las que mejorarán considerablemente la toma de decisiones en el juego del béisbol.

Uno de los ejemplos paradigmáticos de cómo los datos influyen en el deporte es el caso de Billy Beane, presidente del equipo de béisbol americano *Oakland Athletics*, el cual a través del análisis de los datos del rendimiento de sus jugadores, junto con otra serie de datos que habían sido previamente infravalorados, consiguió un equipo ganador con un presupuesto muy reducido en 2002. Igualmente, un modelo matemático desarrollado en la universidad de Oxford predijo correctamente que Italia sería campeón en la final de la Eurocopa de 2020, frente a Inglaterra [2].

Existen muchas otras tareas relacionadas con el mundo del deporte donde la ciencia de datos toma protagonismo en aplicaciones prácticas. Por ejemplo, en el hockey sobre hielo se usa una analítica basada en la posesión del disco, los intentos de tiro o los goles esperados, entre otros, donde cada año se crean nuevas estadísticas para poder ajustar lo mejor posible la capacidad de los jugadores a este deporte [3]. En el voleibol, como otro ejemplo característico, recientemente unos nuevos algoritmos son capaces de predecir las acciones de los jugadores durante el propio partido con una precisión que supera el 80 % [4].

En este caso, los algoritmos usan técnicas de aprendizaje automático para, a partir de vídeos de partidos de voleibol, extraer información de cómo deben realizar el análisis. A través de visión por ordenadores, los sistemas analizan la posición en el campo de los jugadores o la inclinación del cuerpo, entre otras muchas características, para determinar en tiempo real cuáles serán los próximos movimientos que realizarán los jugadores.

2.1.2. Aprendizaje automático en aplicaciones de tipo Comunio

Regresando al deporte que es el foco de este trabajo, el fútbol, este es rodeado por herramientas que adquieren cada vez más interés en el mundo de la inteligencia artificial. Existen estimadores que sirven de ayuda acerca de cuáles pueden ser las posibles alineaciones que presenten los equipos antes que se publiquen las oficiales, así como jugadores recomendados, jugadores en racha, y demás datos de interés para el aficionado.

Recientemente se han creado herramientas para los usuarios o jugadores de estas aplicaciones, donde pueden obtener recomendaciones acerca de cuáles son los jugadores más útiles dentro de la

plantilla (con mejor rendimiento futuro), con el objetivo de poderlos alinear en la formación, donde de nuevo el aprendizaje automático juega un papel fundamental. Un ejemplo pueden ser los análisis de mercado de “Jornada Perfecta”, una aplicación especializada en alineaciones y noticias sobre juegos de tipo Comunio.

Al investigar sobre herramientas existentes, se encontró una denominada Automanager, ésta tiene en cuenta multitud de variables del jugador: su estado de forma, su rol en la plantilla, su racha deportiva, sus sanciones, entre muchos otros factores [5]. De hecho, este sistema tiene en cuenta la opinión de los periodistas deportivos acerca de la valoración que hacen sobre el rendimiento de los jugadores, que es algo que, como ya se ha explicado en la sección 1.1, influye con importancia en la puntuación que reciben los jugadores en este tipo de aplicaciones.

2.2. Recopilación de datos

La recopilación de los datos consiste en coleccionar una cantidad determinada de estadísticas de jugadores para poder entrenar los algoritmos de aprendizaje automático y obtener predicciones sobre el rendimiento que obtienen. Lo que mejor conviene son datos estadísticos orientados a aplicaciones de tipo Comunio, aunque también es importante recolectar información estadística normal y corriente de los jugadores de fútbol, además de su nombre e identificación, estos son los datos que buscaremos recolectar por cada jugador (separados por posiciones):

Datos estándares (para todas las posiciones): Posición, clasificación por posición, partidos jugados, titularidades, asistencias, tarjetas amarillas y tarjetas rojas.

Datos estándares (para las posiciones de jugador de campo): Goles anotados y goles anotados de penalti.

Datos estándares (para la posición de portero): Penaltis detenidos y número de porterías a cero logradas.

Datos orientados a aplicaciones de tipo Comunio (para todos los jugadores): Puntuación, media de puntuación, precio de mercado actual, precio de mercado máximo, precio de mercado mínimo, media de las últimas cinco puntuaciones y las puntuaciones por separado de las últimas cinco jornadas.

2.2.1. Arañas de datos

Las arañas de datos son una herramienta muy útil para la recopilación de los datos, una vez obtenida la fuente web de la que extraer la información, el funcionamiento de la araña consiste en recorrer diferentes páginas a las que se accede a partir de múltiples enlaces partiendo de un enlace raíz. Se conoce como araña ya que crea una toda una red de enlaces de la que luego se extraen los datos, simulando conceptualmente una tela de araña.

Esta extracción de los datos y su posterior indexación es muy característica de los rastreadores (del inglés *crawlers*) que serán comentados en la siguiente sección. Hoy en día las arañas son muy utilizadas en las tecnologías de la información, desde en motores de búsqueda como Google o Bing, hasta en seguridad informática.

2.2.2. Crawling/Scraping

Los rastreadores web son el conjunto de métodos de extracción e indexación de información que permiten obtener la información necesaria para crear el conjunto de datos, en inglés se conocen como técnicas de *crawling/scraping*.

El rastreo web (del inglés *web crawling*) es un concepto más general, consiste en la extracción de los enlaces de una página web, mientras que el procesado web (del inglés *web scraping*) se refiere a la obtención de los datos de una o múltiples páginas web [6]. Su funcionamiento consiste en recopilar código codificado en lenguaje web el cual contiene todos los enlaces, donde la araña de forma recursiva accederá a estos, iterando y almacenando la información requerida. Por lo tanto, con la combinación de ambos procedimientos se logra una herramienta capaz de reunir los datos deseados partiendo de un enlace principal de una página web.

La librería Python que se utiliza en este proyecto es *BeautifulSoup*. Su funcionalidad es la extracción de enlaces a partir del código fuente de una página web, dicho código se transformará en objetos de tipo *BeautifulSoup*, a partir de los cuales podrán iterarse para extraer la información deseada.

Además, hay que tener en cuenta un factor muy importante: el agente de usuario (del inglés *User Agent*), es importante cuando la página web elegida para obtener los datos carece de una API (*Application Programming Interfaces*). Siempre que se accede a una página web, el navegador que se utiliza envía un nombre identificando al agente de usuario que está accediendo a dicha web. En esta recolección de información es conveniente poder modificar (sobre todo durante el periodo de pruebas) lo que el navegador web envía como agente de usuario para poder realizar una recopilación de los datos minimizando posibles negativas o bloqueos del servidor. Para lograr este objetivo, se utiliza un módulo Python llamado *UserAgent* importado desde la librería *fake-useragent*, que permite disfrazar con un nombre diferente al identificador del agente de usuario que envía el navegador.

2.2.3. Selección de páginas web

Llegados a este punto, la elección de las páginas web que actuarán de fuente de datos ha sido un factor de alta importancia. Primeramente, al iniciar el proceso de búsqueda de fuentes de datos se consideró una página web muy completa sobre estadísticas de jugadores de la liga de fútbol profesional española. Este es el enlace: (`https://whoscored.com`; [accedido por última vez en Octubre 2022]).

Finalmente, no fue posible decantarse por dicha página web, dado que al hacer pruebas usando arañas, la página detectaba que un proceso automático estaba tratando de acceder a la información de la misma, por lo que siempre rechazaba las peticiones que se realizaban, a pesar de estar utilizando un agente de usuario.

Por este motivo, se investigaron otras alternativas, y finalmente se encontró una página web que era óptima para obtener los datos, debido a que estos estaban orientados a aplicaciones de tipo Comunio, este es el enlace: (`https://comuniate.com`; [accedido por última vez en Diciembre 2022]). En la Figura 2.1 se puede ver una captura tomada de dicha página web.

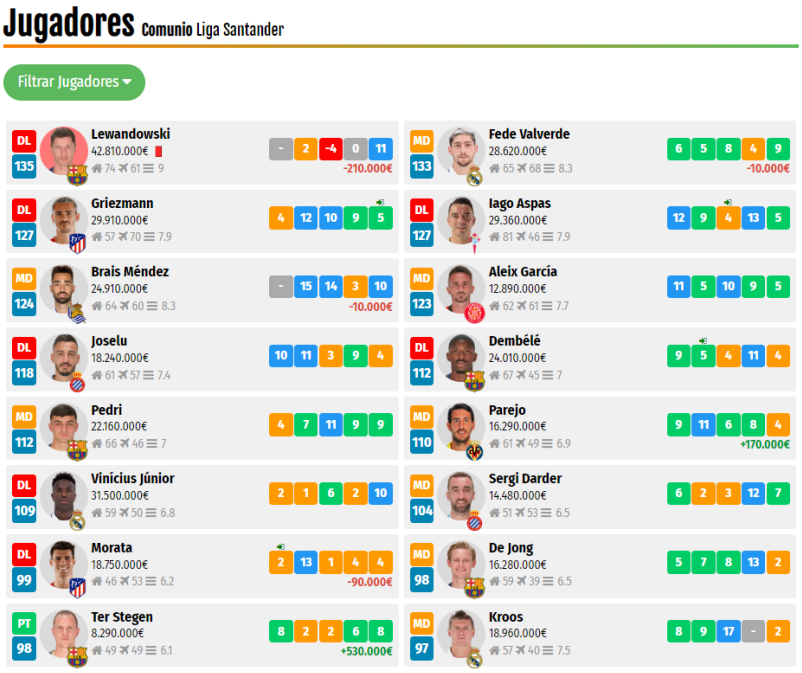

Figura 2.1: Página web comuniate.

La página web seleccionada proporciona información sobre todos los jugadores de todas las plantillas de los equipos de la primera división de la liga de fútbol profesional en España, con lo cual, ha sido elegida como fuente de datos en este TFG.

2.3. Sistemas de predicción

Los sistemas de predicción son un conjunto de técnicas que diseñan y crean previsiones mediante técnicas de inteligencia artificial como las que integran el aprendizaje automático [7]. En esta tecnología de aprendizaje existen una gran cantidad de algoritmos y dependiendo del problema que se pretenda resolver usaremos un algoritmo u otro. A grandes rasgos, estas tareas se dividen en tareas de clasificación y de regresión [8], que se describirán a continuación:

Tareas de clasificación:

Una tarea de clasificación se define como aquella que tiene el objetivo de predecir a qué clase pertenece una determinada serie de datos, refiriéndose a valores discretos. Se debe usar una solución basada en clasificación cuando el resultado a obtener es una clase concreta entre un determinado número de ellas. Por ejemplo, aplicado a nuestro problema podríamos usar clasificación si la cuestión a resolver fuera determinar si un jugador va a disputar o no la siguiente jornada. Este tipo de algoritmos clasifican con una cierta probabilidad, y por norma general se considera como resultado de la clasificación la clase elegida con la probabilidad más alta. Ejemplos de este tipo de algoritmos pueden ser los árboles de decisión, los bosques aleatorios o las redes neuronales.

Tareas de regresión:

En cuanto a las tareas de regresión, estas tienen como objetivo predecir valores continuos, es decir, valores numéricos. Trasladándolo a nuestro ámbito, consistiría por ejemplo en predecir la puntuación de los jugadores en la jornada que está por disputarse, lo cual es precisamente el objetivo principal de este trabajo. Algunos ejemplos de técnicas de aprendizaje automático que se suelen usar en estos modelos son la regresión lineal, la regresión no lineal y la regresión logística entre otros.

Cabe mencionar que muchas de las técnicas mencionadas anteriormente funcionan para ambas soluciones, tanto para clasificación como para regresión. Esto se debe a que las usaremos con una librería Python de código abierto muy conocida en el mundo del aprendizaje automático y modelado estadístico, llamada *scikit-learn* (`https://scikit-learn.org/`), donde en ciertos casos un problema de clasificación puede convertirse en un problema de regresión, encapsulando las predicciones como un valor continuo y prediciendo una probabilidad para cada clase. A continuación dichas técnicas se comentarán con mayor detalle.

2.3.1. Principales modelos de regresión en aprendizaje automático

En esta parte estudiaremos distintos modelos de regresión que han sido empleados en este proyecto, pero antes comenzaremos adentrándonos brevemente en el aprendizaje supervisado, que es el caso que nos atañe. El aprendizaje automático supervisado consiste en tratar con datos que están etiquetados por clases, que es nuestro caso, ya que queremos obtener predicciones numéricas a partir de unos datos etiquetados. En este tipo de aprendizaje usaremos un conjunto de datos formado por múltiples clases etiquetadas, donde trataremos de predecir nuestra clase objetivo en un número determinado de entradas. Esta estructura etiquetada de datos propia del aprendizaje supervisado puede verse de forma gráfica en la Figura 2.2.

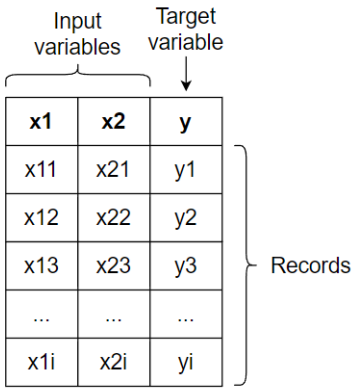

Figura 2.2: Ejemplo visual de conjunto de datos en aprendizaje supervisado.

Regresión lineal:

El modelo de regresión lineal es muy común en aprendizaje automático, se basa en una técnica que consiste en asumir una relación lineal entre las características del conjunto de datos [9]. Este es un algoritmo paramétrico, esto quiere decir que cuando tratamos de aproximar los datos de entrenamiento a la etiqueta que intentamos predecir, lo hacemos con un número fijo de parámetros, independientemente de la cantidad de datos que usemos para entrenar nuestro modelo.

En un modelo de regresión lineal se considera un peso determinado para cada clase, esa es la manera de parametrizar cada clase del conjunto de datos, y dichos pesos se ajustarán para aproximarse de la forma más precisa a nuestra clase objetivo, que es la que se desea predecir.

El modelo de regresión lineal se calcula según la siguiente ecuación 2.1:

$$y = \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{2.1}$$

Donde y es el valor de la clase a predecir, x_i son los datos de entrenamiento, y θ_i se refiere al peso asignado a cada clase de entrenamiento.

A la hora de optimizar el modelo de regresión lineal para saber cómo de preciso es, debemos definir la función de coste o función de pérdida (ecuación 2.2), que sirve para medir el error del algoritmo. La función de coste en este modelo calcula el error cuadrático medio:

$$C(\theta) = \frac{1}{2}\sum(y' - y)^2 \tag{2.2}$$

En esta ecuación y hace referencia al valor a predecir, mientras que y' representa el valor predicho, por lo tanto, se calcula el cuadrado de la diferencia en todas las predicciones realizadas, para posteriormente calcular el error medio considerando el coste (θ) del algoritmo.

Regresión logística:

La regresión logística es un algoritmo de aprendizaje automático supervisado diseñado para resolver tareas de clasificación, este tipo de problemas plantean la clase objetivo como una etiqueta categórica, es decir, una etiqueta fija. El objetivo de esta regresión es predecir la probabilidad de que una nueva entrada pertenezca a una determinada clase [10].

La función logística usa fundamentalmente el modelo sigmoide, que convierte la salida de la función de pérdida en una probabilidad entre 0 y 1, la ecuación 2.3 representa el modelo sigmoide donde y hace referencia a la clase y $h(y)$ a la probabilidad de dicha clase:

$$h(y) = \frac{1}{1 + e^{-y}} \tag{2.3}$$

Esta función sigmoide se parece a una curva en forma de 'S', que crece desde el 0 hasta estabilizarse en 1, y tiene una representación gráfica como la que aparece en la Figura 2.3:

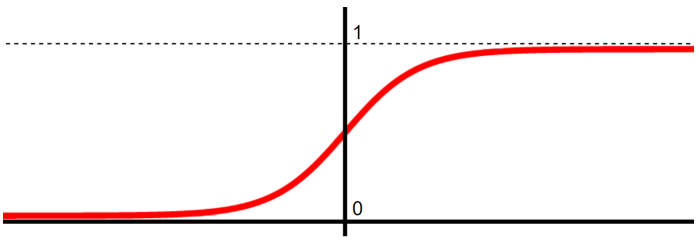

Figura 2.3: Ejemplo gráfico de la función sigmoide.

Como se ha mencionado al inicio de esta sección, la librería que se va a utilizar es scikit-learn, en la cual el modelo de regresión logística resuelve un problema de clasificación (a pesar de su nombre). La ventaja que ofrece esta librería es que este método tiene un funcionamiento similar a otras regresiones a la hora de obtener los resultados de las predicciones, ya que estos son encapsulados. Ambos constan de un método para predecir, y la regresión logística definida como un clasificador calcula las probabilidades de las clases, por lo tanto, se están planteando problemas algo diferentes, aunque de cara a obtener los resultados sean soluciones equivalentes.

2.3.2. Modelos de predicción alternativos

Además de los dos modelos anteriores, que son de los más utilizados en aprendizaje automático, se va a proceder a analizar otras posibles soluciones de regresión, las cuales también se llevarán a la práctica.

Árboles de decisión:

Analizaremos los modelos de regresión basados en árboles de decisión, los cuales pueden representarse como un flujo de decisiones partiendo de un nodo raíz.

Veremos cómo usar este concepto como modelo de predicción. Un árbol se construye dividiendo de forma aleatoria el conjunto de datos en distintas regiones, de modo que los nodos que se clasifiquen en una región determinada, se utilizarán para crear el modelo. De esa manera, podemos usar los árboles tanto para resolver problemas de regresión como para clasificar. Atendiendo a un árbol de regresión, la variable que se obtendrá en la salida será un nodo terminal de valor numérico continuo, y dicho nodo será la media de las muestras de esa región concreta.

Los árboles de decisión son un algoritmo de aprendizaje automático no paramétrico, esto hace que se trate de una técnica más flexible y con un aparente mejor resultado final, a pesar de que generalmente requiere de más datos para poder ser interpretado [11]. El objetivo de este modelo es generar una aproximación a trozos y de forma constante. En la Figura 2.4 puede apreciarse el concepto de esta idea de forma visual.

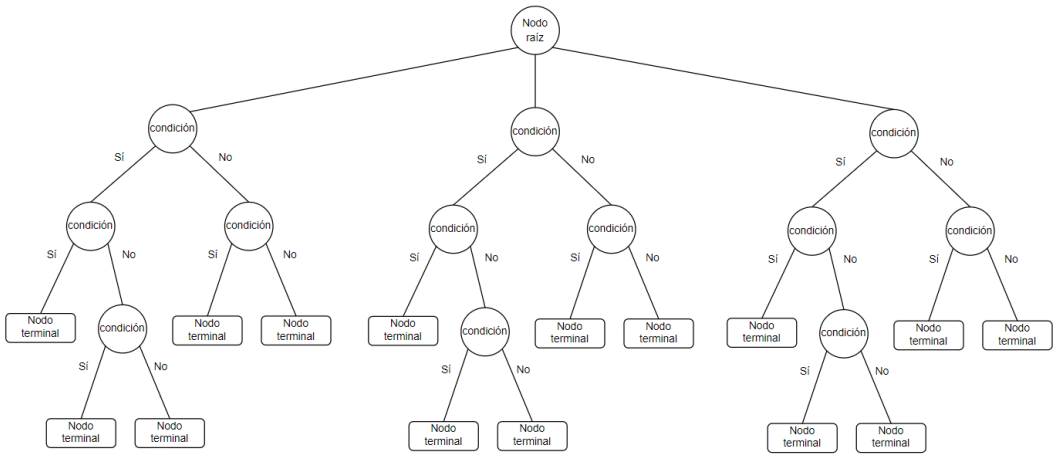

Figura 2.4: Ilustración de árbol de decisión.

Bosque aleatorio:

El bosque aleatorio es un algoritmo muy conocido en aprendizaje automático, el cual construye un clasificador a partir de una combinación de varios clasificadores basados en árboles de decisión, esto concluye en un árbol de decisión a menor escala que suele proporcionar un buen resultado.

Para realizar predicciones con un modelo de regresión basado en bosque aleatorio, se parte de un árbol de decisión entrenado y se estima una predicción para un ejemplo dado considerando la media de los resultados de los árboles anteriores.

Redes neuronales:

Las redes neuronales hoy en día adquieren cada vez más protagonismo, y se basan en emular el funcionamiento del cerebro humano procesando información, en este caso estudiaremos una red neuronal basada en el perceptrón multicapa [13], el cual es un algoritmo de aprendizaje automático que puede realizar tareas de regresión y clasificación. Las redes neuronales se basan en capas, en el caso de un modelo de regresión basado en esta red neuronal, se pueden encontrar diferentes capas de entrada y salida denominadas ocultas. En la Figura 2.5 se puede ver un ejemplo ilustrativo de una red neuronal formada por una capa de entrada, a que le sigue una capa oculta y finalmente mediante una función de salida transforma los valores en un resultado.

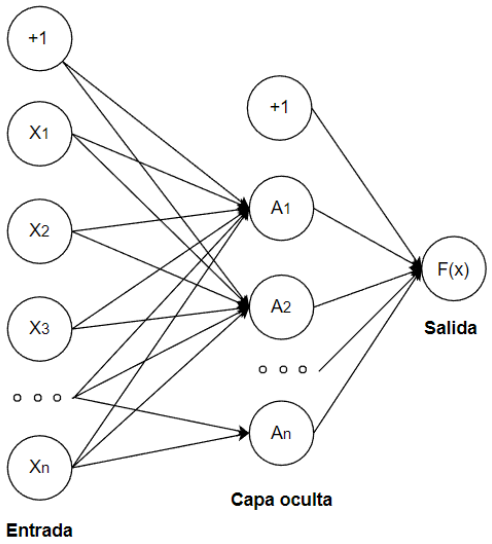

Figura 2.5: Ilustración de red neuronal perceptrón multicapa.

Las redes neuronales tienen ciertas ventajas, como la capacidad de poder aprender de modelos que no son lineales así como de otros modelos en tiempo real, sin embargo también poseen desventajas debido a la necesidad de hiper-parámetros de ajuste referentes al número de capas ocultas de las neuronas, entre otros. Estos parámetros se estudiarán más adelante.

En el caso de la regresión con perceptrón multicapa, este modelo se entrena mediante retropropagación, lo que significa que se propaga el error hacia atrás con el fin de reducirlo desde la salida hasta la entrada de la red neuronal. La salida de esta red puede ser tanto un conjunto de valores continuos como resultados de más de un objetivo, ya que soporta regresión multi-salida.

2.4. Tecnologías web

En esta sección se describirán las tecnologías web que se han utilizado en este trabajo para representar el resultado obtenido mediante una interfaz de usuario donde se pueda ver de forma explícita tanto las predicciones sobre las puntuaciones como el ranking ordenado de los jugadores según estos valores estimados.

El desarrollo de dicha interfaz de usuario se realiza utilizando el lenguaje Python (así como diversas librerías) y el *framework Django* (`https://www.djangoproject.com/`). Los detalles de la implementación se discutirán en el siguiente capítulo.

Django:

A pesar de las diversas opciones que existen, Django ha sido el framework elegido para el desarrollo web, por varios motivos como su sencillez y rapidez en el desarrollo y su gran escalabilidad. El objetivo con esta arquitectura es poder mostrar una versión sencilla del funcionamiento de los algoritmos, donde una posible futura aplicación más compleja y completa podría continuarse debido a la escalabilidad del software. Se valoró también la opción de utilizar el framework de *Flask*, el cuál es también sencillo de utilizar, y que de la misma forma habría servido para representar el resultado en una interfaz de usuario, pero finalmente esa opción se descartó debido a la necesaria instalación de sub-librerías externas y a la falta de experiencia con este framework. Cabe mencionar que no forma parte de los objetivos de este trabajo diseñar una página web compleja, se trata de una versión simple sobre una aplicación con la mera finalidad de mostrar el resultado en una interfaz de usuario de forma visual.

Bootstrap:

Dado que no se ha hecho uso de ninguna API para encapsular los modelos, todo el desarrollo de *backend* y *frontend* es el que forma parte del sistema de Django, en el cual se usa *SQLite* (`https://www.sqlite.org/index.html`) como base de datos. Para la parte de frontend se ha hecho uso del framework *Bootstrap*(`https://getbootstrap.com/`), que es uno de los más conocidos en desarrollo web, el cual permite dotar de un estilo predeterminado a la página web.

3

Diseño e implementación

En este capítulo se detallarán los requisitos del proyecto software, así como el diseño y desarrollo llevado a cabo, y finalmente su implementación en la plataforma web.

3.1. Diseño

En esta parte, se estudiarán los requisitos funcionales y no funcionales de la aplicación, así como la estructura y el ciclo de vida de la misma. Para comprender mejor este análisis, se muestra la Figura 3.1 en la que se representa por módulos la estructura del proyecto.

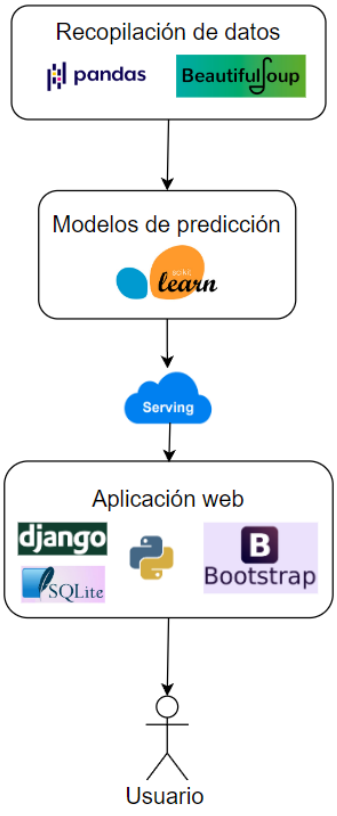

Figura 3.1: Estructura por módulos de la aplicación.

3.1.1. Requisitos funcionales

En esta sección se dividirán los requisitos funcionales en varios grupos: los del conjunto de datos, los de los modelos predictivos y los de la interfaz de usuario.

Requisitos funcionales del conjunto de datos

RF-1.- El conjunto de datos se obtendrá usando técnicas de *crawling/scraping*.

RF-2.- Los datos se basarán en estadísticas de jugadores, tanto estadísticas estándares como orientadas a aplicaciones tipo Comunio.

RF-3.- En el tratamiento de los datos se hará uso de *DataFrames* de la librería Python pandas (`https://pandas.pydata.org/`).

RF-4.- Los datos no se almacenarán en la página web.

RF-5.- Los datos variarán según el momento en el que se coleccionen, debido al rendimiento de los jugadores en ese momento.

Requisitos funcionales de los modelos predictivos

RF-6.- Las predicciones se basarán en las estadísticas del jugador, tanto las orientadas a aplicaciones de tipo Comunio como las que no lo son.

RF-7.- Las técnicas de predicción se basarán en herramientas de aprendizaje automático y usarán modelos de regresión.

RF-8.- El modelo de predicción deberá crear una clasificación ordenada de mayor a menor según la puntuación predicha de los jugadores.

Requisitos funcionales de la interfaz de usuario

RF-9.- La página web tendrá una página principal desde la que se podrá acceder a la funcionalidad de predecir puntuaciones.

RF-10.- El usuario deberá introducir el conjunto de jugadores sobre los que desea obtener predicciones mediante un formulario de texto.

RF-11.- La aplicación se basará en el patrón 'modelo-vista-template' (MVT) que proporciona el framework *Django*.

RF-12.- La interfaz deberá imprimir un ranking ordenado de mayor a menores puntuaciones predichas de los jugadores introducidos.

3.1.2. Requisitos no funcionales

RNF-1.- El lenguaje utilizado será Python, los programas se ejecutarán en un entorno virtual de *Anaconda* (`https://docs.anaconda.com/`).

RNF-2.- El desarrollo se realizará en el sistema operativo Linux, tanto para la parte de pruebas como para la de desarrollo de la aplicación.

RNF-3.- La ejecución del resultado de la aplicación deberá tener un tiempo corto de respuesta.

RNF-4.- La página web será sencilla, no será necesario ningún manual de usuario.

RNF-5.- El estilo del frontend usará el framework *Bootstrap*.

RNF-6.- La base de datos de la aplicación será *SQLite*.

3.1.3. Estructura de la aplicación

Como se puede ver en la Figura 3.1, la estructura de la aplicación es modular, de tal manera que los componentes que la forman están encapsulados, con el objetivo de proporcionar flexibilidad por si se desea en un futuro añadir nueva funcionalidad.

Dichos módulos son los siguientes: La **recopilación de datos**, en la que como hemos visto en la sección 2.2, hemos usado técnicas de crawling/scraping para lograr crear un conjunto de datos que servirá de entrada para los algoritmos. Los **modelos de predicción**, usando diferentes técnicas de regresión de la librería scikit-learn para obtener series de predicciones sobre los datos obtenidos en el primer módulo. Y la **aplicación web**, que servirá para representar el resultado final y donde el usuario podrá interactuar con el sistema mediante una interfaz.

En la Figura 3.2 se representa el diagrama de secuencia de la aplicación, donde se muestra el orden en el que se realizan las interacciones del usuario. Comienza con el usuario accediendo a la interfaz, donde podrá introducir los jugadores sobre los que desea conocer sus futuras puntuaciones, y cuando pulse en el botón de enviar, la web capturará esos jugadores y se los enviará al modelo de predicción, que calculará los resultados de las predicciones para mostrarlos al usuario a través de la misma interfaz.

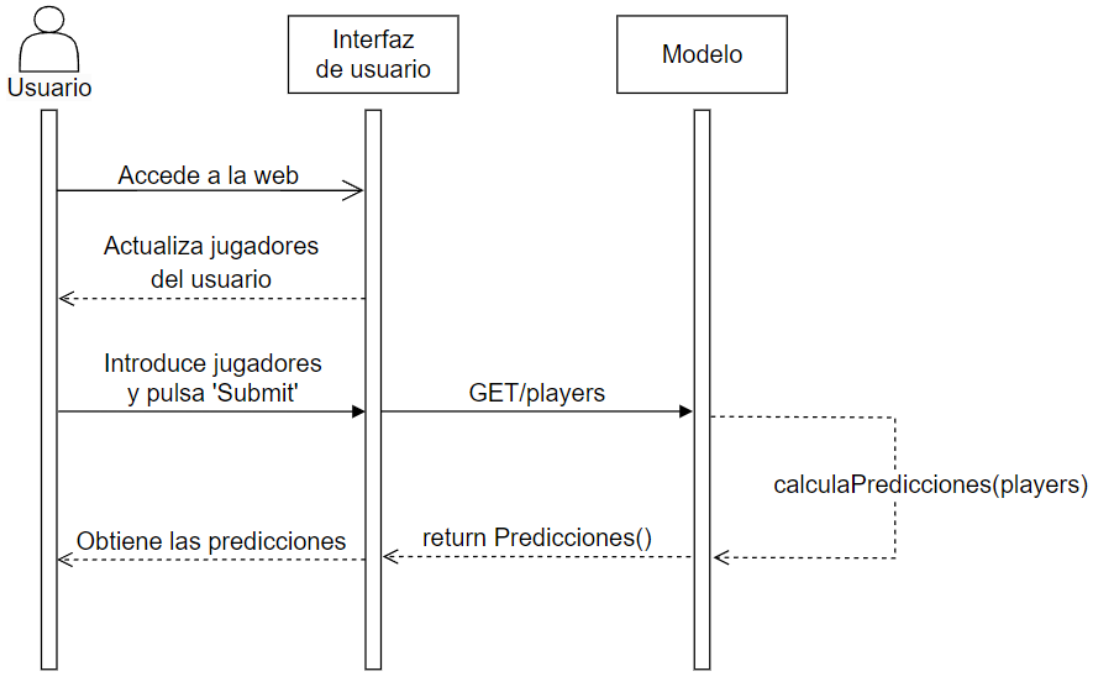

Figura 3.2: Diagrama de secuencia de la aplicación.

3.1.4. Ciclo de vida

Para desarrollar la aplicación se ha optado por un modelo de ciclo de vida iterativo con un ciclo desde la fase de pruebas hasta la de análisis, ya que dependiendo de los resultados obtenidos con los algoritmos, las fases previas de la aplicación pueden corregirse, de hecho se han llevado a cabo múltiples iteraciones para mejorar la aplicación. En la Figura 3.3 se pueden ver las etapas que conforman cada iteración de este ciclo de vida.

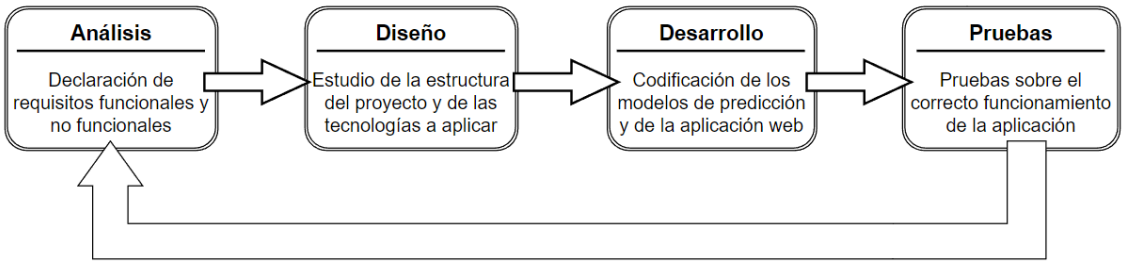

Figura 3.3: Ciclo de vida de la aplicación.

3.2. Implementación y desarrollo

En esta sección se detallará la implementación que se ha llevado a cabo en este proyecto, el desarrollo de la recopilación de los datos, así como el uso de los modelos de predicción y la codificación de la aplicación web.

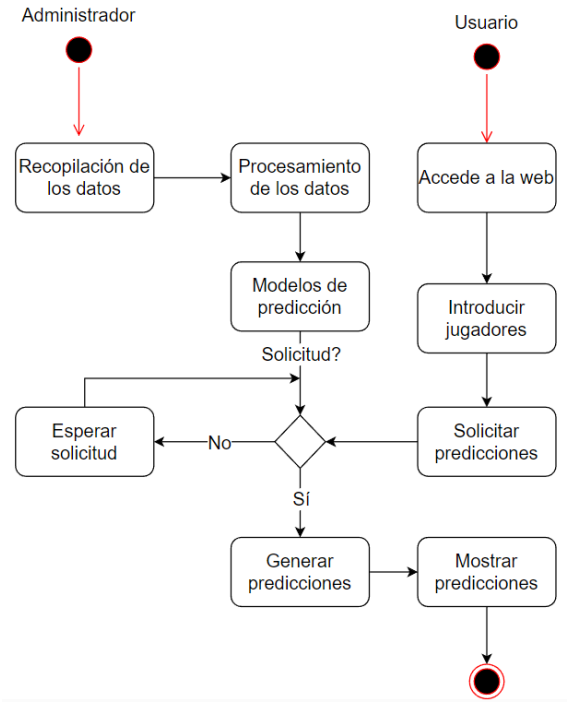

Figura 3.4: Diagrama de flujo de la aplicación.

En la Figura 3.4 podemos ver el diagrama de flujo de la aplicación, partiendo de dos puntos distintos, uno para el administrador del sistema y otro para el usuario que accede a la página web por medio de la interfaz.

3.2.1. Modelos de predicción

Como se ha mencionado en la sección 2.3, los modelos de predicción que se han utilizado se han obtenido de la librería Python scikit-learn. Se ha hecho uso de varios de ellos para poder resolver este problema del cálculo anticipado de la puntuación de los jugadores de la manera más acertada posible, a continuación se explicará cómo se han integrado dichos modelos en el proyecto.

Muchos de los modelos de esta librería hacen uso de hiper-parámetros, que desempeñan un rol importante a la hora de obtener los resultados, estos tienen la función de ajustar lo mejor posible los modelos al problema que se pretende resolver. Saber elegir bien estos parámetros será fundamental para poder sacarles el mejor partido posible a los modelos, para ello haremos uso de una herramienta de la propia librería de *sklearn.model_selection*, llamada *GridSearchCV*, la cual sirve para optimizar

los modelos encontrando los mejores hiper-parámetros en cada uno de ellos, utilizaremos esta herramienta para realizar una búsqueda sistemática de estos parámetros.

Antes de adentrarnos en los modelos, es importante conocer de qué manera se entrenan y validan. Como se mencionó en la sección 2.2, el conjunto de datos contiene información sobre las últimas 5 puntuaciones de cada jugador, la forma de entrenar consistirá en tratar de predecir la penúltima puntuación, para que en el momento de la validación se prescinda de la quinta última, para que así el modelo pueda predecir la última puntuación (la actual), que es el objetivo que se busca. Es decir, si las 5 puntuaciones que tenemos de cada jugador son s_1,..., s_5, entrenaremos con s_1,..., s_3 para predecir s_4, mientras que en validación se entrenará con s_2,..., s_4 para predecir s_5.

También, para mejorar el rendimiento y la eficiencia de los modelos, estandarizaremos los valores de los datos usando la herramienta *StandardScaler* importada de *sklearn.preprocessing*, esto consiste en escalar los datos a la varianza unitaria, que significa dividir todos los valores por la desviación estándar. Por lo general, los algoritmos de aprendizaje automático funcionan mucho mejor cuando usan datos estandarizados en una cierta escala [12], esta estandarización la aplicaremos a todos los modelos que se usarán en este trabajo tras comprobar su considerable mejora de rendimiento en pruebas preliminares, tanto en las propias predicciones como en la reducción del tiempo de entrenamiento.

Regresión lineal:

El modelo de regresión lineal (del inglés, *LinearRegression*) fue importado de *sklearn.linear_model*. El primer método que usaremos en esta regresión será el de *fit* (del inglés, entrenar), como su propio nombre indica dicho método es el que se encarga de entrenar a nuestro modelo, y requiere de dos parámetros de entrada: una matriz X de datos, que serán los datos de entrenamiento, y un vector y que serán los *target values* (del inglés, datos objetivo), que corresponde con la puntuación más reciente del conjunto de datos, bien sea s_5 o s_4, como se explicó anteriormente.

Tras aplicar la función de entrenamiento del modelo, se procede a usar el método *predict* (del inglés, predecir). Esta función se encarga de realizar las predicciones a partir del modelo entrenado, y solo requiere de un parámetro de entrada, que será la matriz de datos de validación para que finalmente el retorno de esta función sea el vector de datos objetivo (las predicciones).

Es importante mencionar que este método de regresión lineal no necesita de hiper-parámetros, por lo que no se precisa de una búsqueda de los mismos.

Regresión logística:

En cuanto a la regresión logística, tras haber estudiado cuál es su funcionamiento anteriormente en la sección 2.3.1, el desarrollo para lograr el resultado de las predicciones ha sido similar al anterior. Se ha entrenado y validado de la misma forma, usando el modelo de regresión logística (del inglés, *LogisticRegression*), importado también de *sklearn.linear_model*.

La aplicación de hiper-parámetros, en este modelo sí que juega un papel importante en términos de eficacia y eficiencia. El primero de ellos es *penalty*, el cual especifica la norma de penalización, cuyo valor y el de los próximos serán los que la herramienta *GridSearchCV* vista anteriormente, encuentre como óptimos. También el parámetro *solver*, que define el algoritmo concreto a utilizar, se deberá usar el que considere conjuntos de datos multiclase, ya que se necesitarán gestionar las pérdidas multinomiales. El número máximo de iteraciones que se desean ejecutar (*max_iter*). Y por último *multi_class*, en función del *solver* que se esté utilizando.

Árboles de decisión:

Este modelo se ha implementado de forma similar a los anteriores, el nombre del modelo es *DecisionTreeRegressor* y se ha importado de *sklearn.tree*. Se han usado hiper-parámetros característicos de los árboles como *max_depth*, que determina cuál será la profundidad máxima del árbol; así como el parámetro que actúa de criterio para medir la calidad de las divisiones de los nodos en los árboles, llamado *criterion* y finalmente *splitter*, que determinará cómo se realizará dicha división de los nodos.

Bosque aleatorio:

El nombre de este modelo es *RandomForestRegressor* y se ha importado de *sklearn.ensemble*, de nuevo, de igual forma que los anteriores.

En este caso se ha hecho uso de dos hiper-parámetros: en primer lugar es importante definir el número de árboles que va a haber dentro del bosque, los cuales se especifican en el parámetro *n_estimators*. Y así como se vio en el modelo anterior, también se hará uso de *max_depth* para definir la profundidad máxima del bosque.

Red neuronal:

Por último, la red neuronal la importamos de *sklearn.neural_network*, el modelo se llama *MLPRegressor*. Y para este modelo hacemos uso de hiper-parámetros como el número máximo de iteraciones (*max_iter*); el parámetro *activation*, para determinar la función de activación de las capas ocultas de la red neuronal; también el parámetro *solver*, para optimizar el peso, en función del tamaño del conjunto de datos; y por último el número de neuronas que habrá en la capa oculta, especificado en *hidden_layer_sizes*.

3.2.2. Métricas de evaluación

Con el objetivo de mejorar lo máximo posible la precisión de los modelos, se ha hecho uso de ciertas métricas de evaluación, usando una conocida librería Python compuesta de herramientas y algoritmos matemáticos, llamada *SciPy* (`https://docs.scipy.org/`). El uso de estas métricas nos interesa fundamentalmente para reducir el error al predecir las puntuaciones o acertar con el ranking de los jugadores, lo que nos ayudará a saber cuál es el mejor modelo.

Para comprobar cómo de bueno es el modelo que se va a utilizar, se comparan dos conjuntos de datos: los datos esperados (a predecir) y las predicciones. Como hemos dicho, no nos interesa tanto saber el error en las predicciones sino determinar si el ranking predicho y el real se parecen. Para ello, usaremos coeficientes de correlación, estos varían entre -1 y +1, siendo 0 la ausencia de correlación entre dichos conjuntos de datos, lo que querría decir que los dos rankings no se parecen, mientras que un número positivo (o negativo) cercanos a 1 indica que existe una correlación positiva (o negativa) entre ambos.

Coeficiente de correlación de *Pearson*:

El coeficiente de correlación de *Pearson* mide la relación lineal que existe entre dos conjuntos de datos distribuidos de forma equivalente, y se calcula según la ecuación 3.1, donde m_x es la media del conjunto de datos x, m_y es la media del conjunto de datos y y r es el coeficiente de correlación:

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2 \sum(y - m_y)^2}} \tag{3.1}$$

Coeficiente de correlación de *Spearman*:

El coeficiente de correlación de *Spearman* hace una medición no paramétrica sobre la monotonía (si conserva el orden dado) de la relación entre dos conjuntos de datos. A diferencia de la correlación de *Pearson*, la de *Spearman* es fiable aunque los dos conjuntos no se distribuyan de forma similar, es decir, que se trate de una correlación no lineal.

Coeficiente de correlación de *Kendall*:

El coeficiente de correlación de *Kendall* es una medida de correspondencia entre dos clasificaciones de datos, cuanto más se acerque el coeficiente a 1 mayor será ese acuerdo de correspondencia. Se diferencia con la de *Spearman* en que cuantifica la diferencia entre el porcentaje de pares que concuerdan y que no concuerdan, para todos los pares de datos de ambos conjuntos. El coeficiente de correlación de *Kendall* también captura correlaciones no lineales.

3.2.3. Aplicación web

En esta parte, se explicará cómo se ha implementado la página web, cuyo principal objetivo es hacer que el usuario pueda obtener los resultados de estas técnicas de predicción mediante una interfaz gráfica.

Se han utilizado diferentes tecnologías, las cuales se han descrito anteriormente en la sección 2.4. La aplicación se ha desarrollado utilizando el framework de Django. Hay diferentes maneras de utilizar Django, y a pesar de que suele usarse más para la parte backend, el enfoque que se le ha dado en este trabajo es el de poder crear una página web completa para poder mostrar los resultados de los modelos implementados usando plantillas para la parte frontend. Usando Django de esa manera conseguimos combinar ambas partes, lo que se conoce como el patrón de diseño modelo-vista-controlador (MVC), que en Django se redefine como modelo-vista-template (MVT).

De acuerdo con los requisitos indicados al principio de este capítulo, para la parte backend se ha utilizado el lenguaje Python y la base de datos SQLite, mientras que para la parte frontend se ha usado el lenguaje HTML que junto a CSS y a la librería Bootstrap dotan a la interfaz de un estilo predeterminado y consistente.

Tras configurar todo el entorno Django, el proyecto y la aplicación correspondiente, comenzamos el desarrollo creando un modelo jugador en el fichero *models.py*, en el cuál estarán definidos todos los campos de información de cada jugador, que fueron descritos en la sección 2.2. Añadiremos ese modelo al sistema de administración de Django y migraremos la estructura para crearla en la base de datos SQLite y posteriormente poder poblarla con los jugadores usando la librería sqlite3 de Python (`https://docs.python.org/es/3.8/library/sqlite3.html`).

Para poder acceder al modelo creado desde el sistema de administración de Django es necesario crear un superusuario (del inglés, *superuser*) donde se establecerán las claves necesarias para acceder al sistema como administrador, donde este podrá acceder a la información de la base de datos pudiendo modificarla como desee. Todo esto corresponde a la parte de **modelo** del patrón MVT, que es el que maneja todo lo relacionado con la información.

Continuando con el desarrollo, en cuanto a la parte de **template** del patrón MVT, que sería lo que corresponde a la vista en el patrón MVC, se creó una plantilla HTML principal, llamada *home.html*, de la cual extienden las demás templates de la aplicación, esto se hace para mantener una cabecera y un mismo estilo en todas las plantillas.

Además de la principal, se crearon dos templates más: *input.html* que sirve para que el usuario pueda introducir a los jugadores que desee mediante un formulario creado con la librería *forms* de Django, y *output.html* que tiene la finalidad de mostrar el resultado de las predicciones en forma de ranking ordenado de mayor a menor.

Posteriormente, para conectar el modelo y la plantilla, hacemos uso de la **vista** del patrón MVT, que corresponde al controlador en el patrón MVC. En el caso de Django hace de enlace entre el modelo y el template, es el que decide cómo se mostrará la información. La vista en Django se representa en los ficheros *views.py* y *urls.py*, los cuales se conectan entre sí para manejar el flujo de la aplicación. El fichero *views.py* es el que obtiene el resultado de las predicciones y el que se encarga de renderizar esa información junto a la plantilla HTML para poder ser mostrada al usuario con el resultado. En la Figura 3.5 se puede apreciar el aspecto de la página web.

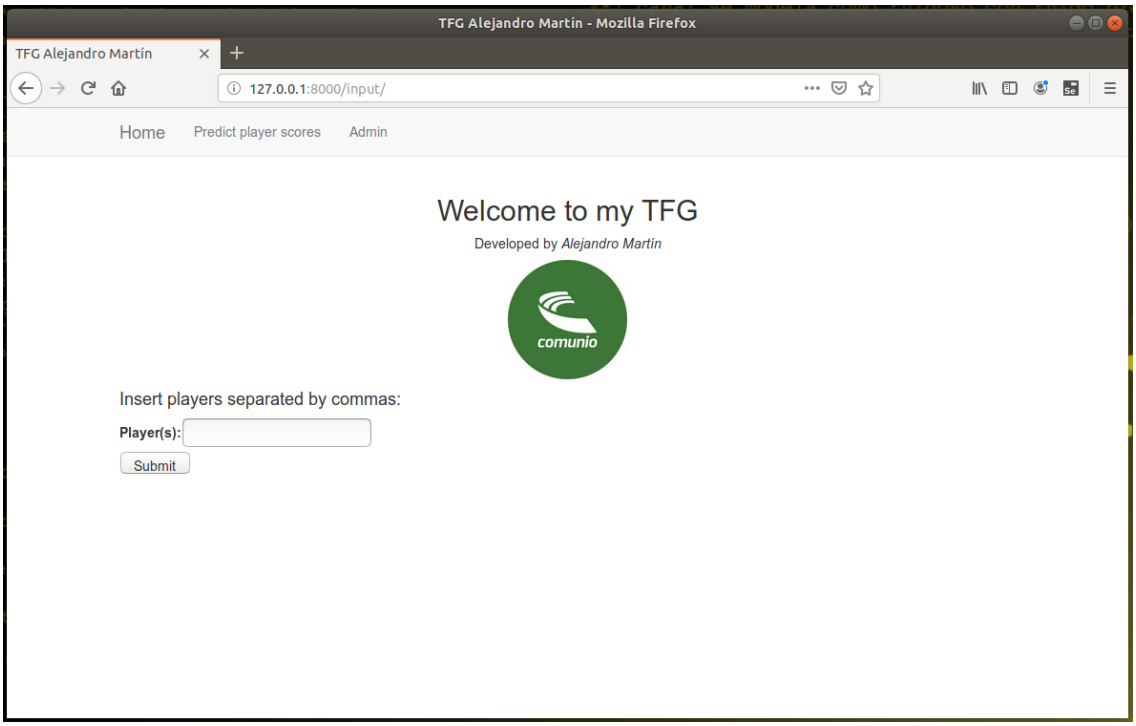

Figura 3.5: Página web principal.

4

Pruebas y resultados

En este capítulo, se analizan y comparan los resultados que se han obtenido a partir de diferentes algoritmos y modelos de predicción. Para ello se describe el entorno donde se han llevado a cabo estos experimentos, se analiza el conjunto de datos, y se discuten los diferentes resultados obtenidos.

4.1. Entorno de pruebas

Para realizar las pruebas se ha elegido un entorno local en el sistema operativo Linux, uno de los motivos es debido a que la aplicación web se ha desarrollado usando el framework de Django y este funciona mucho mejor en este sistema operativo. Además se ha hecho uso de un entorno virtual de Python basado en Anaconda, como se mencionó en los requisitos no funcionales del proyecto (sección 3.1.2).

A pesar de haber elegido dicho entorno para realizar las pruebas de este proyecto, se han considerado otras posibilidades como usar un entorno local en Windows, pero se ha descartado debido a que las tecnologías de las que hacemos uso son más fáciles de utilizar en Linux. En la Tabla 4.1 se pueden ver algunas de las librerías y paquetes del entorno virtual utilizado.

Nombre paquete	Versión
beautifulsoup4	4.11.1
django	3.2.6
fake-useragent	0.1.11
pandas	1.3.5
python	3.7.13
scipy	1.6.2
sklearn	0.24.1
sqlite	3.39.3

Tabla 4.1: Paquetes del entorno virtual de Anaconda.

En la Tabla 4.2 se indican las características del equipo donde se han llevado a cabo todas las pruebas y experimentos.

Componente	Características
Sistema operativo	Ubuntu 18.04.3 LTS
CPU	AMD A12-9700P Radeon R7
Entorno virtual	Python 3.7.13 conda
Memoria RAM	8 GB

Tabla 4.2: Características del equipo.

4.2. Experimentos

En esta sección se analizarán de forma más detallada los experimentos llevados a cabo usando los diferentes modelos de predicción vistos con anterioridad, primeramente comenzaremos con el análisis del conjunto de datos, continuaremos comparando los resultados variando entre diferentes modelos y diferentes hiper-parámetros, y por último discutiremos dichos resultados.

Es importante destacar que los experimentos que se analizarán a continuación no han sido llevados a cabo mediante la interfaz de usuario, sino ejecutando programas en Python que nos permitirán realizar el análisis de forma más rápida y eficiente.

4.2.1. Conjunto de datos

El conjunto de datos que se ha utilizado en este trabajo ha sido enteramente creado desde cero y utilizando técnicas de web scraping/crawling, como se ha explicado en la sección 2.2.2. Estos datos se han almacenado, como consecuencia de este proceso, en un fichero de extensión csv.

En la primera división de la liga de fútbol profesional española hay 540 jugadores, por lo que el conjunto de datos consta de 540 filas (entradas) que corresponden a todos los jugadores, las columnas las cuales fueron descritas en la sección 2.2, se refieren a los datos estadísticos (estadísticas estándares y orientadas a aplicaciones de tipo Comunio) de los jugadores y son 23 campos.

A continuación describiremos las diferentes columnas de datos estadísticos que hay por cada jugador: en primer lugar su **id**, es un valor entero gracias al cual la araña web accede al enlace del jugador en la página; su **nombre**, una simple cadena de caracteres para identificar al jugador; su **posición**, el número 0 hace referencia a que es un portero, el número 1 a un defensa, el número 2 a un centrocampista y el número 3 a un delantero; la **posición en el ranking**, esto se refiere al puesto que ocupa en la clasificación ordenada por puntuación dentro de los jugadores de misma posición, por ejemplo en la Figura 4.1 donde podemos visualizar el conjunto de datos, el primer jugador que aparece, de

nombre Fernando Martínez, dado que su posición tiene valor 0 y su posición en el ranking tiene valor 6, significa que es el sexto mejor portero (con mayor puntuación) de la liga en ese momento.

Continuando con los campos de cada de jugador: sus **partidos jugados**, el número (entero) de partidos en los que ha jugado hasta ese momento; seguido del **porcentaje de partidos jugados**, es importante tener el campo anterior además de este, ya que no es equivalente tener un 100 % de partidos jugados cuando el jugador solo ha disputado 1 sobre 1 (primera jornada liguera) que habiendo disputado 14 sobre 14 (temporada más avanzada); el campo de **titular habitual**, donde el valor 0 significa que dicho jugador no es titular habitual, y un 1 indica que sí lo es; los **goles o penaltis detenidos**, dependiendo de si es un jugador de campo (defensa, centrocampista o delantero) o un portero, respectivamente; y los **goles de penalti o porterías a cero**, de nuevo dependiendo de si se trata de un jugador de campo o de un portero, respectivamente.

Siguiendo con las columnas de datos: las **asistencias**, el número entero de asistencias del jugador; las **tarjetas amarillas**, en número entero; las **tarjetas rojas**, en número entero; los puntos, la puntuación acumulada (en número entero) que lleva el jugador en lo que va de temporada; la **media de puntos**, la media aritmética del campo anterior, un valor decimal que se obtiene al dividir la puntuación acumulada entre los partidos disputados; el **precio actual** de mercado del jugado; el **precio máximo** de mercado; el **precio mínimo** de mercado (estos tres últimos como valores enteros); la **media de las últimas 5 puntuaciones**, un valor decimal que refleja la media aritmética de los 5 campos que vienen a continuación; las **últimas 5 puntuaciones** por separado (números enteros), cabe remarcar que la puntuación de -50 corresponde a no haber jugado esa jornada, dicha puntuación no puede alcanzarse de forma real.

Cabe mencionar que se valoró la opción de prescindir de los jugadores que no puntuaban con regularidad para no introducirlos en los algoritmos, pero tras realizar varias pruebas sin incluirles se vio que se obtenían peores resultados, por lo cual finalmente se incluyen todos los jugadores, de esa manera el algoritmo aprenderá en el entrenamiento a predecir una puntuación nula (-50), cuando el jugador no va a disputar la siguiente jornada.

	Player_id	Name	Position	Ranking_position	Matches_played	...	J_minus4	J_minus3	J_minus2	J_minus1	J_actual
0	3259	Fernando Martinez	0	6	13	...	8	-50	2	11	7
1	1983	Pacheco	0	29	1	...	-50	4	-50	-50	-50
2	2365	Rodrigo Ely	1	31	14	...	3	3	4	3	0
3	2589	Akieme	1	37	14	...	3	5	2	1	4
4	3263	Babic	1	43	13	...	2	-50	4	0	7
..	...	...	...	...	...	...	...	...	...	...	...
535	3210	Nicolas Jackson	3	24	13	...	0	10	2	2	-50
536	1868	Morales	3	44	13	...	1	2	2	0	-50
537	3150	Danjuma	3	43	9	...	2	2	4	4	2
538	1927	Gerard Moreno	3	60	6	...	-50	-50	-50	-50	2
539	2895	Fernando Niño	3	92	0	...	-50	-50	-50	-50	-50

Figura 4.1: Conjunto de datos del TFG.

4.2.2. Estudio de hiper-parámetros en modelos de regresión

Como hemos visto en la sección 3.2.1, los hiper-parámetros juegan un papel muy importante en el resultado de las predicciones obtenidas por los modelos. Saber elegir bien estos parámetros hará que se consiga un resultado considerablemente mejor.

A pesar de que en el modelo de **regresión lineal** no aplican, en el resto de modelos de predicción tienen bastante peso, a continuación se mostrarán los modelos con los mejores hiper-parámetros, que se han obtenido a partir de estimaciones con *GridSearchCV* junto a múltiples pruebas realizadas.

En el caso del modelo de **regresión logística**, los hiper-parámetros óptimos son los que pueden verse en la Tabla 4.3.

Hiper-parámetro	Valor
penalty	l1
solver	saga
max_iter	10000
multi_class	auto

Tabla 4.3: Mejores hiper-parámetros para la regresión logística.

Penalty es el que especifica la norma de penalización, se añade la norma 'l1' dado que es la que mejor resultado genera; *solver*, que se refiere al algoritmo a utilizar para la optimización, se elige 'saga' ya que es una buena opción para los conjuntos de datos con un número de muestras mayor al de características, como es nuestro caso y porque este algoritmo depende de la norma de penalización elegida, que es 'l1'; *max_iter*, en el que se establece un número elevado de iteraciones, que convienen al usar este solver; y por último, *multi_class*, donde podríamos haber elegido tanto 'auto' como 'multinomial', pero con 'auto' se lograban mejores métricas y mejor precisión de predicciones.

Continuando con el modelo de **bosque aleatorio**, los mejores hiper-parámetros aparecen en la Tabla 4.4.

Hiper-parámetro	Valor
n_estimators	11
max_depth	1

Tabla 4.4: Mejores hiper-parámetros para el bosque aleatorio.

N_estimators es el parámetro que se refiere al número de árboles del bosque, se ha decido establecer 11 árboles, ya que con valores superiores apenas mejoraba el rendimiento y cuando el número de árboles era inferior en algunos casos los coeficientes de correlación y la precisión del modelo decrecían considerablemente; la profundidad del árbol, *max_depth*, donde puede tener un valor nulo o un valor específico, en este caso no nos conviene que sea nulo ya que los nodos de los árboles se

expandirían en exceso y este problema es multiclase. Siguiendo las estimaciones de *GridSearchCV* se ha decidido establecer una profundidad igual a 1.

En cuanto al modelo de **árbol de decisión**, en la Tabla 4.5 aparecen los hiper-parámetros óptimos:

Hiper-parámetro	Valor
splitter	best
max_depth	2
criterion	friedman_mse

Tabla 4.5: Mejores hiper-parámetros para el árbol de decisión.

Donde el parámetro *splitter* define cómo será la división de los nodos, la elegida es 'best', descartando el criterio de elección aleatorio; *max_depth* para la profundidad del árbol, en el cuál se ha comprobado que el mejor resultado se consigue con un árbol de máximo 2 niveles de profundidad; y finalmente *criterion*, el criterio que determina la función para medir la calidad de las divisiones de nodos, donde en este caso se tomará el de 'fiedman_mse' que utiliza el error cuadrático medio, aunque también existen otras opciones como 'poisson', 'squared_error' o 'absolute_error', descartados por generar peores resultados.

Y por último, el modelo de **red neuronal**, los mejores hiper-parámetros de este modelo son apreciables en la Tabla4.6:

Hiper-parámetro	Valor
hidden_layer_sizes	[6,6]
max_iter	10000
activation	logistic
solver	adam

Tabla 4.6: Mejores hiper-parámetros para la red neuronal.

Hidden_layer_sizes especifica el número de neuronas existentes en la capa oculta, que serán [6,6]; *max_iter* se refiere a las iteraciones máximas también llamadas épocas, que se establecen en un número elevado como 10000, dado que incrementarlas no supone ninguna mejora de rendimiento; *activation* para especificar la función de activación de la capa oculta, donde la que mejor resultado obtiene es 'logistic', la función logística sigmoide; y finalmente *solver* para optimizar el peso, como hacíamos en modelos anteriores, donde usaremos 'adam', un optimizador basado en gradiente.

Una vez que hemos comprobado de forma teórica y práctica cuáles son los mejores hiper-parámetros para cada modelo, compararemos los resultados optimizados que generan en la siguiente sección.

4.2.3. Comparativa de resultados entre modelos predictivos

En esta sección compararemos los resultados que obtenemos habiendo determinado cuáles son los mejores hiper-parámetros para cada modelo, los compararemos calculando el porcentaje de acierto en las predicciones, y teniendo en cuenta el valor de los coeficientes de correlación que se describieron en la sección 3.2.2.

Antes de adentrarnos en comparar los modelos, es importante observar cómo cambian los valores de los coeficientes de correlación así como la precisión de los resultados cuando los datos están estandarizados y cuando no lo están, para ello tomaremos un simple ejemplo de un modelo de regresión logística sin hiper-parámetros. Como podemos ver en la Tabla 4.7, los 3 coeficientes tienen un valor considerablemente más alto, que también se refleja en el porcentaje de acierto o precisión del modelo, de ahí la importancia de estandarizar los datos, para facilitar el trabajo a los algoritmos.

	Pearson	Spearman	Kendall	Precision (%)
Datos sin estandarizar	0.155	0.252	0.207	74.33
Datos estandarizados	0.634	0.657	0.562	88.87

Tabla 4.7: Coeficientes de correlación con y sin estandarización de los datos (usando R. Logística).

Bien, ahora para todos los modelos que hemos probado, usando los hiper-parámetros que mejor resultado consiguen, calcularemos sus coeficientes de correlación y los recopilaremos. En la Tabla 4.8, podemos apreciar cómo se comportan los valores de las distintas métricas para todos los modelos optimizados. Los modelos de regresión lineal y logística tienen valores muy similares en Pearson y Spearman, sin embargo en Kendall el modelo de regresión logística es superior, aunque son modelos con coeficientes muy parecidos. Esto mismo ocurre con los modelos de árbol de decisión y red neuronal, donde los coeficientes son prácticamente idénticos en Pearson y Kendall, aunque de valores más bajos que los dos modelos anteriores, y por último el modelo del bosque aleatorio, que tiene el valor más elevado en Kendall, aunque en Spearman es el modelo con peores resultados. Se observa que las mejores métricas son las del modelo de regresión logística.

Modelos optimizados	Pearson	Spearman	Kendall
Regresión lineal	0.716	0.721	0.553
Regresión logística	**0.720**	**0.722**	0.584
Bosque aleatorio	0.678	0.682	**0.590**
Árbol de decisión	0.702	0.675	0.576
Red neuronal	0.696	0.706	0.562

Tabla 4.8: Coeficientes de correlación de los modelos optimizados.

Sin embargo, no podemos usar los coeficientes de correlación como la única vía para comprobar cómo de buenos son los modelos, por ello también calcularemos el error que los modelos obtienen en

el resultado de las predicciones. Para obtener ese error, y por tanto la precisión de todos los modelos, trataremos de predecir con cada modelo la puntuación actual (la más reciente) de los jugadores, y calcularemos el sumatorio de la diferencia del valor obtenido frente al valor real.

Los resultados de la precisión de los modelos pueden verse en la Figura 4.2, donde a pesar de que a grandes rasgos todos tienen una precisión alta, siempre existirá un margen de error. Vemos que el modelo menos preciso de los 5 es el de regresión lineal (86.21 %), el segundo menos preciso es el de la red neuronal (88.68 %), después con una precisión similar están los modelos de bosque aleatorio y árbol de decisión, con un 88.37 % y un 88.51 % respectivamente, y finalmente el modelo más preciso de todos es el de regresión logística (90.13 %).

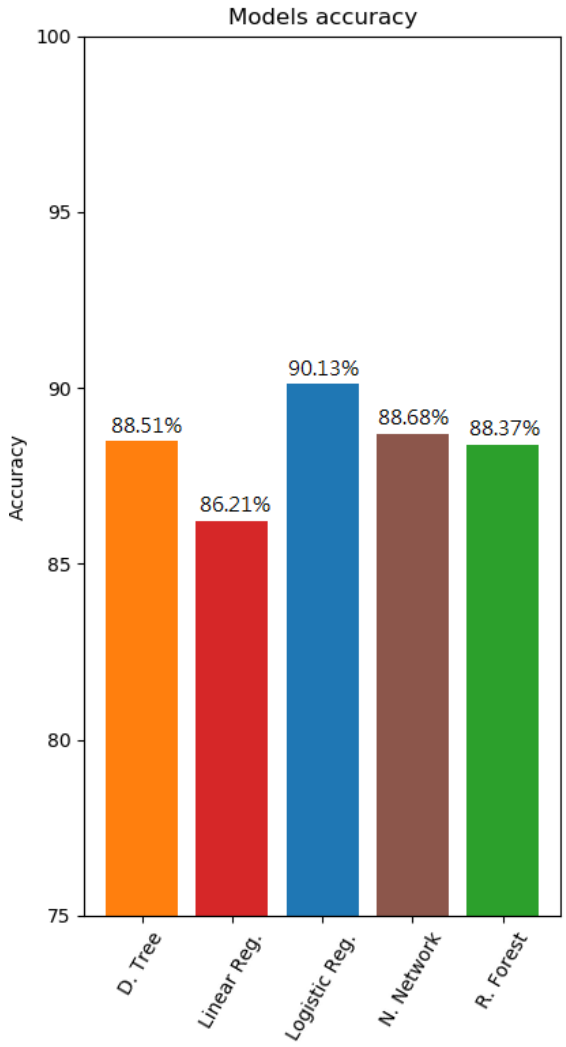

Figura 4.2: Precisión de los modelos optimizados.

Una vez habiendo analizado los coeficientes de correlación y las precisiones de los modelos, hay un factor más a tener en cuenta, y se trata de comprobar que las predicciones se ajustan a la realidad. Dado que son modelos de precisiones parecidas, el objetivo ahora será determinar cuál de todos es el que predice de forma más realista, por ejemplo, no se corresponde con la realidad una puntua-

ción demasiado elevada, o de igual forma una demasiado reducida, debe puntuar un valor que pueda realmente darse.

Analizaremos los resultados de los datos reuniendo en la Tabla 4.9 los valores máximos y mínimos de las predicciones obtenidas, así como la media aritmética y la moda en cada uno de los modelos optimizados. En el caso de la regresión lineal, llega a predecir puntuaciones máximas de valor 36 o 31, esos datos nunca se corresponden con la realidad, ya que en aplicaciones de tipo Comunio no es posible sobrepasar puntuaciones superiores a 25, este es el único modelo al que le ocurre este error con un valor tan elevado, y también falla con valores negativos excesivamente bajos, su media y moda obtienen valores que reflejan que no se obtienen predicciones que correspondan demasiado con la realidad, ya que una media con valores cercanos a -15 es muy poco probable que se dé, y el dato de -49 como valor más repetido termina de confirmárnoslo, al modelo de red neuronal le ocurre algo similar.

Sin embargo, en varios modelos ocurre un patrón parecido, como ya se mencionó con anterioridad, la puntuación -50 corresponde a no haber disputado esa jornada, y las puntuaciones mínimas en aplicaciones de tipo Comunio nunca son inferiores a -10. Pues bien, en los modelos de regresión lineal, bosque aleatorio y red neuronal se dan casos donde ciertos jugadores tienen puntuaciones negativas que oscilan entre -10 y -49, y eso claramente no se corresponde con puntuaciones reales.

Los modelos que más se corresponden con la realidad son el de regresión logística y el de árbol de decisión, ambos modelos predicen valores máximos y mínimos razonables, aunque en el caso del modelo de árbol de decisión hay algunos valores negativos que se salen de la normalidad, pero son muy escasos. Mientras que en el modelo de regresión logística no sucede ninguno de los casos anteriormente mencionados, sus valores máximos y mínimos son muy razonables, así como su media y su moda también lo son, sin duda es el modelo que claramente predice las puntuaciones que más se ajustan a la realidad.

Modelos optimizados	Valor máximo	Valor mínimo	Media	Moda
Regresión lineal	36	-49	-15.81	-49
Regresión logística	17	-8	3.35	2
Bosque aleatorio	14	-49	-11.60	2
Árbol de decisión	17	-15	1.2	1
Red neuronal	20	-49	-17.90	-49

Tabla 4.9: Estadísticas sobre las predicciones de una jornada.

Habiendo analizado todo lo anterior, se llega a la conclusión de que el mejor modelo es el de **regresión logística**, no solo porque es el que mejor acierto genera, ni solo por sus buenos valores de coeficientes de correlación, sino también porque sus predicciones son las que mejor se corresponden con la realidad.

4.2.4. Comparación con herramientas ya existentes

Finalmente, se mostrará el resultado final usando la interfaz de usuario aplicando como algoritmos los mejores modelos (regresión logística y árbol de decisión), y los trataremos de comparar con herramientas ya existentes. Aunque gran parte de estas son código cerrado, compararemos nuestros resultados con la herramienta Automanager, vista en la sección 2.1.2. Lo que haremos será tomar una serie de jugadores al azar, y compararemos los rankings generados.

Los resultados obtenidos son parecidos, en el Apéndice A.1 puede verse el ranking generado por nuestro modelo optimizado con **regresión logística**, y en el Apéndice A.2 figura el ranking obtenido con **Automanager**, donde vemos que el resultado es idéntico. Finalmente, en el Apéndice A.3 se incluye la clasificación generada mediante el modelo optimizado de **árbol de decisión**, donde podemos ver que difiere claramente del anterior.

5

Conclusiones y trabajo futuro

En este último capítulo se comentarán las conclusiones a las que se han llegado durante el transcurso de este proyecto, así como el trabajo futuro que podría realizarse como ampliación.

5.1. Conclusiones

A lo largo de este proyecto, el objetivo ha sido crear una herramienta tecnológica capaz de ayudar a los usuarios de juegos tipo Comunio a lograr ser el mejor gestor deportivo de su liga virtual, y para ello se han llevado a cabo múltiples tareas.

Por una parte, se ha investigado sobre herramientas que se han empleado en la recopilación de los datos (sección 2.2), como la extracción de datos de páginas web mediante técnicas de crawling/scraping que han servido para crear el conjunto de datos del proyecto. Por otra parte se han analizado y aplicado los diferentes algoritmos de predicción (sección 2.3), obtenidos de la librería scikit-learn de Python. Y por último, se ha representado el resultado en una página web usando diferentes tecnologías aprendidas en el grado (sección 2.4).

Habiendo llevado todos estos conocimientos a la práctica y habiendo analizado diferentes modelos de predicción acerca del rendimiento de jugadores de fútbol de la liga española, se ha llegado a la conclusión de que el mejor algoritmo para realizar esta tarea es regresión logística, debido a su alta precisión y a que es el modelo que obtiene las predicciones que más se asemejan a la realidad.

Por lo que concluimos que nuestro modelo de regresión logística es no solo el mejor de todos los analizados, sino un modelo bastante acertado, al ser muy similar a una herramienta disponible para el usuario en el mercado, como se ha visto en la sección 4.2.4.

Finalmente, como conlusión personal, durante la realización de este proyecto he aprendido a combinar en un solo trabajo gran parte de las herramientas tecnológicas que me han enseñado en la Escuela, a pesar de que es un campo donde queda mucho por investigar y realizar, como se comentará en la siguiente sección.

5.2. Trabajo futuro

Durante la realización del proyecto, aparecieron muchas posibles mejoras del mismo, las cuales por falta de tiempo y recursos no se han tratado, aunque podrían figurar como trabajo futuro de este TFG.

En este trabajo se analizan las estadísticas de los jugadores para poder anticiparnos a su rendimiento futuro, pero no se tienen en cuenta factores externos. En este sentido, una posible mejora podría ser: incluir cómo de buenas son las defensas de los equipos rivales para tener en cuenta si supondrá un mayor o menor reto para los delanteros que se están evaluando. O cómo de buena es la delantera de los equipos rivales de cara a evaluar el rendimiento futuro del portero que se está analizando, es decir, tener en cuenta en los algoritmos de predicción las características del rival para poder acercarse más al rendimiento real de los jugadores.

Por otro lado, otra posible ampliación podría ser incrementar la funcionalidad de la interfaz web, creando una herramienta en la que el usuario pueda introducir su plantilla al completo y que los algoritmos generen el mejor sistema de alineación con los mejores jugadores, por ejemplo una formación del tipo 1-4-3-3 si en la plantilla hay buenos delanteros, o una formación 1-5-4-1 si hay mejores defensas y centrocampistas que delanteros, entre muchas otras opciones, de tal forma que se pueda sacar mucho más rendimiento a esta tecnología.

Finalmente, otra posible mejora que resulta interesante sería incorporar este sistema a las propias aplicaciones de tipo Comunio, es decir, que de la misma forma que el usuario puede acceder al mercado de fichajes o a gestionar su plantilla, pueda también acceder desde la propia aplicación a esta herramienta de predicción sin tener que recurrir a una página web externa.

Bibliografía

[1] P. Birnbaum, "A guide to sabermetric research." https://sabr.org/sabermetrics, 2021. Last accessed 1 December 2022.

[2] M. J. Penn and C. A. Donnelly, "Analysis of a double poisson model for predicting football results in euro 2020," *PLOS ONE*, vol. 17, pp. 1–23, 05 2022.

[3] G. Liu and O. Schulte, "Deep reinforcement learning in ice hockey for context-aware player evaluation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden* (J. Lang, ed.), pp. 3442–3448, ijcai.org, 2018.

[4] J. Dong, Q. Huo, and S. Ferrari, "A holistic approach for role inference and action anticipation in human teams," *ACM Trans. Intell. Syst. Technol.*, vol. 13, sep 2022.

[5] D. Peláez, "Revolución 'fantasy' desde málaga: esta ia te dice qué delantero alinear y tiene en cuenta hasta al cronista." https://www.elespanol.com/malaga/economia/tecnologia/20220218/revolucion-fantasy-malaga-ia-delantero-alinear-cronista/650935270_0.html, 2022. Last accessed 4 December 2022.

[6] R. Zambrano, "Scraping, crawling y parsing." https://openwebinars.net/blog/diferencias-entre-scraping-crawling-y-parsing/, 2019. Last accessed 8 December 2022.

[7] Spri, "Sistemas de predicción." https://www.spri.eus/es/teics-comunicacion/sistemas-de-prediccion-machine-learning/, 2014. Last accessed 8 December 2022.

[8] J. M. Heras, "Clasificación y regresión en machine learning." https://www.iartificial.net/clasificacion-o-regresion/, 2020. Last accessed 10 December 2022.

[9] E. Bisong, "Linear regression," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, (Berkeley, CA), pp. 231–241, Apress, 2019.

[10] E. Bisong, "Logistic regression," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, (Berkeley, CA), pp. 243–250, Apress, 2019.

[11] E. Bisong, "Ensemble methods," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, (Berkeley, CA), pp. 269–286, Apress, 2019.

[12] SKlearn, "Standardscaler in scikit-learn." https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2022. Last accessed 19 December 2022.

Apéndices

A

Página web

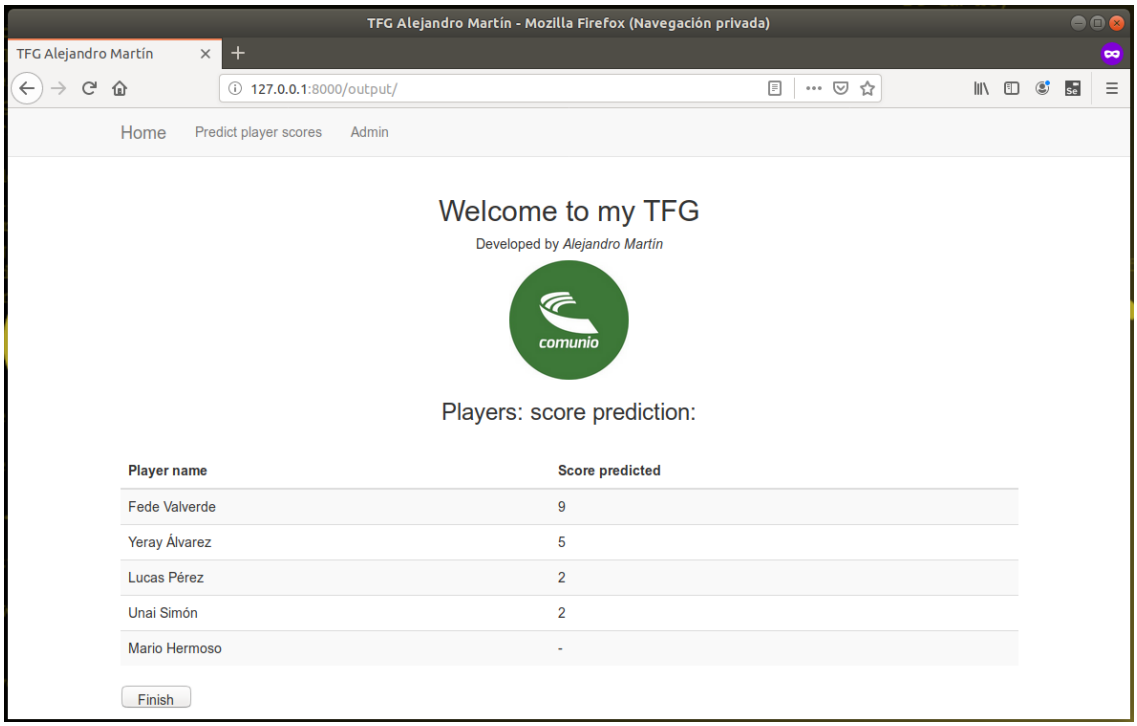

Figura A.1: Resultado ejemplo de ranking generado en la página web (Reg. Logística).

Figura A.2: Resultado ejemplo de ranking generado por “Automanager“.

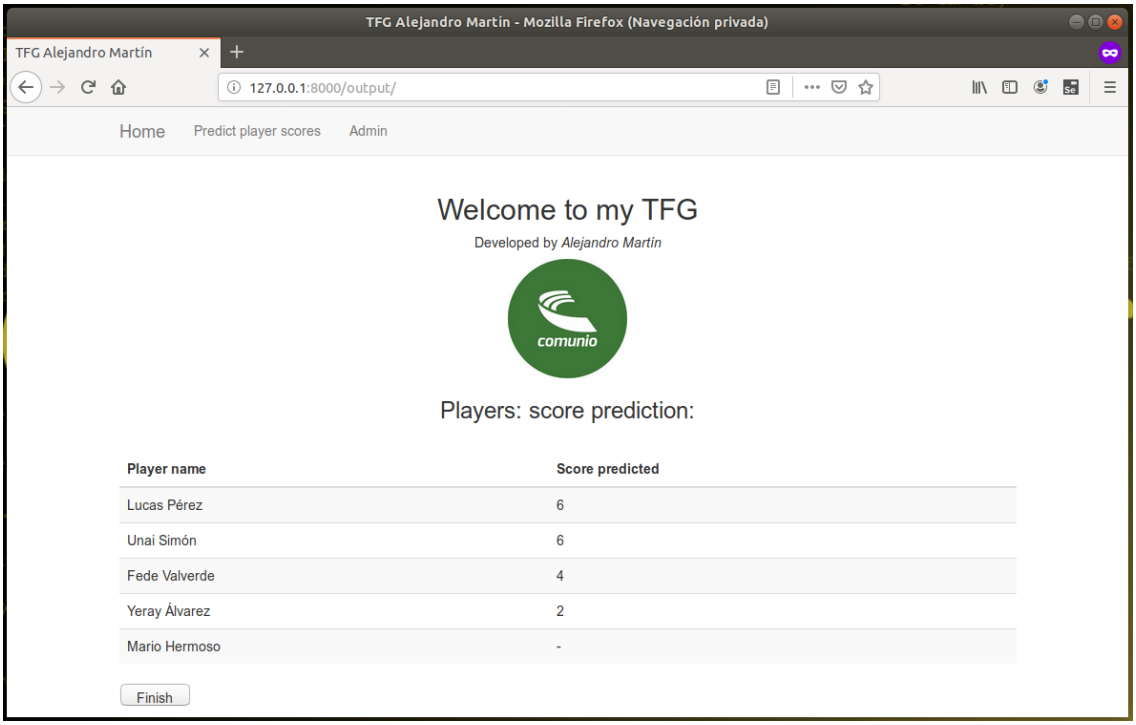

Figura A.3: Resultado ejemplo de ranking generado en la página web (Árbol decisión).


UÁM
Universidad Autónoma
de Madrid