Predicción de cambios en el rendimiento de futbolistas

Autores:

Alberto Gutiérrez Gallego

Daniel Parra Rodríguez

Trabajo de fin de grado del Grado en Ingeniería Informática, Facultad de Informática, Universidad Complutense de Madrid

Director: Rafael Caballero Roldán

Tabla de contenidos

Agradecimientos

Principalmente dar gracias a nuestro director Rafael Caballero Roldán (pese a sus repetidas peticiones de no incluirle), ya que desde nuestro primer curso nos ha acompañado a lo largo de la carrera sirviéndonos de inspiración y abriéndonos un mundo de posibilidades.

Otro gran pilar de nuestro trabajo han sido los miembros de Driblab, debido a que sin su colaboración este proyecto no habría sido posible. Gracias a las constantes reuniones y una comunicación fluida nos permitieron tener claro en todo momento su objetivo.

Agradecer también a la Cátedra de Big Data y analítica HPE-UCM por hacer posible el curso de verano que nos permitió conocer diferentes empresas en el sector del Big Data, entre las cuales se encuentra Driblab, nuestra entidad colaboradora.

Por último y no menos importante dar gracias a nuestras familias por apoyarnos durante estos años de carrera y a lo largo de nuestras vidas.

Resumen

Nuestro proyecto trata de predecir la mejora de rendimiento entre dos temporadas consecutivas de un jugador de fútbol a través del análisis de datos. El principal objetivo, es ser capaces de determinar si el jugador mejorará su rendimiento en al menos un tanto por ciento dado, y tener además alguna indicación de lo probable que resulta que esto suceda. Para esto último, proponemos utilizar métodos de aprendizaje automático como la regresión logística, que indican qué probabilidad hay de que el elemento sea clasificado en una clase u en otra. Esta probabilidad se verá afectada, a la hora de hablar de su fiabilidad, por la precisión del propio modelo, el cual generaremos a partir de un conjunto de datos de jugadores cedido por la empresa Driblab, que se dedica al recuitring de jugadores y que ha mostrado su interés en este proyecto.

El trabajo trata el preprocesamiento de los datos, la obtención de los ficheros con los que entrenamos y probamos el modelo, los clasificadores que hemos utilizado, y los motivos que nos encaminan a tomar ciertas decisiones.

Cabe destacar que, a pesar de la complejidad del problema, los resultados obtenidos en clasificación son buenos y permiten detectar a jugadores que van a incrementar su rendimiento de forma notable la siguiente temporada.

Abstract

Our project tries to predict the improvement of performance between two consecutive seasons of a soccer player through data analysis. The main objective, is to be able to determine if the player will improve his performance in at least a given percentage, and also have some indication of how likely it is that this happens. For this reason, we propose to use machine learning methods such as logistic regression, which indicate the probability that the item will be classified in one class or another. This probability will be affected, when talking about its reliability, by the accuracy of the model itself, which we will generate from a set of player data provided by the company Driblab, which is dedicated to recuitring players and has shown interest in this project.

The paper discusses preprocessing of data, obtaining the files with which we train and test the model, the classifiers we used, and the reasons that lead us to make certain decisions.

It should be noted that despite the complexity of the problem, the results obtained in classification are good and allow detecting players who will increase their performance significantly next season.

Palabras clave

- Tier
- Score
- Posición
- Clasificador
- Modelo
- Dataframe(df)
- Regresión Logística
- Edad
- Driblab

Keywords

- Tier
- Score
- Position
- Classifaer
- Model
- Dataframe(df)
- Logistic Regression
- Age
- Driblab

Capítulo 1

Introducción

1.1. Origen del Proyecto

Tras asistir al curso de verano de Big Data de la Universidad Complutense en el Escorial ("Big Data: La mejora en la toma de decisiones en el mundo del deporte profesional", organizado por la Cátedra de Big Data y analítica HPE-UCM en verano de 2018), el ponente D. Salvador Carmona, de la empresa Driblab, propuso a nuestro mentor intentar predecir el rendimiento de un futbolista en los próximos años y nos comentó que sería un excelente tema para un Trabajo fin de grado.

Con esta idea en mente, decidimos concertar una reunión con dicha empresa para tratar de concretar el alcance y los objetivos de este trabajo. Tras dicha reunión decidimos focalizar nuestros esfuerzos en un caso más concreto del tema inicial, consistente en tratar de predecir el rendimiento de un jugador cuando cambia de liga.
Al inicio nuestro objetivo era predecir el rendimiento de un jugador tras un cambio de liga y de **Tier**[1]. El concepto de la división anterior no quedó lo suficientemente claro, ya que nos dedicamos a identificar aquellos jugadores que cambiaron de equipo y liga, pero sin tener en cuenta el Tier.

1.2. Estado del arte

Al investigar sobre estudios similares al nuestro, encontramos varios trabajos centrados en la predicción de los goles de un jugador en concreto [5], el análisis de datos para predecir los resultados de los encuentros en distintas competiciones [8] y otros tantos estudios que versan acerca del uso del *Machine Learning* para el

[1]Un Tier es un conjunto de ligas. Existen 4 Tiers, desde Tier 1 hasta Tier 4, donde en Tier 1 se encuentran las ligas más importantes, y en el Tier 4 las menos relevantes

análisis de los partidos (en vídeo) para apoyar al entrenador en sus decisiones [10]. En otros deportes como el baloncesto y béisbol [4], el análisis de datos está mucho más presente y arraigado en comparación con el fútbol, puesto en este deporte la analítica de datos es relativamente nueva, ya que todavía se mira con cierto recelo.

Uno de los problemas con el que se han encontrado los pioneros al utilizar estas nuevas tecnologías, es la falta de confianza en dichos resultados, ya que el papel que desempeñan los ojeadores a día de hoy está muy presente y se tiende a pensar que los datos de ambos están enfrentados, cuando en realidad podrían complementarse y apoyarse mutuamente.

La idea sería utilizar los datos obtenidos con estas tecnologías para que sirvan de filtro inicial, y después basarse en la experiencia de los ojeadores para tomar la decisión última, o también se puede comprobar la viabilidad de un posible fichaje que presente un ojeador comparando con la puntuación del objetivo.

Dado que es un proyecto que se puede considerar pionero en España en lo que a fútbol se refiere, vimos una gran oportunidad al poder colaborar con Driblab y por eso decidimos hacer nuestro trabajo fin de grado sobre este tema.

1.3. Objetivos

Nuestros objetivos son los siguientes:

- Poder predecir si un jugador será capaz de mejorar su rendimiento o no en un porcentaje concreto en la siguiente temporada. Para ello nos basaremos en un conjunto de datos que incluye el incremento de rendimiento de numerosos jugadores. La utilización de técnicas de aprendizaje automático supervisado nos permitirá predecir el incremento de rendimiento a partir de la información recopilada para jugadores con rasgos similares, como pueden ser la posición y la edad.

- Una vez completado el objetivo anterior, dependiendo de los resultados la idea sería integrar el modelo dentro de la empresa, mediante un servidor en Node.js que lanzaría nuestro modelo por cada petición realizada desde la página de la empresa Driblab.

1.4. Tecnologías

A lo largo del proyecto se han empleado tecnologías que detallamos a continuación.

1.4.1. PostgresSQL

La base de datos cedida por Driblab se encuentra soportada por PostgresSQL. Para el uso de ésta, en un principio, la empresa dio acceso a nuestro director para así poder descargar y hacer uso de las tablas. Más adelante, se nos facilitó un usuario y contraseña, permitiendo la conexión directa con la base de datos. En una fase final del proyecto se ha integrado nuestra aplicación con el resto de software de la empresa, accediendo directamente a esta base de datos.

1.4.2. Python

A la hora de decidir el lenguaje de programación, se tuvo en cuenta la potencia del mismo, la accesibilidad a los métodos de aprendizaje automático y la compatibilidad con otras tecnologías. Por estos motivos, y tras comentarlo con nuestro director, decimos usar Python como nuestro lenguaje de programación.

1.4.3. Machine Learning

Para hacer posible nuestro proyecto decidimos recurrir al aprendizaje automático [1] (también conocido por el término en inglés *machine learning*). Para entrar en contexto, se entiende por machine learning al método de análisis de datos que automatiza la construcción de modelos analíticos, se trata de una rama de la inteligencia artificial basada en la idea de que los sistemas pueden aprender de datos, identificar patrones y tomar decisiones con mínima intervención humana [6].

Sklearn

Debido al uso del lenguaje Python para la codificación, se optó por el uso de la biblioteca *Sklearn* [7] (Scikit-learn), la cual incluye varios algoritmos de regresión, clasificación y análisis, a la vez que permite interoperar con otras bibliotecas, como por ejemplo la biblioteca de tratamiento numérico *NumPy*.

1.4.4. MongoDB

Debido al volumen de datos con el que estamos tratando y a la necesidad de ordenar éstos para determinados pasos, decidimos aprovechar la potencia de los índices con los que trabaja la base de datos NoSQL MongoDB convirtiendo lo que inicialmente eran casi 4 horas de cómputo a apenas unos minutos.

1.4.5. Node.js

Para hacer posible la correcta implementación de nuestro proyecto en la página de la empresa Driblab se tomó la decisión de levantar un servidor que hiciese de

intermediario entre el modelo escrito en Python y su código JavaScript, para ello decidimos usar Node.js.

Capítulo 2

Introduction

2.1. Origin of project

After the Big Data summer course of the Complutense University in El Escorial ("Big Data: How to improve decision-making in the world of professional sport", summer 2018, organized by the *Cátedra en Big Data y Analítica HPE-UCM*), the first speaker D. Salvador Carmona, from the company Driblab, proposed to our tutor to try to predict the performance of a player in the coming years.

With this purpose, we decided to arrange a meeting with them to try to determine the scope and objectives of this work.

After that meeting, we decided to focus our efforts on a more concrete case of the initial theme, consisting of trying to predict the score of a player when he changes from one league to another.

At the beginning our goal was to predict the player's performance after a change of league and **Tier**[1]. The concept of the previous division was not clear enough, which meant that we went the wrong way by focusing for a while on changing equipment exclusively.

2.2. State of art

In researching similar to our studies, we found several jobs focused on predicting the goals of a particular player [5], data analysis to predict the results of the various competitions [8] and many other studies dealing about providing data on players in

[1] A Tier is a set of leagues. There are 4 Tiers, from Tier 1 to Tier 4, where in Tier 1 are the major leagues, and the Tier 4 less relevant

a game to support the coach in their decisions [10].
If we go to other sports, highlight basketball and baseball [4], the analysis of data is much more present and more ingrained in comparison with football, since in this sport the data analytics is relatively new, and is still viewed with some suspicion.
One of the problems that the pioneers have encountered when using these new technologies is the lack of confidence in these results, since the role played by scouts today is very present and there is a tendency to think that the data of both have to be confronted, when in reality they could be complemented and support each other.

The idea would be to use the data obtained with these technologies to serve as an initial filter and then rely on the experience of the scouts to make the final decision or check the feasibility of a possible transfer that a scout presents compared to the goal score.

Given that it is a project that can be considered novelty in the context of Spanish football, we considered a great opportunity to be able to collaborate with Driblab and we decided to do our final degree work on this subject.

2.3. Goals

Our main objective is to predict if a player will be able to improve his performance or not in a specific percentage in the following season. For this we will rely on a set of data that includes the performance increase of many players. The use of machine learning techniques will allow us to predict the increase in performance from the information collected for players with similar traits, such as position and age.

2.4. Technologies

Throughout the project, the following technologies have been used.

2.4.1. PostgresSQL

The database provided by Driblab is supported by PostgresSQL. For the use of this, in the beginning, the company gave access to our tutor in order to download and be able to use the tables. Later, we were provided with a username and password, allowing direct connection to the database. In a final phase of the project our application has been integrated with the rest of the company's software.

2.4.2. Python

When deciding the programming language, the power of programming,the accessibility to the automatic learning methods and the compatibility with other technologies were taken into account. For these reasons, and with our director, we say using Python as our programming language.

2.4.3. Machine Learning

To make our project possible, we decided to use machine learning [1]. Machine learning is understood as the method of data analysis that self-qualifies the construction of analytical models, it is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention [6].

Sklearn

Due to the use of the Python language for coding, the use of the *Sklearn* [7] library (Scikit-learn) was chosen, which includes several regression, classification and analysis algorithms, while allowing interoperation with other libraries, as for example numerical treatment library *NumPy*.

2.4.4. MongoDB

Due to the volume of data with which we are dealing and the need to order them for certain steps, we decided to take advantage of the power of the indexes with which the MongoDB NoSQL database works, converting what were initially almost 4 hours of computing to only about few minutes

2.4.5. Node.js

To make possible the correct implementation of our project on the company's Driblab page, the decision was made to build a server that would act as an intermediary between the model written in Python and its JavaScript code, so we decided to use Node.js.

Capítulo 3

El conjunto de datos

En este capítulo se describe brevemente los datos de los que se parten.

3.1. Tablas SQL

En el momento de comenzar el proyecto, la empresa utilizaba para su gestión diaria varias decenas de tablas SQL integradas en un sistema gestor de bases de datos relacionales PostgreSQL, con datos de:

- Jugadores. En primer lugar, de entre los datos personales del jugador, nos interesan especialmente:

  1. La *edad* del jugador, que como veremos tiene gran influencia en el rendimiento.
  2. El *identificador* interno del jugador, que nos permite relacionar las diversas tablas.
  3. *Datos globales* del jugador por temporada.

- Partidos. Se tienen datos de cada partido y datos del jugador en cada partido (pases realizados, fallados, con éxito, etc). En nuestro caso no hemos utilizado estos datos porque nos interesan las estadísticas agrupadas por temporada que ya están consideradas en el apartado anterior.

- Equipos: entre las distintas tablas que hacen referencia a los equipos, encontramos información acerca de sus estadísticas, métricas y datos propios de estos. En nuestro caso empleamos dicha información para detectar los cambios de equipo, pero con vistas a futuro, podría ser interesante añadir las características de juego del equipo actual y las del equipo en el que va a jugar, para la mejora de la precisión del algoritmo.

- Temporadas: en este caso los datos que nos facilitan los empleamos principalmente para ubicar un jugador en el tiempo y en un equipo en concretos, a la vez que podemos descartar los datos de las temporadas no regulares, ya que estas no aportan una información real.

  Por ejemplo, en España, podemos diferenciar la liga (temporada regular) de la copa del rey (no regular).

- Ligas: respecto a las ligas, nos interesa principalmente el fichero en el que se identifican cada una de ellas y se les asigna un *tier*.

3.2. Volumen de datos

En total hemos trabajo con datos correspondientes a más de 50 000 jugadores, 1396 equipos que juegan en 154 ligas, en un periodo de tiempo que abarca alrededor de 10 años.

Se trata por tanto de un volumen considerable de datos, pero no tantos como para considerar que estamos en un entorno Big Data, lo que nos ha permitido utilizar tecnologías de aprendizaje automático como la biblioteca de Python para aprendizaje automático *sklearn* que están pensadas para tamaños medios de datos.

Capítulo 4

Preprocesamiento de datos

Para alcanzar nuestros objetivos en este proyecto, la predicción del rendimiento de un jugador y posteriormente la incorporación del modelo predictivo a la aplicación de la empresa, proponemos utilizar técnicas de *aprendizaje automático*. Las bibliotecas usuales requieren que los datos que se van a utilizar para la predicción se encuentren en una estructura prefijada (fichero, dataframe, etc.). Sin embargo, en nuestro caso partimos de unos datos repartidos en multitud de tablas relacionales, por lo que es necesario combinar todos estos datos. Además, los datos no se combinan sin más, hay que realizar numerosas transformaciones para llegar a unos datos de entrada verdaderamente útiles.

4.1. El papel del preprocesamiento de datos

En cualquier proyecto que incluya el análisis de datos, antes de empezar a trabajar es importante realizar un preprocesamiento adecuado, que abarca gran parte de las primeras etapas del proyecto y tiene gran influencia en el resultado final.

Durante esta etapa se toman decisiones que se pueden considerar conocimiento *a priori* que añadimos a los datos. Este tipo de conocimiento es fundamental en el tratamiento de datos, porque como se indica en [9]:

"Roughly speaking, the stronger the prior knowledge (or prior assumptions) that one starts the learning process with, the easier it is to learn from further examples. However, the stronger these prior assumptions are, the less flexible the learning is - it is bound, a priori, by the commitment to these assumptions."

En efecto, decisiones como eliminar ciertas columnas porque "harán más lentos los cómputos y, en la práctica, no guardan relación con el valor que queremos estudiar", pueden impedir que descubramos una relación novedosa e interesante.

Un estudio de la revista Forbes indica que la limpieza y preparación de los datos ocupa entre el 60 y el 80 por ciento del tiempo destinado al análisis de datos. Por

ello, y aunque se consideran estas tareas entre las menos gratificantes de todo el proceso, el preprocesamiento merece ser considerado con atención.

A continuación, describimos el objetivo de esta etapa en el contexto de nuestro proyecto.

4.2. Objetivo

Para afrontar este trabajo y evitar errores irrevocables, decidimos ir generando archivos de datos en función de objetivos muy concretos, permitiendo así volver a una versión anterior de un archivo sin tener que repetir todo el proceso de nuevo.

La obtención del fichero final se explicará detenidamente más adelante.
Tras familiarizarnos con los datos, llegamos a la conclusión de que las entradas en la tabla se diferenciarían entre sí a través del *id_jugador*, la temporada y el equipo conformando así la clave primaria. Gracias a este conjunto podemos diferenciar todos los posibles cambios de un jugador, incluyendo aquellos que se dan en el mercado de invierno.

El objetivo es conseguir un primer fichero en el que cada línea corresponda a un jugador en una temporada y equipo, que recoja la siguiente información:

- Rendimiento.
- Tier en el que ha jugado.
- Edad que tenía esa temporada.
- Estadísticas.

Este fichero posee una gran cantidad de información en bruto, siendo así un buen pilar para el proyecto y será necesario estandarizarlo para así hacer posible la comparación de diferentes métodos.

Con esto conseguido, el siguiente paso sería realizar las parejas entre temporadas consecutivas de un jugador, uniendo las filas con los datos de la temporada siguiente y añadiendo las métricas del mismo.

Con el fichero final terminado, ya podemos empezar a tratar de predecir el incremento de rendimiento.

4.3. Predicción frente a clasificación

La base de datos de la que partimos ya tiene un *score* que mide el rendimiento, que ha obtenido la empresa por sus propios métodos. Nuestro objetivo se centra en intentar predecir este *score* para la próxima temporada. Para eso vamos a utilizar aprendizaje supervisado: entrenamos un modelo con parte del conjunto de datos, incluyendo su *score*, y a continuación intentamos predecir el *score* del resto.

Con los primeros intentos de predecir el *score* de un jugador, utilizando entre otros regresión Lineal, regresión gaussiana y modelos basados en árboles, observamos que el mejor error que se obtiene es de +- 1.6, esto implica que para una predicción de 5 el valor real se encontrará en el intervalo (3.4-6.6) lo cual no nos aporta ninguna información.

Tras la reunión con nuestro director planteamos cambiar la perspectiva con la que estábamos abordando el problema, de este modo cambiamos el ejercicio de predicción en uno de clasificación.

Para ello pusimos un valor (mejora) para el cual aquellas filas en la tabla cuyo incremento de *score* lo superaran pasarían a tener como valor un 1 (mejora) y para aquellas filas en el cual el incremento de *score* fuese inferior se les pondría como valor un 0 (no mejora). De esta forma convertimos el problema en un ejercicio de clasificación binario, en cual identificamos dos clases de rendimiento:

- Mejora $x\,\%$: valor 1
- No mejora $x\,\%$: valor 0

4.3.1. Selección de columnas

Con vistas a empezar ya a trabajar con los datos y obtener las primeras predicciones tratamos de ultimar los retoques.

Las únicas filas que dependen del año siguiente son:

- *inc_score*, calculado como:

$$Incremento\_Porcentaje = \frac{(b-a)*100}{a} \tag{4.1}$$

donde:

a = el score del año correspondiente
b = el score del año siguiente

- *inc_tier*, calculado como

$$Incremento\_Tier = (d - c) \tag{4.2}$$

donde:

c = el tier del año correspondiente
d = tier en el año siguiente

Omitimos las estadísticas del año siguiente, ya que no tiene mucho sentido intentar predecir el rendimiento si tenemos como entrada los datos de la temporada a predecir.

También decidimos eliminar las columnas que hemos usado para navegar entre tablas ya que no nos aportan una información real:
player_id, team_id, after_team_id, season_id, after_season_id, first_name, year, bef_team, aft_team, league_id, aft_league, secondary_position, after_position.

Durante el desarrollo, la empresa nos sugirió incluir los datos de los equipos destino de los jugadores, ya que sería interesante apoyar las estadísticas del jugador con las características del equipo. Para justificar dicha decisión nos basamos en que un jugador ofensivo debería desenvolverse mejor en un equipo que facilite ese rol frente a un equipo que base enteramente su estrategia de juego en una defensa férrea.

Pero debido al pequeño volumen de datos y a la falta de datos de un gran número de equipos referenciados decimos descartar esta posibilidad.

Capítulo 5

Etapas para la obtención del fichero final

Previamente se ha comentado que para alcanzar nuestros objetivos necesitamos obtener un archivo cuya información permitiese aplicar aprendizaje automático. En este capítulo haremos una guía detallada paso a paso de cómo generarlo.

5.1. Paso 1: tablas originales

Para poder generar el archivo con el que entrenaremos y probaremos nuestros modelos, en primer lugar, debemos obtener aquellas tablas que servirán de pilares. En particular, nos interesan las siguientes tablas:

1. Estadísticas del jugador por temporada (tabla *vt_season_stat_player*).
2. Jugadores con sus datos personales actualizados como el equipo actual, edad o peso entre otros (tabla *t_player*).
3. Temporadas con datos informativos básicos de la mismas, permitiéndonos encontrar ente ellos el año y el identificador de liga (tabla *t_season*).
4. Puntuación de cada jugador en las distintas posiciones clasificado por temporada (tabla *vt_season_score_player*).
5. Equipos y datos referentes a los mismos (tabla *t_team*).
6. Ligas, la información referente de si es regular a través de is_regular y clasificación de tier (tabla *t_league*).
7. Métricas de cada jugador por temporada (tabla *vt_season_metric_player*).

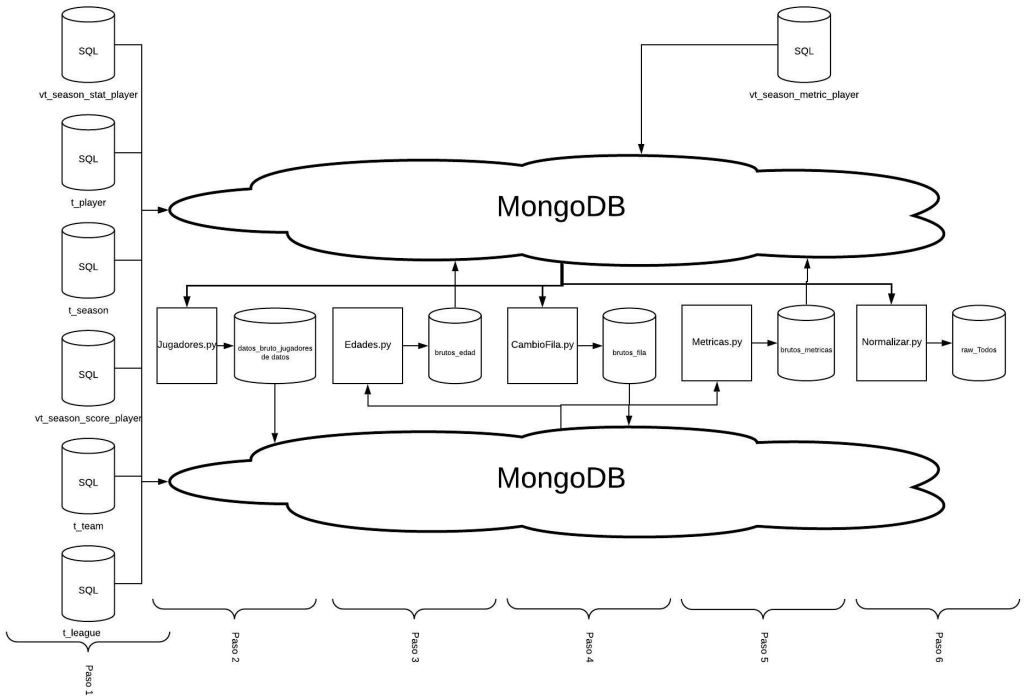


Figura 5.1: Generación del fichero *rawTodos*

Estas tablas se descargaron de la base de datos de Driblab. Para los siguientes pasos, todas estas tablas y las que se vayan generando se cargarán en la base de datos MongoDB, con el fin de aprovechar la optimización de consultas basada en índices, permitiéndonos reducir los tiempos e los programas de preprocesamiento de forma considerable.

En un primer momento, se intentó trabajar sin el apoyo de MongoDB, pero los tiempos de ejecución superaban con facilidad las 3 horas, mientras que tras la implementación utilizando Mongo, este tiempo se redujo a apenas unos pocos minutos.

5.2. Paso 2: jugadores

Entrada: Tablas del 1 al 6 del paso anterior (la tabla 7 se utilizará posteriormente).
Este paso nos permitirá unificar las tablas del 1 al 6 en una sola tabla. Partiendo de las estadísticas de un jugador por temporada (tabla 1) sacamos el identificador del jugador, el del equipo y la temporada. Con estos elementos podremos navegar por las demás tablas sin problema. El siguiente paso es extraer los datos personales del jugador (edad, posición secundaria, nombre,..) de la tabla 2. Seguidamente, con la información de las tablas 3 y 6, añadiremos el año y el tier en el caso de encontrarnos con una liga regular. Para finalizar añadiremos el nombre del equipo y las puntuaciones de las tablas 5 y 4 respectivamente.
Tras dicha unión se obtendrá un archivo que contendrá en cada fila la información personal del jugador en una temporada y equipo concreto, sus estadísticas, tier en el que se encuentra, score (*position_ score*) y Score2 (*position_ predictive_ score*).
Salida: datos_bruto_jugadores

5.3. Paso 3: edades

Entrada: datos_bruto_jugadores
La edad que se tiene es la edad actual. Como ahora tenemos distintas filas correspondientes a las diferentes temporadas en las que ha participado el jugador, tenemos que recalcular la edad para registrar la que tenía en la temporada correspondiente. Un jugador que tiene 28 años en la temporada actual tenía 25 hace 3 temporadas, y así sucesivamente.
Salida: brutos_edad

5.4. Paso 4: cambios de temporada o equipo

Entrada: brutos_edad

El objetivo es generar una fila por cada cambio de temporada/equipo (importante el matiz del equipo, debido al mercado de invierno) de un jugador, con esto obtenemos en cada fila los datos de la temporada|equipo anterior (columnas cuyo nombre incluye el prefijo bef_) y los de la nueva (con aft_), el incremento de score y el incremento de tier.

Para este proceso, ordenamos la tabla de entrada *brutos_edad* por jugador y a continuación por año de la temporada, de forma que al recorrerla estos cambios correspondan a filas consecutivas.

Por ejemplo, 3 temporadas consecutivas de un jugador con identificador *idjug1*:

idjug1	datostemporada1
idjug1	datostemporada2
idjug1	datostemporada3

dan lugar dos filas en la tabla de salida

idjug1	datostemporada1	datostemporada2
idjug1	datostemporada2	datostemporada3

En este punto se puede optar por generar un archivo que tenga como objetivo solo las filas que impliquen un cambio de tier o por el contrario quedarnos con todos los datos, obteniendo así un archivo que nos permite realizar el seguimiento de los jugadores. Para este ejemplo optaremos por este segundo caso, los siguientes pasos son comunes para ambas posibilidades.
Salida: brutos_fila

5.5. Paso 5: métricas

Entrada: brutos_fila
Aunque entre los datos de cada temporada ya se encuentran las estadísticas globales del jugador en esa temporada (pases realizados, etc), incorporamos en este paso a cada fila los datos de la tabla 7 (*vt_season_metric_player*), que corresponde a otros datos del jugador referentes calculados por la empresa para la temporada/equipo anterior.
Salida: brutos_metricas

5.6. Paso 6: normalizar

Entrada: brutos_metricas
Debido a la dependencia de las características de los jugadores y su tiempo jugado, se

decidió realizar una normalización de dichas características por el total de minutos de un jugador en cada temporada correspondiente. Para llevar a cabo dicha operación previamente hemos diferenciado aquellos datos que no permiten la normalización por minutos, como por ejemplo la edad, los porcentajes de las métricas, etc.
Salida: raw_Todos.

Capítulo 6

Predicción de rendimiento mediante aprendizaje automático

Este capítulo presenta los pasos seguidos para la elaboración de los modelos de predicción de incremento de rendimiento a partir de técnicas de aprendizaje automático.

Comenzamos presentando la biblioteca Python utilizada, *Sklearn*.

6.1. ¿Qué es Sklearn?

Sklearn (Scikit-learn) es una biblioteca de aprendizaje máquina de software libre para Python. Dicha biblioteca incluye diversos algoritmos de clasificación, regresión, y análisis de grupos entre los cuales podemos encontrar: regresión logística, Gradient boosting, K-means, etc. [7]. En nuestro caso utilizamos esta biblioteca para intentar predecir el incremento de rendimiento de un jugador en temporadas consecutivas. Algunos de los modelos usados de dicha biblioteca han sido Gradient Boosting, Logistic Regression y árboles de decisión. Estos nos ofrecen diferentes configuraciones y métodos en función del modelo.

Muchos algoritmos de Sklearn requieren que los datos de entrada estén agrupados en un Dataframe. Un DataFrame es una estructura de datos similar a una tabla que facilita el tratamiento de datos.

6.2. Reasignación del volumen de datos

Antes de seleccionar un modelo, hay que preparar los datos, para ello lo primero que se debe realizar es una conversión de los mismos (fichero *raw_ Todos*, generado en el capítulo 5) a un *DataFrame*.

Tras la conversión, debemos eliminar aquellas columnas que no queremos que nuestro modelo tenga en cuenta, es decir, el id del jugador, el id del equipo, etc. ya que no son características que le sirvan al modelo para poder clasificar a un jugador. Una vez eliminadas dichas columnas, se renombra la columna que queremos predecir por el nombre "label".

Llegados a este punto, vamos a dividir el conjunto en dos partes, de diferente tamaño (la información de las tuplas se asigna aleatoriamente en cada ejecución):

- Train: 70 % datos
- Test 30 % datos

Una vez generados ambos conjuntos, es muy importante equilibrar los datos, ya que si tenemos un mayor número de componentes de una clase de rendimiento específica (Mejora/No mejora), el modelo intentará predecir correctamente la clase mayoritaria, ignorando las características principales que identifican a la contraria.

Por este motivo ha sido necesario realizar un procedimiento de equilibrado al 50-50 dentro los conjuntos de entrenamiento y test. Tras equilibrar los datos procedemos a hacer una copia de nuestra columna a predecir:

- train_labels
- test_labels

Posteriormente se eliminará la columna *label* del conjunto *train* y *test*, dado que es lógico no incluir la variable a predecir dentro del modelo.

6.3. Selección del modelo

Sklearn nos ofrece una gran variedad de modelos tanto para predicción como para clasificación.
En nuestro caso se trata de clasificación ya que buscamos predecir si un jugador va a mejorar un tanto por ciento de un año para otro. Para ello probamos con 15 algoritmos, de redes neuronales, basados en árboles y de regresión, resaltando este último grupo del cual podemos destacar dos modelos:

- Logistic Regression: es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes

o predictoras. Es útil para modelar la probabilidad de un evento ocurriendo como función de otros factores. El análisis de regresión logística se enmarca en el conjunto de Modelos Lineales Generalizados (GLM por sus siglas en inglés) que usa como función de enlace la función logit. Las probabilidades que describen el posible resultado de un único ensayo se modelan, como una función de variables explicativas, utilizando una función logística [3].

- Gradient Boosting: construye un modelo aditivo de manera progresiva hacia el escenario buscado, permitiendo la optimización de funciones de pérdida diferenciables arbitrarias. En cada etapa, un árbol de regresión se ajusta al gradiente negativo de la función de pérdida dada [2].

Elegimos dichos modelos debido a los buenos resultados obtenidos tras realizar una serie de pruebas, y a su vez disponen de un método que nos permite ver qué porcentaje se ha asignado en cada caso de clasificación, para un jugador concreto mostrará los porcentajes obtenidos en la clasificación (*probabilidad_ mejora, probabilidad_ no_ mejora*).

6.4. Evaluación del rendimiento

La forma más sencilla de evaluar el rendimiento de clasificadores es mediante el análisis de la matriz de confusión. En la siguiente tabla tenemos dos tipos de clases: positivas y negativas. Una vez completada la matriz es posible extraer una serie de métricas para medir el rendimiento del modelo.

	Positive Prediction	*Negative Prediction*
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

- **Tasa de error (Misclassification Rate)**: porcentaje de datos clasificado incorrectamente

$$TasaDeError = \frac{FP + FN}{Total} \tag{6.1}$$

- **Exactitud (Accuracy)**: porcentaje de datos clasificado correctamente

$$Exactitud = \frac{TP + TN}{Total} \tag{6.2}$$

- **Sensibilidad (Recall)**: es la Tasa de verdaderos positivos, es decir, el porcentaje de datos que logra clasificar de la clase positiva.

$$Sensibilidad = \frac{TP}{TP + FN} \tag{6.3}$$

- **Precisión (Precision)**: es el porcentaje de predicciones correctas que ha hecho.

$$Precision = \frac{TP}{TP + FP} \tag{6.4}$$

6.5. Resultados

Tras realizar varias pruebas con el conjunto de datos, decidimos generar diferentes modelos dependiendo de la posición y la edad, siendo estos resultados algo mejores que los obtenidos con el conjunto de todos los jugadores y por posición. Cabe recordar que en ningún caso se tienen en cuenta los porteros, y que todos los resultados mostrados a continuación parten de un conjunto de datos equilibrado.

Tabla con todo el conjunto de datos.

Logistic Regression

Precisión	Recall
0.79	0.70

Gradient Boosting

Precisión	Recall
0.77	0.77

Tabla para defensas centrales sin tener en cuenta rangos de edad.

Logistic Regression

Precision	Recall
0.76	0.70

Gradient Boosting

Precision	Recall
0.71	0.75

Tabla para defensas centrales (DC) de entre 20 y 40 años.

Logistic Regression

Precision	Recall
0.76	0.70

Gaussian Process Classifier

Precision	Recall
0.73	0.80

Tabla para defensas laterales (DL, DR) de entre 20 y 40 años.

Logistic Regression

Precision	Recall
0.80	0.77

Baggin Classifier

Precision	Recall
0.75	0.80

Tabla para delanteros (FW) de entre 20 y 40 años.

Logistic Regression

Precision	Recall
0.80	0.72

Gaussian Process Classifier

Precision	Recall
0.77	0.78

Capítulo 7

Incorporación del modelo a la dinámica de la empresa

En las últimas reuniones con los miembros de Driblab se trató el tema de añadir nuestro proyecto a su empresa, para ello era necesario que los datos se obtuviesen a través de consultas a la base de datos en lugar de unir archivos descargados previamente.

El objetivo es conseguir que, a través de una petición, los miembros de Driblab pueden obtener información de si un jugador mejorará un 10, 20 o 30 por ciento la temporada siguiente a partir de los datos almacenados en su propia base de datos.

7.1. Conexión del modelo con la base de datos

Para hacer posible la correcta incorporación, nos propusieron crear una nueva tabla en su base de datos que contara con los campos de nuestro archivo final. Para la realización de esta tarea nos pusimos en contacto con el encargado de la base de datos, en la que se detallaba detenidamente cada uno de los elementos necesarios.

A medida que se trabajaba con dicha tabla fueron necesarias varias correcciones debido a pequeños malentendidos, los cuales fueron resueltos satisfactoriamente.

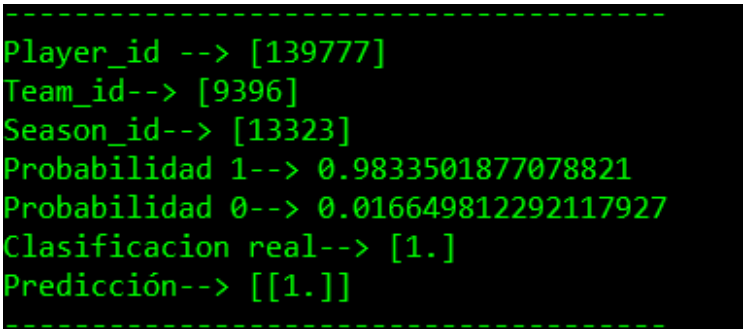
7.2. Creación del servidor

Una vez generada la tabla por parte de los miembros de Driblab, creamos un servidor que se encarga de gestionar las llamadas a nuestro modelo. El objetivo de dicha tarea era ser capaces de comunicar código JavaScript con nuestro modelo escrito en Python y viceversa. Para realizar dicha labor, y tras estudiar las distintas

opciones a nuestra disposición, nos decantamos por usar Node.js. Una vez creado el servidor en Node.js, creamos dentro del mismo un lanzador de consola Python mediante la biblioteca Python-Shell, el cual nos permite lanzar nuestro *bicho.py*.

7.3. Puesta en escena

Una vez realizada una petición desde su página web a nuestro servidor, ya sea de uno o varios jugadores con los datos pertinentes (identificadores: jugador,equipo y temporada), se generará un archivo CSV con dichos parámetros, que posteriormente utilizará el script *bicho.py* para realizar las predicciones. Tras esta tarea, se procederá a devolver un objeto con la predicción, porcentaje con el que el modelo ha tomado la decisión a la hora de determinar si un jugador mejora el umbral o no, y datos referentes al modelo generado por cada uno de los jugadores.

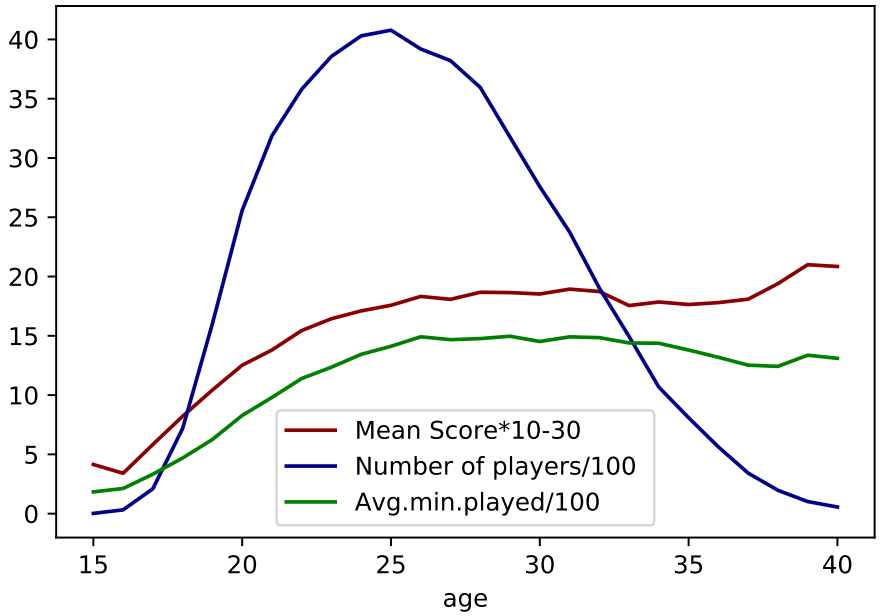


Capítulo 8

Trabajo Futuro

En este capítulo discutimos posibles líneas de trabajo futuro y ampliaciones del proyecto actual.

8.1. Estudio de la relación entre edad y rendimiento



La relación entre el rendimiento de un jugador y su edad, nos ha parecido un tema interesante y significativo, que puede merecer un desarrollo posterior e incluso dar

lugar a otro proyecto.

En efecto, la edad y el rendimiento son dos variables muy relacionadas entre sí y con un gran peso en nuestro proyecto, pudiendo llegar a afirmar que son las más importantes. Observando la gráfica podemos apreciar un comportamiento bastante peculiar. Hay que señalar que la gráfica esta escalada, multiplicando o dividiendo por constantes, para poder comparar la forma de las diferentes representaciones.

En primer lugar, vemos que el volumen máximo de los jugadores lo encontramos a la edad de aproximadamente 25 años y un rendimiento de 4.5 siendo este el punto de inflexión, por la izquierda vemos como los jugadores jóvenes van ganando experiencia lo cual se aprecia por la forma creciente de la línea roja a la vez que encontramos el mayor número de jugadores, y por la derecha podemos apreciar que el score (rendimiento) tiende a normalizarse y en una última instancia a decrecer como sería lógico.

Sin embargo nos encontramos con una irregularidad en el intervalo de 35 - 40, en este punto vemos como el score vuelve a incrementar, esto se debe a que los jugadores realmente "buenos" tardan más en retirarse y al comprobar el número de jugadores (azul) es justo este segmento donde alcanza sus niveles más bajos, dando lugar a los jugadores que se nos presentaron con el sobrenombre de supervivientes y es por ellos, junto con sus altos rendimientos, los que generan esta anomalía en la gráfica y explica ese curioso comportamiento.

8.2. Ranking de mejoras de rendimiento

Actualmente nuestra aplicación indica si un jugador va a mejorar previsiblemente al menos en un tanto por ciento dado. Como información adicional, sería muy útil disponer de un ranking de incrementos de rendimiento para una posición de juego dada. Por ejemplo, saber qué centrales de los que han jugado este año en segunda división van a despuntar más previsiblemente el año próximo. Además, se podría relacionar esta información con el precio del fichaje del jugador, y establecer un ranking ponderado que indique qué jugadores pueden salir más rentables.

Capítulo 9

Conclusiones

Tras una larga etapa de preprocesamiento de los datos, la cual se caracterizó por la búsqueda de las características realmente influyentes en la evolución del rendimiento de un jugador, conseguimos crear un fichero final que incluye las estadísticas y métricas de cada jugador en un par de temporadas consecutivas.

Con estos datos comenzamos a realizar las primeras pruebas de predicción del rendimiento (valor concreto), dando lugar a unos resultados poco esperanzadores y con un margen de mejora prácticamente nulo. Debido a ello, tomamos la decisión de reorientar el proyecto, pasando de un problema de predicción a uno de clasificación, tratando de dar respuesta a la pregunta de si un jugador será capaz de mejorar su rendimiento un tanto por ciento.

Gracias a este cambio de rumbo en el proyecto, se observó una cuantiosa mejora en los modelos, la cual siguió aumentando gracias al estudio de las características de los jugadores mediante un árbol de decisión. A partir de éste, concluimos que la edad y la posición en el campo eran los dos elementos que regirían nuestros modelos, siendo los más representativos al ostentar las ramas más altas del árbol (omitiendo el rendimiento). Contrastando los distintos modelos, nos quedamos con aquel que presentaba los mejores resultados y enfocamos nuestros esfuerzos en maximizar su precisión realizando agrupaciones de edad/posición.

Finalmente adaptamos el modelo a las demandas de los miembros de Driblab, permitiendo desplegar un servidor que gestione las peticiones de la página web, devolviendo el resultado de la predicción y los datos informativos referentes a la misma.

Para finalizar se ha subido la ultima versión de *bicho.py*, junto con los elementos necesarios para desplegar el servidor, *mi.js* (versión de prueba para entorno local,

en desarrollo versión para empresa) y elementos web para simulación de peticiones. Destacar que en *bicho.py* se han eliminado los datos de acceso a la base de datos de la empresa. Dichos archivos se encuentran en este directorio.

Conclusions

After a long stage of data preprocessing, which was characterized by the search of the really influential features in the evolution of a player's performance, we create a final file that includes statistics and metrics for each player in a couple of consecutive seasons.

With this data, we performed the first tests of prediction of the score (i.e. the number measuring the player's performance), giving rise to low quality results and with a margin of improvement practically null. For this reason, we decided to reorient the project, going from a prediction problem to a classification one, trying to answer the question of whether a player will be able to improve their performance a in a given percentage.

With this change in the project, a great improvement was observed in the models, which continued to increase thanks to the study of the features through a decision tree. From this, we concluded that age and position were the two elements that would govern our models, being the most representative when showing the highest branches of the tree (omitting the score). Contrasting the different models, chose the technique that provided the best results, and we focused our efforts on maximizing their accuracy by making age / position groupings.

Finally, we adapt the model to the demands of the Driblab's members, allowing to deploy a server that manages the requests of the web page, returning the result of the prediction and the informative data referring to it.

At the end, the latest version of *bicho.py* has been loaded, along with the necessary elements to implement the server, *mi.js* (test version for the local environment, in the development version for the company) and web elements for the simulation of requests. Note that in *bicho.py* was removed the access to the company's database. These files are in this directory.

Capítulo 10

Contribuciones

En este proyecto ambos integrantes hemos trabajado de forma conjunta a lo largo de todos los apartados. Debido a la naturaleza del mismo, podemos dividirlo en tres etapas bien diferenciadas:

- Preprocesamiento de datos
- Aprendizaje automático
- Incorporación del modelo

A continuación, procedemos a describir nuestras contribuciones:

10.1. Alberto Gutiérrez Gallego

A partir de las primeras reuniones con nuestro director y los miembros de Driblab, mi compañero Daniel y yo debatimos diferentes propuestas en común para iniciar el proyecto.

10.1.1. Preprocesamiento de datos

Una vez planteada la idea, generé una lista con las tablas que contenían la información necesaria, recalcando en cada tabla las características que a nuestro parecer eran las más relevantes y que con posterioridad contrastamos con los miembros de Driblab.

Como nos cedieron las tablas en formato CSV, nuestra primera intención fue trabajar con esta estructura de datos.

A continuación, generé un primer archivo (*datos_bruto_jugadores*) que reunía todas las estadísticas por temporada de un jugador. Sobre éste, se intentó generar un nuevo fichero (*brutos_fila*) que contendría los datos de un jugador por pares de temporada, refiriéndome a pares de temporada como la unión entre datos de un jugador de dos temporadas consecutivas. Al tratar de obtener el segundo fichero, los tiempos de ejecución eran excesivamente altos, llegando incluso hasta las 4 horas, por ello se decidió incluir en MongoDB las tablas necesarias para la optimización de las diferentes consultas a realizar, obteniendo así unos tiempos aceptables gracias al sistema de indexación que ofrece MongoDB.

También generé un script que incorporaba sobre el fichero *bruto_fila* las métricas correspondientes a ese jugador, en la temporada y equipo concreto, generando así el fichero *brutos_metricas*. A partir de aquí mi compañero Daniel se encargó de generar dos archivos, uno de normalización y otro de filtrado de características.

10.1.2. Aprendizaje automático

En esta etapa, gracias a la experiencia obtenida en las prácticas como BigData Developer en la empresa IDOM, generé un notebook en Python (*bicho.py*), el cual contenía principalmente bibliotecas como Numpy, Pandas, Statsmodels y Sklearn.
En este fichero incluí 10 modelos de predicción y 14 clasificadores. Realicé pruebas con todos los modelos sobre el conjunto de datos, sin aplicar ningún filtro sobre los mismos, obteniendo así unos resultados en predicción pésimos (con un rmse nunca inferior a 1) y en clasificación con una precisión de 52 %, teniendo un recall del 60 %. Tras comentar los resultados con mi compañero y director, decidimos centrarnos en los modelos clasificatorios.

Ante la necesidad de realizar filtros sobre el conjunto de datos generé un proceso de filtrado de edades, posición, rendimiento, incremento de score, que permitía realizar conjuntos de datos más específicos.
También incluí una función encargada de eliminar aquellas columnas que eran irrelevantes para el modelo, como nombre del jugador, nombre del equipo, etc.

Posteriormente observé que si se filtraba por posición el modelo presentaba un mejor comportamiento y en combinación con un rango de edades en concreto (aportación de mi compañero Daniel), observamos una gran mejora en los resultados, creando así un modelo específico dependiendo de la posición del jugador y su edad. Para mejorar aún más dichos resultados añadí una parte de equilibrado de datos, donde se reestructura el conjunto para que contenga el mismo número de jugadores de clase positiva y negativa, entiendo por clase positiva a aquellos jugadores que

superan el tanto por ciento de mejora y por clase negativa los que nos superan dicho porcentaje.

10.1.3. Incorporación del modelo

Después de varias reuniones con algunos de los miembros de Driblab, se llegó a la conclusión de que lo más eficiente era generar una tabla con las mismas propiedades que nuestro fichero final y así poder realizar en tiempo real las consultas a su base de datos. Para llevar a cabo dicha tarea modulé el código en diferentes funciones, creando a su vez una función principal que recibiese como entrada una lista de elementos *jugador* (player_id, season_id, team_id), la mejora deseada, y un dataframe con los datos de todos los jugadores obtenidos de la nueva tabla generada.

Cabe destacar también la función *mymodeloPre(...)* dentro de *bicho.py*, la cual permite conocer al mismo tiempo la precisión y Recall obtenida del modelo a partir del conjunto al que pertenece dicho jugador y la probabilidad de que mejore un tanto por ciento.

A continuación, necesitábamos buscar un *intermediario* entre su página web y nuestro código. Para ello mi compañero Daniel y yo buscamos diferentes formas de implementar dicha opción, llegando a la conclusión de que la mejor alternativa era Node.js. Encontramos dos formas:

- Python-Shell: buscando en diferentes repositorios y blogs encontré esta bilbioteca que permite lanzar código Python de manera sencilla, ya que crea un proceso hijo, el cual se encargará de lanzar dicho ejecutable.

- Child-Process: Este método lo encontró mi compañero Daniel, que posteriormente explicará.

Después de una reunión junto a mi compañero decidimos que la forma más sencilla de implementar dicho lanzador era mediante el uso de Python-Shell.
En un primer momento nuestra idea fue pasar una lista de jugadores al código Python por parámetro, pero esto no funcionó, así que decidí generar un DataFrame con los datos de los jugadores solicitados por la petición desde la página web, dicho DataFrame se guarda en un archivo CSV que posteriormente leerá *bicho.py*.

Para hacer posible el uso continuado del recurso, sin tener que relanzar manualmente el proceso, creé un servidor en Node.js, que gestiona las peticiones de la página web de Driblab y a su vez lanza por cada petición el ejecutable *bicho.py*.

10.2. Daniel Parra Rodríguez

En un principio el foco de nuestra atención fue puesto principalmente en la identificación de las tablas que serían relevantes a la hora de caracterizar un jugador.

10.2.1. Preprocesamiento de datos

Tras una puesta en común acerca de los objetivos del proyecto, mientras mi compañero Alberto generaba el primer fichero, debido a la estructura de ciertas tablas me vi en la obligación de realizar manualmente una clasificación de los identificadores de las ligas según el Tier, para ello fue necesario identificar cada una de ellas por sus siglas y gracias a un esquema cedido por Driblab, conseguí crear 4 enumerados representando los 4 Tiers que existen.
Durante este proceso se creó un script (*limpiadorpro.py*) destinado a filtrar las temporadas regulares gracias a la información de la tabla *t_season*, ya que como se ha comentado a lo largo del proyecto, los datos referentes a las temporadas no regulares terminan generando ruido en el modelo.
Analizando las distintas tablas comprobamos que se puede saber si una temporada es regular de dos formas distintas:

- *is_regular_season*: ubicada en la tabla *t_season*.
- *is_regular*: ubicada en la tabla *t_league*.

Con posterioridad, este trabajo se vio sustituido por un campo facilitado en una nueva versión de la tabla en concreto. Por otro lado, se cambió *limpiadorpro.py* por un filtro en el script encargado de generar el archivo *datos_bruto_jugadores* al hacer una comprobación de si la temporada es regular durante la llamada a *t_league*.

Partiendo del fichero bruto_fila creé dos scripts. Uno de ellos tenía como función corregir las edades de los jugadores por año. Por otra parte, el segundo script tenía como objetivo normalizar aquellos datos de una temporada que dependieran de los minutos jugados.

A parte de estos scripts se generó otro más, cuyo objetivo era eliminar ciertas características no normalizables o que eran irrelevantes en el apartado de aprendizaje automático. Dicho script se quedó obsoleto, ya que, tras una reunión con nuestro director, se nos recomendó realizar dicha tarea justo antes de lanzar el modelo, pasando a ser una de las funciones previas en *bicho.py*.

10.2.2. Aprendizaje automático

Gracias a los conocimientos obtenidos por mi compañero en su estancia en IDOM, obtuvimos un fichero con un gran número de modelos tanto para predicción como clasificación. Durante esta etapa me centré principalmente en realizar pruebas sobre los modelos, centrándome especialmente en los modelos de clasificación.

Tras el descubrimiento de mi compañero al ver las mejoras en ciertos modelos al agrupar el conjunto de datos por posición, tuve la idea de añadir a cada conjunto un rango de edades, después de varias pruebas combinando las posiciones y varios rangos de edades descubrimos que ambos parámetros están estrechamente relacionados con el rendimiento.

Al tener tantos modelos, mi compañero me propuso que hiciera una lista con los mejores modelos, buscando una mayor precisión frente al Recall. Después de este proceso llegamos a la conclusión de que los dos mejores clasificadores eran *Gradient Boosting* y *Logistic Regression*.
Una vez elegidos estos dos modelos se realizaron más pruebas para identificar qué rangos exactos se combinaban mejor con determinadas posiciones permitiendo maximizar los resultados.

10.2.3. Incorporación del modelo

Como mi compañero ha comentado en su apartado, realizamos una búsqueda de un intermediario entre la página de Driblab y nuestro *bicho.py*.

En este punto encontré otro método que nos podría permitir lanzar nuestro código *Python*, mediante la biblioteca *child_process* usando spawn. El principal atractivo de este método es la gestión a bajo nivel de diferentes errores, pero su implementación es más elaborada, por lo que se decidió escoger el método que ofrece la biblioteca *Python-Shell*.

Finalizado el proceso de petición, procedí a realizar ciertos cambios en el código *bicho.py* para poder devolver los datos que nos proporcionaban las funciones en un formato para el uso de la página. Para ello procedí a generar un DataFrame con la información obtenida del modelo por cada jugador y posteriormente convertir dicha estructura en un formato JSON (Java-script-Object-notation).

Al tratar de devolver el objeto con toda la información desde *bicho.py* al servidor, nos encontramos con ciertas dificultades a la hora de decodificarlo. Al investigar un poco concluí que la fuente del problema derivaba de la etapa de codificación, en

lugar de en la etapa de decodificación como había supuesto en primera instancia.

Gracias a este sutil cambio, nos fue posible devolver el *JSON* de una manera mucho más sencilla permitiendo así una conversión al objeto *JavaScript* rápida y efectiva.

Bibliografía

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[3] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[4] Michael Lewis. *Moneyball: The art of winning an unfair game*. WW Norton & Company, 2004.

[5] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. quality vs quantity: Improved shot prediction in soccer using strategic features from spatiotemporal data. In *Proc. 8th annual mit sloan sports analytics conference*, pages 1–9, 2014.

[6] Tom Mitchell, Bruce Buchanan, Gerald DeJong, Thomas Dietterich, Paul Rosenbloom, and Alex Waibel. Machine learning. *Annual review of computer science*, 4(1):417–433, 1990.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[8] Havard Rue and Oyvind Salvesen. Prediction and retrospective analysis of soccer matches in a league. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(3):399–418, 2000.

[9] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[10] Hossam M Zawbaa, Nashwa El-Bendary, Aboul Ella Hassanien, and Tai-hoon Kim. Machine learning-based soccer video summarization system. In *International Conference on Multimedia, Computer Graphics, and Broadcasting*, pages 19–28. Springer, 2011.