TRABAJO FIN DE GRADO

Doble Grado en Matemáticas y Estadística

La ciencia de datos como herramienta diferenciadora en el scouting deportivo en el fútbol de élite

José Manuel Maqueda Ruiz

Tutor: Luis Valencia Cabrera

Sevilla, Junio de 2022

Índice general

Prólogo

Este estudio surgió hace más de un año, debido al interés de poder aprovechar las herramientas que nos proporcionan las matemáticas y la estadística para desarrollar una aplicación práctica para el mundo del deporte. Con este trabajo de fin de estudios se puede comprobar cómo se pueden conseguir objetivos con datos públicos y gratuitos, además de comprender la dificultad que conlleva realizar este tipo de proyectos sin la financiación de una entidad que te respalde.

Se ha pretendido mostrar las posibilidades que brindan R y RStudio para el desarrollo de la inteligencia artificial a partir de algunas librerías de paquetes como tidyverse o tidymodels y el uso de R Markdown para Latex y la exportación de documentos PDF.

Se ha creado un proyecto dividido en capítulos donde se desarrolla todo el proceso de búsqueda, importación, transformación y análisis descriptivo de los datos, finalizando con la modelización de los mismos y un apartado de conclusiones.

El objetivo principal del estudio es realizar la mejor predicción posible del valor de mercado de los futbolistas para poder asesorar a las direcciones técnicas de los equipos a la hora de tomar decisiones, como pueden ser la venta o compra de jugadores.

Quiero agradecer a mis padres, a mi hermana y a Rocío por haberme acompañado en esta etapa.

Deseo agradecer especialmente a mi tutor de este Trabajo de fin de Grado, Luis Valencia Cabrera, por su implicación y ayuda constante en la realización del mismo.

Información sobre el autor:

José Manuel Maqueda Ruiz

Estudiante del Doble Grado en Matemáticas y Estadística en la Universidad de Sevilla

Email: josmaqrui@alum.us.es

Resumen

En este documento se explica cómo utilizar algunas herramientas de la ciencia del dato para el *scouting* de futbolistas de élite.

El objetivo principal del estudio es el análisis y predicción del valor de mercado de los futbolistas que se encuentran en activo en la temporada 2021/2022, es decir, realizar un *scouting* de futbolistas y poder ayudar aconsejando a secretarías técnicas de clubes a la hora de tomar decisiones para la confección de la plantilla.

La fuente de datos original, elegida tras un largo proceso de búsqueda, es pública y sus conjuntos de datos se transforman para dar lugar a un conjunto de datos final sobre el que se realizan los análisis descriptivos de las variables. Primero, se describe la variable objetivo *valor de mercado*. Posteriormente, se analiza conjuntamente con las demás variables, para descubrir cuáles son las variables más influyentes en el *valor de mercado* de los futbolistas profesionales. Los análisis incluyen gráficos y tablas para facilitar la compresión de los contenidos.

Además del análisis, la predicción del *valor de mercado* de los futbolistas es uno de los objetivos principales del estudio. Para dicha predicción se usan diferentes modelos de regresión y se obtienen las métricas de cada uno de ellos, para posteriormente elegir cuál de ellos será el modelo final y con el que se realizarán futuras predicciones si esta aplicación se instaurara en el departamento de *scouting* y análisis deportivo de un club.

Para todos los procesos explicados anteriormente se utilizan paquetes de R, sobre todo las librerías *tidyverse* y *tidymodels*.

Como conclusión, se debe destacar la posibilidad de usar este estudio en la práctica real de una secretaría técnica de un equipo de élite y la capacidad de mejora que tiene el proyecto si se tuvieran medios económicos suficientes para ello. Además, se obtienen diferentes estrategias de mercado que deben seguirse en función de si se quiere comprar o vender un futbolista de determinada posición. Por último, se concluye que la predicción realizada para el *valor de mercado* puede usarse como una estimación propia del mismo para la toma de decisiones.

Abstract

This document explains how to use some data science tools for elite soccer players *scouting*.

The main goal of the study is the analysis and prediction of the market value of football players who are active in the 2021/2022 season. That is, we are going to carry out a scouting of football players, so as to be able to help club´s technical staffs make decisions concerning the shaping of their squad.

The original data source, chosen after a long search process, is public and its datasets are transformed in order to reach a final dataset where we make the descriptive analysis of the variables. Firstly, the variable *market value* is described. Subsequently, it is analyzed together with the other variables, to discover which are the most influential variables when it comes to *market value* of professional football players. This analysis include graphics and tables to facilitate the understanding of the contents.

In addition to the analysis of the variables, the prediction of the *market value* of football players is one of the main goals of the study. To do so, different regression models are used and the metrics that we get, are later used to let us discern which of these models will be chosen as the definitive one. Future predictions will be made using this model, if this application is established in a scouting and analysis department of a sports club.

For all of these processes, R packages are used, especially the *tidyverse* and *tidymodels* libraries.

It may be concluded that there are possibilities of using this study in real practice in the technical secretaria of an elite team. It is important, that the project could be improved if we have sufficient funding for it. In addition, different market strategies are obtained if a team wants to buy or sell a football player of a certain position. Finally, we conclude that the prediction made for the market value can be used as an own estimate for making decisions.

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El *scouting* es una palabra de origen anglosajón, que se puede definir cómo el proceso de recogida de información para su posterior análisis. En el futbol de élite se define *scouting* como la recogida de información que posteriormente se transforma en ideas que van a ayudar a identificar talento de jugadores para incorporarlos al equipo, ponerles precio para su venta y estimar un presupuesto para temporadas futuras.

Según la compañía Oracle, una de las más importantes en el mundo del software y las bases de datos, la ciencia de datos es un campo de estudio interrelacionado con la inteligencia artificial que aborda principalmente las áreas interconectadas de la estadística, métodos científicos y análisis de datos, todas ellas utilizadas para extraer significado y conocimientos a partir de los datos.

El estudio va a consistir en utilizar el segundo concepto explicado antes como herramienta para realizar el primero, es decir, se va a utilizar la ciencia del dato para el *scouting* de futbolistas. El objetivo principal de este proyecto es obtener una aplicación que nos proporcione una forma de predecir el *valor de mercado* de los jugadores en función de diferentes parámetros y variables.

El estudio se puede utilizar para calcular una primera estimación del *valor de mercado* de un jugador según nuestro criterio o según el criterio marcado por la secretaría técnica para la que se esté trabajando. Los diferentes criterios dependerán de las variables que esa secretaría quiera utilizar. Si se desea eliminar o introducir variables, estos cambios se podrían implementar en el estudio de manera rápida y eficaz, por lo que hay que destacar que no es un estudio fijo y que puede sufrir las modificaciones necesarias adaptándose a condiciones particulares.

La obtención de una predicción del *valor de mercado*, puede servir de apoyo a la dirección técnica de un club a la hora de tomar diferentes tipos de decisiones.

Uno de los usos más comunes puede ser la de darle un *valor de mercado* a un futbolista que no se encuentre en la base de datos mediante la que se ha realizado el estudio y así poder tasarlo, tanto si está en nuestro club para venderlo o para hacerle una oferta a otro club para ficharlo. Estos jugadores pueden ser aquellos que suben de las categorías inferiores o de los que juegan en ligas extranjeras no registradas en nuestra base de datos.

También, puede servir de apoyo para tener una estimación diferente a la que se conoce públicamente a partir de webs como Transfermarkt, es decir, se consigue un *valor de mercado* propio, lo que provoca una ventaja sobre el competidor.

De esta forma, a la hora de vender un jugador de la plantilla, la tasación del traspaso se realiza con una aplicación propia lo que provoca un mayor beneficio o un negocio más rentable. Lo mismo ocurre cuando se quiere ir al mercado para realizar un fichaje, el club poseedor del futbolista puede pedir una cifra concreta, pero gracias a esta predicción, la dirección técnica sabe el precio máximo que se puede pagar por un jugador para que el negocio sea rentable y así poder llevar a cabo diferentes ofertas por dicho futbolista.

Como tenemos una predicción calculada por nosotros mismos, también puede ser útil para realizar el presupuesto de la plantilla para siguientes temporadas, como por ejemplo, para saber si subirle el sueldo a un futbolista, ofrecerle la renovación o colocarlo en la rampa de salida para una futura venta. Así, se pueden evitar problemas económicos de sueldos estratosféricos para no sobrepasar el límite salarial o con el Fair-Play financiero que muchas ligas han instaurado unos años atrás.

Por supuesto, todo lo contado antes, sirve como apoyo a los clubes, ya que hay muchas circunstacias circunstancias difícilmente, como lesiones o enfermedades, que puedan hacer que un fichaje se trunque y que el jugador cuando esté en la entidad baje su *valor de mercado* "exponencialmente". Aún así, gracias a la predicción y a los análisis descriptivos se pueden definir ciertas estrategias de mercado, tanto a la hora de comprar como de vender jugadores, que se seguirán en función de las variables que describen a los futbolistas, como son la edad, el lugar de nacimiento o la liga que juegan.

Por tanto, hay que destacar que es una herramienta de guía y de apoyo para las secretarías técnicas, no de estricto seguimiento, ya que lo que obtenemos son predricciones.

La idea inicial del proyecto, incluía ver cómo fluctuaban los valores de mercado a lo largo del tiempo, en función de la liga, la edad y otras variables. Sin embargo, esta idea resultó imposible, ya que todos los conjuntos de datos con esta información eran privados (no gratuitos) y no se podían obtener. Por lo tanto, se decidió llevar a cabo la predicción del *valor de mercado* actual (enero de 2022) utilizando bases de datos públicas.

Cabe destacar que, realizando pequeños cambios sobre la aplicación desarrollada en el proyecto, se pueden conseguir otras predicciones del *valor de mercado*. Ese ha sido el problema más importante que se ha tenido durante el estudio, que no teníamos medios económicos para calcular variables que podían ser de interés relacionadas con el rendimiento de los futbolistas.

La mayoría de organizaciones e instituciones que llevan a cabo proyectos de este tipo son aquellas que tienen cantidad de medios económicos y de infraestructuras, lo que es imposible de asumir para un estudio realizado a partir de datos públicos y gratuitos. Por lo tanto, las variaciones de este estudio mediante la implementación de nuevas variables sería interesante si se obtuviera financiación por una entidad con medios económicos y estructurales suficientes.

Para el planteamiento del proyecto, se ha utilizado una fuente de datos (pública y gratuita) de jugadores formada por varios conjuntos de datos. De estos, se han obtenido las variables que se consideran más importantes, todas características de los futbolistas, se han creado nuevas mediante diferentes técnicas y una vez se hayan obtenido se realiza un análisis descriptivo de ellas. Luego, se definen varios modelos predictivos de regresión y se obtienen diferentes métricas. Finalmente, a partir de estas, se decide cuál es el mejor modelo y sobre el que se van a realizar predicciones en el futuro.

Por lo tanto, el estudio se plantea como un primer paso hacía un modelo predictivo

que proporcione información sobre el *valor de mercado* de los jugadores y que puede ser mejorado si se obtuviera financiación de una entidad privada o mediante cualquier otro medio que nos diera acceso a los mismos datos con las que cuentan dichas instituciones deportivas.

El estudio se divide en cuatro capítulos. En el primero de ellos, se define qué es el *scouting* deportivo y el por qué la utilización de la ciencia del dato para su desarrollo. Se cita que el objetivo del trabajo es la estimación de mercado de los futbolistas como herramienta de apoyo a las secretarías técnicas de clubes de élite para la confección de la plantilla.

El segundo capítulo trata sobre las técnicas y las herramientas utilizadas para la realización del estudio. En las técnicas se incorporan definiciones de conceptos que son utilizados en el estudio, como coeficiente de correlación, coeficiente de determinación, la raíz del error cuadrático medio o bagging. Además, se describen las características principales de los gráficos que se usan para analizar las variables. En el ámbito de las herramientas se muestra el uso R y Latex. Dentro de R destaca la importacia de los ecosistemas (entendiendo por tal un conjunto integrado de paquetes que trabajan conjuntamente hacia unos objetivos determinados) *tidyverse* y *tidymodels*.

El tercer capítulo explica el proceso de búsqueda, importación, transformación y descripción de los datos. Es el capítulo más extenso. En primer lugar, se relata la dificultad de encontrar datos públicos sobre el tema del estudio y por qué se toma la fuente de datos elegida para él. Una vez obtenida la fuente de datos, se importan los diferentes conjuntos de datos que la forman. Posteriormente, se realizan transformaciones sobre sus variables, como la eliminación de valores perdidos y se crean nuevas variables a partir de estas, que serán importantes a la hora de analizar y predecir la variable objetivo *valor de mercado*. Una vez se realiza este proceso, se define una tabla de datos, con las variables necesarias, se describen cada una de ellas y se realiza el análisis de la variable objetivo. Posteriormente, se representan las correlaciones entre las variables cuantitativas y se realiza un análisis gráfico de ellas frente a la variable *valor de mercado*. Por último, se analiza si las diferentes clases de las variables nominales afectan al valor de la variable objetivo.

Finalmente, en el cuarto capítulo se lleva a cabo la modelización, es decir, aplicamos diferentes modelos de regresión para predecir el *valor de mercado* de los futbolistas. Los modelos que se desarrollan son los siguientes: árbol de decisión, bosques aleatorios, vecino más cercano, regresión lineal y redes neuronales. Una vez implementados, se toman como métricas el coeficiente de determinación (R^2) y error cuadrático medio. Se concluye que el modelo con mejores métricas es el de bosques aleatorios, con un R^2 muy cercano a 0.9 y una raíz del error cuadrático medio de 3 millones.

Capítulo 2

Preliminares

Una vez introducido el estudio, en este apartado se van a presentar las técnicas y las herramientas utilizadas para su realización. Primero, se describirán las técnicas con las que se ha desarrollado el proyecto y posteriormente las herramientas que han permitido la implementación de dichas técnicas.

2.1. Técnicas

A continuación, se describen las técnicas que se han usado para la importación, tratamiento, análisis y modelización de los datos.

2.1.1. Importación y tratamiento

Los conjuntos de datos importados tienen formato *csv* (formato específico que permite que los datos se guarden en un formato con estructura de tabla, donde las columnas quedan definidas por cada punto y coma, mientras que cada fila se define mediante una línea adicional en el texto), por lo que se utilizaron técnicas para importar este tipo de datos de forma que cada columna tuviera la clase adecuada, en función del tipo de dato de cada variable. Esta misma técnica de importación de datos, se usa para importar un conjunto de datos directamente desde la web.

Una vez importados todos los conjuntos necesarios, para el proceso de limpiado de los datos se usan técnicas como el filtrado y la selección columnas, de forma que sólo se toman las que van a ser útiles para el desarrollo del proyecto. Además, para el tratamiento, se eliminan todas las filas que tengan valores perdidos en variables importantes del estudio y se crean nuevas variables realizando operaciones sobre las propias columnas del conjunto de datos, ya sean operaciones sobre un única variable, involucrando dos o más (por ejemplo, restándolas) o realizando agrupaciones por columnas de variables nominales, en función de las clases de las mismas. Posteriormente, estas nuevas columnas se añaden como nuevas variables a los conjuntos de datos.

También destaca la la realización de joins, que consisten en la unión de dos tablas por una columna donde los valores son iguales y así se obtiene una tabla final con las variables de ambas (si queremos que aparezcan todas las variables de ambas tablas, realizamos *inner_join*, que es utilizada en el estudio). Así, se unen todas las variables útiles para

analizar y predecir el *valor de mercado* de los futbolistas y realizando procesos de limpieza antes descritos obtenemos el conjunto de datos final con el que realizaremos el estudio.

2.1.2. Análisis

El análisis de los datos se realizará a partir de gráficos, resúmenes y coeficientes de correlación. Destaca la utilización de la **transformación logarítmica** sobre la variable objetivo para poder obtener una visualización más clara de los datos.

Realizar una **transformación logarítmica** provoca que los datos sean más sencillos de interpretar a partir de gráficos, ya que la distribución de la variable *valor de mercado* es muy asimétrica, ya que la mayoría de los valores se encuentran por debajo de un millón y muy pocos cercanos a los cien millones. Si se toma la transformación logaritmo, se obtiene una distribución algo más simétrica. Además, las medidas basadas en el orden de los datos, como la mediana o los cuartiles se mantienen iguales cuando se realiza este tipo de transformación, por lo que se obtiene una representación más fácil de interpretar y de la que se pueden obtener las mismas conclusiones que de la representación sin transformar.

La descripción individual de las variables se realizará a partir de cinco gráficos, que proporcionan información de como se distribuye la variables, representando algunas medidas de orden como la mediana y la media:

- **Diagrama de caja y bigotes:** es un método estandarizado para representar gráficamente una serie de datos numéricos a través de sus cuartiles, donde se representan la mediana, los cuartiles y valores atípicos.
- **Diagrama de violín:** se utiliza para visualizar la distribución de los datos y su densidad de probabilidad.
- **Histograma:** representación gráfica en forma de barras, que simboliza la distribución de un conjunto de datos.
- **Estimación de densidad:** visualiza la distribución estimada de datos en un intervalo continuo.
- **Gráfico Q-Q:** se utiliza para comparar las formas de las distribuciones de probabilidad mediante el trazado de sus cuantiles uno contra el otro.

El estudio de la relación entre variables cuantitativas se realiza a partir de un gráfico de la librería *ggplot2*. Este nos proporciona una matriz de correlaciones donde en la diagonal superior aparece el coeficiente de correlación entre las variables, en la diagonal aparece una estimación de la densidad de cada variable y en la inferior una representación de puntos donde se cruzan las variables.

El **coeficiente de correlación** es la medida específica que cuantifica la intensidad de la relación lineal entre dos variables en un análisis de correlación, los valores de este coeficiente van de -1 a 1 y cuanto más se acerca a 0, menor será la relación lineal entre las variables. Si es mayor que cero se denomina correlación positiva y si es menor, correlación negativa.

Para analizar variables cuantitativas con respecto a la variable objetivo, representaremos diagramas de puntos donde se incluye la estimación de la función de densidad. Se llevarán a cabo transformaciones logarítmicas en la variable objetivo y facetados por posición, que envuelven una secuencia unidimensional de paneles en dos dimensiones.

En el caso de las variables cualitativas, se va a mostrar las posibles diferencias de la variable objetivo con respecto a cada clase. Para ello, se van a realizar gráficos como los de las descripciones individuales, pero separando las clases, por ejemplo, si una variable tiene cuatro clases y queremos analizar la variable objetivo mediante un diagrama de violín, se representarán cuatro diagramas de manera independiente.

2.1.3. Modelización

La modelización de los datos se realizará siguiendo diferentes tipos de modelos: árbol de decisión, bosques aleatorios, vecino más cercano, regresión lineal y redes neuronales (se explicarán con exactitud más adelante). Se dividirá la muestra en entrenamiento y test, se utilizará una receta (ingeniería de características), un workflow para trabajar, el entrenamiento se llevará a cabo por validación cruzada y para determinar los hiperparámetros se utilizará el método tuning (consiste en tomar los mejores hiperparámetros según las métricas elegidas).

A continuación, se definen algunos conceptos importantes que serán utilizados el apartado de modelización:

- **Bagging:** se utilizará en los bosques aleatorios y es una agregación Bootstrap que consigue la combinación de diversos modelos, a partir de una familia inicial, lo que disminuye la varianza y evita un sobreajuste.
- **Coeficiente de determinación:** conocido como R^2, refleja la bondad del ajuste de un modelo a la variable que pretender explicar y se define matemáticamente como sigue:

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

siendo n el tamaño muestral.

- **Raíz del error cuadrático medio:** la raíz del error cuadrático medio (RMSE) mide la cantidad de error que hay entre dos conjuntos de datos y se define matemáticamente como sigue:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y})^2},$$

siendo n el tamaño muestral.

R^2 y $RMSE$ son las dos métricas que se van a utilizar para calificar los modelos. Un modelo será bueno, si tiene un valor alto del coeficiente de determinación y un valor bajo de la raíz error cuadrático medio.

2.2. Herramientas

Las dos herramientas principales para la realización y descripción del trabajo son R (en su versión RStudio) y Latex, usándolas de forma conjunta a partir de R-Markdown que permite exportar el documento a PDF.

Los paquetes principales de R utilizados son las siguientes:

- **tidyverse:** es un conjunto de librerías de R diseñadas para la Ciencia del Dato ("Data Science"). Se utilizará para la importación, transformación y análisis descriptivo de los datos. Estas librerías son: *ggplot2*, *dplyr*, *tidyr*, *readr*, *purrr*, *tibble*, *stringr* y *forcats.* Todas comparten la misma filosofía de trabajo, la misma gramática y las mismas estructuras de datos. Las más importantes para nosotros son:
  - **readr** tiene como objetivo proporcionar una forma rápida y sencilla de leer datos en formato 'csv', 'tsv' y 'fwf', entre otros. Se utilizará para importar el conjunto de datos.
  - **tidyr** permite ordenar datos para obtener objetos formato ordenado.
  - **tibble** provee una versión de data frame que facilita el trabajo con el tidyverse. Los tibbles son data frames que facilitan su uso.
  - **dplyr** proporciona una "gramática" para la manipulación y realización de operaciones con data frames. Esto es muy útil, ya que proporciona una abstracción que anteriormente no existía y está formada por funciones muy rápidas.
  - **ggplot2** proporciona una filosofía y una implementación de la gramática de gráficos de Wilkinson, pasando a definir la "layered grammar of graphics", dota a los gráficos de entidad propia como objetos que se pueden manipular, reutilizar, etc. en lugar de presentar una serie de funciones que dibujan en pantalla.
- **tidymodels:** es una colección de paquetes para la modelización y el aprendizaje automático utilizando los principios de tidyverse. Se usará para la creación de modelos con los cuales se van a predecir la variable objetivo *valor de mercado*. De estos paquetes, destacaremos:
  - **rsample** proporciona infraestructura para la división y el remuestreo eficiente de los datos.
  - **parsnip** prueba una gran variedad de modelos sin atascarse.
  - **recipes** es una interfaz para las herramientas de preprocesamiento de datos en la ingeniería de características.
  - **workflows** agrupan el preprocesamiento, el modelado y el posprocesamiento.
  - **tune** ayuda a optimizar los hiperparámetros del modelo.
- **Keras:** es un interfaz de deep learning para python, es decir, Keras es una API para tensorflow, de redes neuronales de alto nivel, que facilita la creación rápida de prototipos de modelos de aprendizaje profundo. Tiene compatibilidad integrada con redes convolucionales, redes recurrentes y cualquier combinación de ambas. También admite arquitecturas de red arbitrarias, por lo tanto, Keras es apropiado para construir desde una red de memoria hasta una máquina neuronal de Turing. Hay que destacar que el acceso a Keras desde R ser realiza a través de *reticulate*.
- **kableExtra:** su objetivo es construir tablas complejas y manipular estilos de tablas de una forma similar a cómo se proporciona estilo a los gráficos por medio de *ggplot2*.

Capítulo 3

Tratamiento y análisis de datos

En este capítulo se va a comentar todo lo relacionado con el proceso de búsqueda, importación, tratamiento y análisis descriptivo de los datos utilizados para el estudio.

3.1. Búsqueda

La búsqueda de datos relacionados con el tema del trabajo fue un proceso arduo, debido a las pocas posibilidades de encontrar fuentes de datos públicas, gratuitas y que tuvieran información adecuada y suficiente para la realización del estudio. Por ejemplo, la página de noticias deportivas *Besoccer* tiene una opción para descargar bases de datos muy completas sobre diferentes aspectos del fútbol (Besoccer Pro), pero al no ser gratuita no pudimos hacer uso de ella y así ocurrió con muchos otros sitios webs especializados en deporte.

Inicialmente, se intentó descargar información de la página web Transfermarkt, que es la que tiene más datos y estadísticas de los futbolistas. Puesto que no se podían descargar directamente de esta web, se tuvo que usar otra alternativa.

También se solicitó información a diferentes foros que tratan sobre este tema, los cuales o no contestaron o negaron la información.

Finalmente y después de todas las negativas explicadas anteriormente, decidimos buscar una base de datos relacionada con el tema a tratar en Kaggle, que es una plataforma web que reúne la comunidad Data Science más grande del mundo, con más de 536 mil miembros activos en 194 países, recibe más de 150 mil publicaciones por mes, que brindan todas las herramientas y recursos más importantes para el máximo rendimiento de la ciencia del dato.

En esta plataforma se encontraron algunas publicaciones interesantes, las cuales fueron analizadas para ver las posibilidades que ofrecía cada una. Tras este proceso, se consiguió un conjunto de distintas tablas de datos relacionadas. Esta fuente de datos se basa en publicaciones de Transfermarkt, por lo que una vez revisados los datos y comprobado que eran datos reales, se decidió tomar para la realización del estudio. También se intentó unir esta fuente de datos con otras que también tenían alguna información interesante, pero al no provenir de Transfermarkt no era posible realizar joins entre ellas con un porcentaje de unión alto.

Esta fuente de datos está formada por seis conjuntos de datos que tratan distintos aspectos que tienen que ver con las estadísticas de los futbolistas, equipos y ligas de catorce países europeos.

Por último, hay que puntualizar que, si hubiéramos tenido posibilidad de acceder a los datos citados al principio, se podría haber realizado también el estudio de cómo varía el *valor de mercado* de los jugadores a lo largo del tiempo, como se ha comentado en la introducción. Como no hemos tenido la oportunidad, ni los medios económicos para obtener esos datos, el estudio se centrará sólo en el *valor de mercado* actual de los jugadores (a enero 2022).

3.2. Importación

Los datos se descargaron en formato *csv*, por lo que serán importados con la función "read_csv" de *tidyverse*, herramienta que, como se ha dicho anteriormente, va a ser usada durante todo el estudio.

Los conjuntos que forman la fuente de datos contienen información desde el año 2014 hasta enero de 2022.

Hay que destacar, que cuando el documento se refiera a "tabla", se está de hablando de un "dataframes" o un "tibble".

A continuación, se muestra cómo se importan los seis conjuntos de datos que se van a usar, junto a una pequeña explicación de lo datos que contienen. Además, se mostrarán las seis primeras filas de cada uno de los conjuntos de datos utilizados.

Se va a incluir el código de la importación del conjunto jugadores, que va a ser el conjunto principal del estudio (los demás conjuntos de datos se importan de forma análoga).

```
jugadores <-read_csv("Datos/jugadores.csv",
  col_types = cols(
    last_season = col_factor(),
    current_club_id = col_integer(),
    country_of_birth = col_factor(),
    country_of_citizenship = col_factor(),
    position = col_factor(),
    sub_position = col_factor()
  ))
```

De este trozo de código sobre la importación, se puede destacar que al conjunto de datos se le fuerza mediante las funciones *col_factor()*, y *col_integer()*, a tomar correctamente el tipo de las variables durante la importación, ya que en caso contrario, se podrían importar como cadenas de caracteres, en un caso, o como números reales, en otro, que no corresponderían con el tipo de dato de las variables.

En este primer archivo (tablas 3.1 y 3.2) se obtienen las variables que caracterizan a los jugadores de nuestra base de datos. Además, de estas variables, en el siguiente apartado se definen algunas nuevas, a partir de los datos que se muestran en las tablas siguientes.

Tabla 3.1: Jugadores (primera parte)

player_id	last_season	current_club_id	name	pretty_name	country_of_birth	country_of_citizenship
45247	2014	1162	damien-perquis	Damien Perquis	France	France
317086	2014	114	enes-fidayeo	Enes Fidayeo	Turkey	Turkey
129067	2018	543	ethan-ebanks-landell	Ethan Ebanks Landell	England	England
269298	2018	4603	samy-frioui	Samy Frioui	Algeria	Algeria
35645	2018	173	kamil-vacek	Kamil Vacek	CSSR	Czech Republic
221873	2018	173	mikkel-desler	Mikkel Desler	Denmark	Denmark

Tabla 3.2: Jugadores (segunda parte)

date_of_birth	position	sub_position	foot	height_in_cm	market_value_in_gbp	highest_market_value_in_gbp
1986-03-08	Goalkeeper	Goalkeeper	Right	186	360000	900000
1997-03-01	Goalkeeper	Goalkeeper	Right	190	23000	45000
1992-12-16	Defender	Centre-Back	Right	188	225000	225000
1991-09-07	Attack	Centre-Forward	NA	180	540000	540000
1987-05-18	Midfield	Central Midfield	Right	184	180000	3150000
1995-02-19	Defender	Right-Back	Right	184	900000	900000

La tabla 3.3 describe información de los **clubes** sobre los que se va a trabajar. Aparecen todos los clubes que han estado en la primera división de los catorce países de los que se tiene información. Por ejemplo, el Córdoba C.F. (llamado Fc Cordoba en el conjunto de datos) bajó a segunda división en 2015 y aparece en la base de datos. El *valor de mercado* de los equipos que no están actualmente en las primeras divisiones es el que tenía cuando descendió de estas categorías, aunque esta variable no se va a usar en el proyecto.

Tabla 3.3: Clubs

club_id	name	pretty_name	domestic_competition_id	total_market_value
3302	ud-almeria	Ud Almeria	ES1	51.66
5572	niki-volou	Niki Volou	GR1	3.40
20698	balikesirspor	Balikesirspor	TR1	1.58
1429	cesena-fc	Cesena Fc	IT1	6.82
5220	gs-ergotelis	Gs Ergotelis	GR1	3.96
993	fc-cordoba	Fc Cordoba	ES1	2.99

En el conjunto de datos **competiciones** (tabla 3.4) se presenta información sobre las diferentes torneos y sus partidos. De todos los que hay, sólo vamos a utilizar las competiciones domésticas para crear variables nuevas que se explicarán a continuación.

Tabla 3.4: Competiciones

competition_id	name	type	country_id	country_name	domestic_league_code	confederation
L1	Bundesliga	First Tier	40	Germany	L1	Europa
DFB	Dfb Pokal	Domestic Cup	40	Germany	L1	Europa
DFL	Dfl Supercup	Domestic Super Cup	40	Germany	L1	Europa
NL1	Eredivisie	First Tier	122	Netherlands	NL1	Europa
NLP	Toto Knvb Beker	Domestic Cup	122	Netherlands	NL1	Europa
NLSC	Johan Cruijff Schaal	Domestic Super Cup	122	Netherlands	NL1	Europa

La tabla 3.5 muestra información sobre las **Ligas** de cada país. Se observa que hay catorce ligas de diferentes países, por lo que los partidos sobre los que se van a realizar estudios van a ser de las competiciones domésticas de estos catorce países, es decir, no se van a tener en cuenta competiciones europeas. Esta exclusión, es debida a que queremos

centrarnos en los datos disponibles y uniformes para todos los jugadores (no todos los jugadores juegan competiciones europeas), aunque podría tener sentido incorporarlas a los modelos más prometedores que se tengan, lo que sería un método para mejorar la aplicación.

Tabla 3.5: Ligas

league_id	name	confederation
L1	Bundesliga	Europa
NL1	Eredivisie	Europa
BE1	Jupiler Pro League	Europa
ES1	Laliga	Europa
PO1	Liga Nos	Europa
FR1	Ligue 1	Europa
GB1	Premier League	Europa
RU1	Premier Liga	Europa
UKR1	Premier Liga	Europa
SC1	Scottish Premiership	Europa
IT1	Serie A	Europa
GR1	Super League 1	Europa
TR1	Super Lig	Europa
DK1	Superligaen	Europa

La tabla 3.6 incluye estadísticas de las **apariciones** de los jugadores en los partidos que ha participado desde el año 2014 hasta enero de 2022. En esta tabla está parte de la información más relevante sobre los jugadores, incluyendo goles, asistencias, minutos jugados, tarjetas amarillas y rojas, aunque faltaría mucha información que sería relevante pero no teníamos disponible sobre tackles, despejes, etc.

Tabla 3.6: Apariciones

player_id	game_id	appearance_id	competition_id	player_club_id	goals	assists	minutes_played	yellow_cards	red_cards
52453	2483937	2483937_52453	RU1	28095	0	0	90	0	0
67064	2479929	2479929_67064	RU1	28095	0	0	90	0	0
67064	2483937	2483937_67064	RU1	28095	0	0	90	0	0
67064	2484582	2484582_67064	RU1	28095	0	0	55	0	0
67064	2485965	2485965_67064	RU1	28095	0	0	90	0	0
67064	2487345	2487345_67064	RU1	28095	0	0	90	1	0

Por último, en la tabla 3.7 se muestra información sobre cada uno de los **partidos** que se han jugado en las competiciones anteriores durante los años citados.

Tabla 3.7: Partidos

game_id	competition_code	season	round	date	home_club_id	away_club_id	home_club_goals	away_club_goals
2459774	NL1	2014	21. Matchday	2015-02-05	610	1090	0	1
2480169	PO1	2014	16. Matchday	2015-01-11	2425	1301	0	0
2490887	PO1	2014	20. Matchday	2015-02-08	1085	2990	0	0
2498723	PO1	2014	22. Matchday	2015-02-21	1085	3327	0	1

Por último, se importó desde la web un archivo donde aparece a qué continente pertenece cada país. Esta variable se asociará a los futbolistas y se usará para la predicción y análisis del *valor de mercado*.

```
continent <- read_csv(
 "https://raw.githubusercontent.com/dbouquin/IS_608/
                  master/NanosatDB_munging/Countries-Continents.csv",
                     col_names = c("continent","country_of_citizenship"))
```

Tabla 3.8: Continente

continent	country_of_citizenship
Africa	Cameroon
Asia	Laos
Europe	Monaco
North America	Dominican Republic

Para obtener más información sobre el código de la importación de los datos, lea la sección A.1 del apéndice A.

3.3. Tratamiento

Una vez se han importado los datos, se utilizan las variables de las tablas anteriores para definir otras nuevas, que puedan ser útiles para la predicción y el análisis del *valor de mercado* de los futbolistas.

Antes de definir estas variables, se crea una tabla que va a ser con la que se realizarán los modelos posteriormente. Los jugadores que aparecen en esta tabla son aquellos que están actualmente en activo y no tienen valores perdidos en ninguna de sus variables. Dichos futbolistas son aquellos cuya última temporada es la actual, es decir la 2021/2022. En la base de datos, las temporadas se nombran por el año de inicio de las mismas. Por lo tanto, se considera que un futbolista está activo si su última temporada (*last_season*) es igual a 2021.

Además, en esta tabla sólo se incluyen las variables necesarias para el estudio.

A continuación se muestra el código donde se crea este conjunto de datos:

```
jugadores_con_valor <- jugadores %>%
  select( pretty_name,player_id,current_club_id,last_season,
          country_of_birth, country_of_citizenship, position,
          sub_position,foot, height_in_cm,market_value_in_gbp,
          highest_market_value_in_gbp,age) %>%
  drop_na(market_value_in_gbp) %>%
  drop_na(age) %>%
  drop_na(country_of_birth) %>%
  filter(last_season == 2021) %>%
  filter(height_in_cm > 0) %>%
  filter(last_season == 2021) %>%
  filter(position != 0)
```

Se observa, que se están llevando a cabo labores de limpieza, eliminando observaciones con valores no disponibles con "drop_na", o filtrando por aquellos registros cuya altura tiene sentido o cuya posición está informada.

Las nuevas variables que se crean son las siguientes:

- **age:** creada restando el 2021 al año de nacimiento de los futbolistas (variable edad).
- **continent:** se realiza un *inner_join* entre las tablas "jugadores_con_valor" y "continent" (para ver código ir a la sección A.2.2 del apéndice A).

```
jugadores_con_valor <- jugadores_con_valor %>%
  inner_join(continent)
```

- A partir de las tablas 3.6 y 3.7 (a las que unimos mediante *inner_join*) vamos a obtener cuatro variables nuevas que son **goals_average** (media de Goles), **assists_average** (media de Asistencias), **min_played_average** (media de Min.Jug), **yellow_cards_average** (media de amarillas) y **red_cards_average** (media de rojas). Para ello, se agrupan los datos por los jugadores y se realiza la media por temporada, todo ello con funciones de *tidyverse* (ver código en la sección A.2.3 del apéndice A para más información). Hay que destacar, que se toman sólo los valores de años posteriores a 2017, debido a que se ha decidido que los valores de años anteriores no influyen en el *valor de mercado* actual de forma significativa, ya que la correlación de estas variables antes de 2017 es mucho más baja, la mitad o menos en todos los casos, que la de 2017 en adelante. Así, también se tiene en cuenta la trayectoria del jugador, pero no de manera exagerada, ya que sólo se tienen en cuenta las cinco temporadas anteriores.

```
estats_desde_2017 <- jugadores_con_valor %>%
  inner_join(aparaciones) %>%
  inner_join(partidos) %>%
  group_by(player_id,season,pretty_name) %>%
  summarise(Goles = sum(goals),
            Asistencias = sum(assists),
            `Minutos Jugados` = sum(minutes_played),
            `Tarjetas amarillas` = sum(yellow_cards),
            `Tarjetas rojas` = sum(red_cards)) %>%
  filter(season > 2017) %>%
  inner_join(jugadores_con_valor[,c(2,11)])

statisticss_jugadores <- estats_desde_2017 %>%
  group_by(player_id,pretty_name) %>%
  summarise(goals_average = mean(Goles),
            assists_average = mean(Asistencias),
            min_played_average = mean(`Minutos Jugados`),
            yellow_cards_average = mean(`Tarjetas amarillas`),
            red_cards_average = mean(`Tarjetas rojas`),
            market_value_in_gbp = mean(market_value_in_gbp))
```

- **Statistics:** es una variable de elaboración propia, relacionada con el equipo al que actualmente pertenece el jugador. Consiste en una suma entre dos valores, a la que se le multiplica un índice de valor de cada liga. Este índice, que proporciona un valor a cada liga en función de su nivel, es utilizado para la clasificación de la bota de oro, con el objetivo de darle más valor a los goles de las ligas más competitivas. En este estudio se usa con el mismo significado. Los valores que se suman son los siguientes:

- SumaPuntos: consiste en la suma de los puntos que ha conseguido cada equipo desde 2014, sumando 0 los años en los que el equipo no haya disputado la primera división del país al que pertenezca.
- SumaClasificacion: se trata de sumar la posición inversa que ha conseguido cada equipo desde 2014, sumando 0 los años en los que el equipo no haya disputado la primera división del país al que pertenezca.
- Estas dos variables anteriores, se sumarán también centradas y escaladas (media cero y varianza uno) definiendo una variable denominada **statistics.scale** que será la que realmente se va a usar para el estudio.

Si se desea ver el código completo de la creación de esta variable ir al apartado A.2.4 del apéndice A.

- **goals_conceded_season_average (media de goles encajados):** a partir de la tabla "partidos" se toman aquellos disputados desde 2017 (siguiendo el criterio usado anteriormente) y en los que los jugadores implicados hayan jugado más de 45 minutos (si juega menos, no se cuenta como partido jugado). Luego, se agrupan por jugador y temporada y se suman los goles encajados. A partir de esta variable se obtiene **goals_conceded_minute (media de goles encajados por minuto)** dividiendo la anterior por los minutos totales jugados en estas temporadas (ver código completo en la sección A.2.5 del apéndice A).

Hay que destacar, como se ha citado antes, que estas nuevas variables son creadas por medio de fuentes de datos gratuitas y que con estos mismos procedimientos y otros conjuntos de datos más completos (que son privados) se podrían crear variables muy interesantes que mejorarán los resultados

Finalmente, tenemos el siguiente conjunto de datos (tabla 3.9, 3.10 y 3.11) sobre el que se va a realizar el análisis descriptivo y la modelización:

Tabla 3.9: Jugadores con valor (primera parte)

player_id	pretty_name.x	market_value_in_gbp	goals_average	assists_average	min_played_average	yellow_cards_average
3333	James Milner	1800000	3.00	3.750000	1718.750	6.500000
3455	Zlatan Ibrahimovic	3600000	12.00	3.333333	1503.667	3.333333
4188	Ricardo Quaresma	270000	3.25	5.250000	1618.500	4.750000
4311	Maarten Stekelenburg	495000	0.00	0.000000	780.000	0.000000
4391	Boy Waterman	135000	0.00	0.000000	1183.000	0.500000
5336	Anastasios Tsokanis	315000	0.50	0.250000	1640.500	4.250000

Tabla 3.10: Jugadores con valor (segunda parte)

red_cards_average	position	sub_position	foot	height_in_cm	country_of_birth	highest_market_value_in_gbp
0.0000000	Midfield	Central Midfield	Right	175	England	18900000
0.3333333	Attack	Centre-Forward	Both	195	Sweden	41400000
0.5000000	Attack	Right Winger	Right	175	Portugal	22500000
0.0000000	Goalkeeper	Goalkeeper	Right	197	Netherlands	9900000
0.5000000	Goalkeeper	Goalkeeper	Right	188	Netherlands	2700000
0.0000000	Midfield	Defensive Midfield	Left	176	Greece	405000

Tabla 3.11: Jugadores con valor (tercera parte)

age	statistics	statistics_scale	goals_conceded_season_average	goals_conceded_minute	name	continent
35	1314.0	6.583268	20.50000	0.0119273	Premier League	Europe
40	1108.0	4.653212	23.33333	0.0155176	Serie A	Europe
38	643.5	1.611510	23.25000	0.0143652	Liga Portugal Bwin	Europe
39	997.5	5.107919	6.50000	0.0083333	Eredivisie	Europe

Tabla 3.11: Jugadores con valor (tercera parte) (continúa)

age	statistics	statistics_scale	goals_conceded_season_average	goals_conceded_minute	name	continent
37	187.5	-3.440089	20.50000	0.0173288	Super League 1	Europe
30	106.5	-4.158338	26.00000	0.0158488	Super League 1	Europe

3.4. Análisis

Una vez obtenido el conjunto de datos final con las variables necesarias para el estudio, en este apartado se realiza el análisis descriptivo de las variables que van a ser útiles para predecir el *valor de mercado* de los futbolistas.

3.4.1. Descripción de las variables predictoras

En primer lugar se incluye una descripción sobra las variables predictoras para saber en qué consisten, cuales son sus unidades de medida, qué significan, etc.

- **Media de Goles (goals_average):** variable cuantitativa que se corresponde a la media de los goles por temporada marcados por los futbolistas desde el año 2017.
- **Media de Asistencias (assists_average):** variable cuantitativa que se corresponde a la media de los asistencias dadas por temporada por los futbolistas desde el año 2017.
- **Media de Minutos Jugados (min_played_average):** variable cuantitativa que se corresponde a la media de los minutos por temporada jugados por los futbolistas desde el año 2017.
- **Media de amarillas (yellow_cards_average):** variable cuantitativa que se corresponde a la media de tarjetas amarillas por temporada recibidas por los futbolistas desde el año 2017.
- **Media de rojas (red_cards_average):** variable cuantitativa que se corresponde a la media de tarjetas rojas por temporada recibidas por los futbolistas desde el año 2017.
- **Posición (position):** variable cualitativa que indica el lugar del campo de juego donde suele jugar cada futbolista. Sus valores posibles son Goalkeeper (portero), Defender (defensa), Midfield (cetrocampista) y Attack (delantero).
- **Pie (foot):** variable cualitativa que indica los tipos de lateralidad del pie de los jugadores. Sus valores posibles son Right (diestro), Left (zurdo) y Both (ambidiestro).
- **Altura (height_in_cm):** variable cuantitativa que indica la altura medida en centímetros.
- **Mayor valor de mercado (highest_market_value_in_gbp):** variable cuantitativa que proporciona el valor de mercado más alto (según Transfermarkt), en libras, que ha tenido el futbolista en su carrera.
- **Edad (age):** Variable cuantitativa que indica la edad en años de los jugadores.
- **Estadística (statistics_scale):** variable cuantitativa de creación propia explicada en el apartado anterior. Sólo se incluye la que realiza el escalamiento de las variables antes de la suma. Esta variable está relacionada con las ligas, es decir, que todos los

jugadores de la misma liga tendrán el mismo valor en esta variable. Otra opción para ver si las ligas influyen en el *valor de mercado* hubiera sido introducirla como variable predictora para que posteriormente en el modelado se definiera como variable dummy.

- **Goles encajados por minuto (goals_conceded_minute):** variable cuantitativa que se corresponde a los goles que le han marcado a los futbolistas por minuto desde el año 2017. Esta variable se crea debido a que cuantos más minutos se juegan, más goles se reciben (se verá en la correlación entre ambas variables), por lo tanto, es interesante observar cuantos goles recibe cada jugador por minuto.
- **Continente (continent):** variable cualitativa que proporciona el continente de nacimiento de cada jugador (Africa, Asia, Europe, North America, Oceania y South America).

3.4.2. Descripción de las variable objetivo

La variable objetivo es **valor de mercado (market_value_in_gbp)**, que define el valor de mercado actual (a enero 2022) de los futbolistas en activo y se mide en libras, moneda oficial de Reino Unido.

A continuación se representan el diagrama de cajas y bigotes, el diagrama de violín, el histograma, la estimación de densidad y el gráfico Q-Q de la variable objetivo.

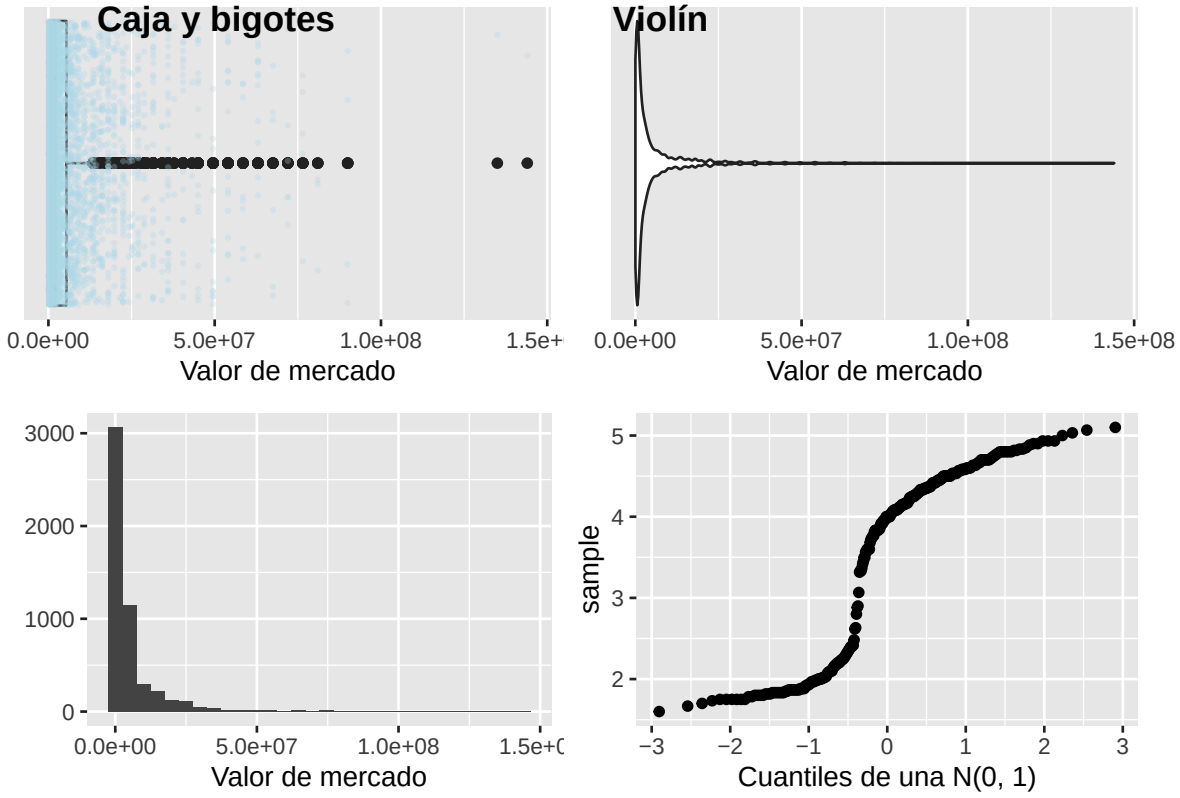


Figura 3.1: Diagramas valor de mercado

En los gráficos incluidos en la figura 3.1 se observa que la mayoría de los valores de mercado son bajos (por debajo de 2 millones de libras), como puede verse en el diagrama

de violín, donde la mayor amplitud está en los valores más bajos. En el histograma, se observa que por debajo de los 10 millones hay más de 3000 valores.

El diagrama de cajas y bigotes considera valores que pueden ser outliers a los jugadores con un *valor de mercado* mayor de 10 millones. Aunque en algunos manuales de análisis de datos se recomienda la eliminación de estos valores tan alejados de la media, para nosotros pueden ser importantes ya que el *valor de mercado* de estos jugadores suele ser muy relevantes para equipos con mucho poder económico ya que suelen fijarse en los futbolistas más caros, famosos e importantes. Por lo tanto, para poder aconsejar a este tipo de entidades si nos lo solicitan no vamos a eliminar dichos valores outliers.

Al haber valores muy altos, la diferencia entre estos y los más bajos son muy grandes, por lo que las representaciones anteriores no proporcionan una información muy visual. Para mejorar estos gráficos, a continuación, en la figura 3.2 se representará el *valor de mercado* aplicándole la transformación logarítmica explicada en la introducción. En este caso, en vez de mostrar el histograma, se va a mostrar el gráfico de densidad.

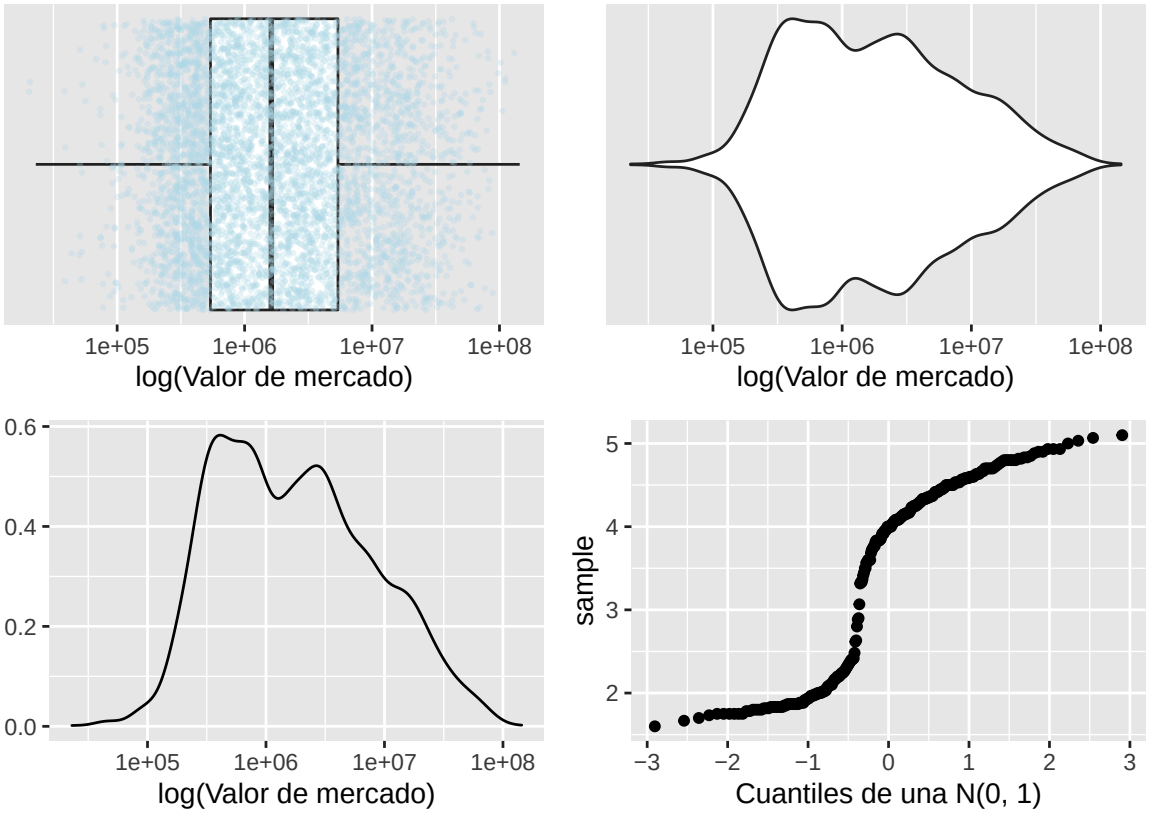


Figura 3.2: Diagramas valor de mercado transformada

Una vez realizada la transformación, la distribución del *valor de mercado* es más simétrica y podemos observar en los tres primeros gráficos que la mayoría de los valores son muy bajos, por ejemplo, la zona más oscura en el diagrama de cajas y bigotes está por debajo de la línea de la mediana y casi al final de la caja. Las conclusiones son similares a los gráficos anteriores, pero estos nos dan una visión más clara de cómo está distribuido el *valor de mercado* de los futbolistas.

3.4.3. Correlaciones

Hay doce variables cuantitativas, once de ellas predictoras (para la modelización sólo serán 10, ya que los goles medios recibidos no se utilizan para la predicción de la variable objetivo *valor de mercado*, ya que tiene correlación positiva, porque los goles encajados tiene mucha relación con lo minutos jugados) y son muchas para que a partir de la función *ggpairs* de *ggplot* se muestre un gráfico de correlaciones adecuado. Como se pretende obtener la correlación entre las variables predictoras y la variable objetivo, se realizan dos gráficos con las correlaciones y en ambos se va a incluir a la variable *valor de mercado* (figura 3.3).

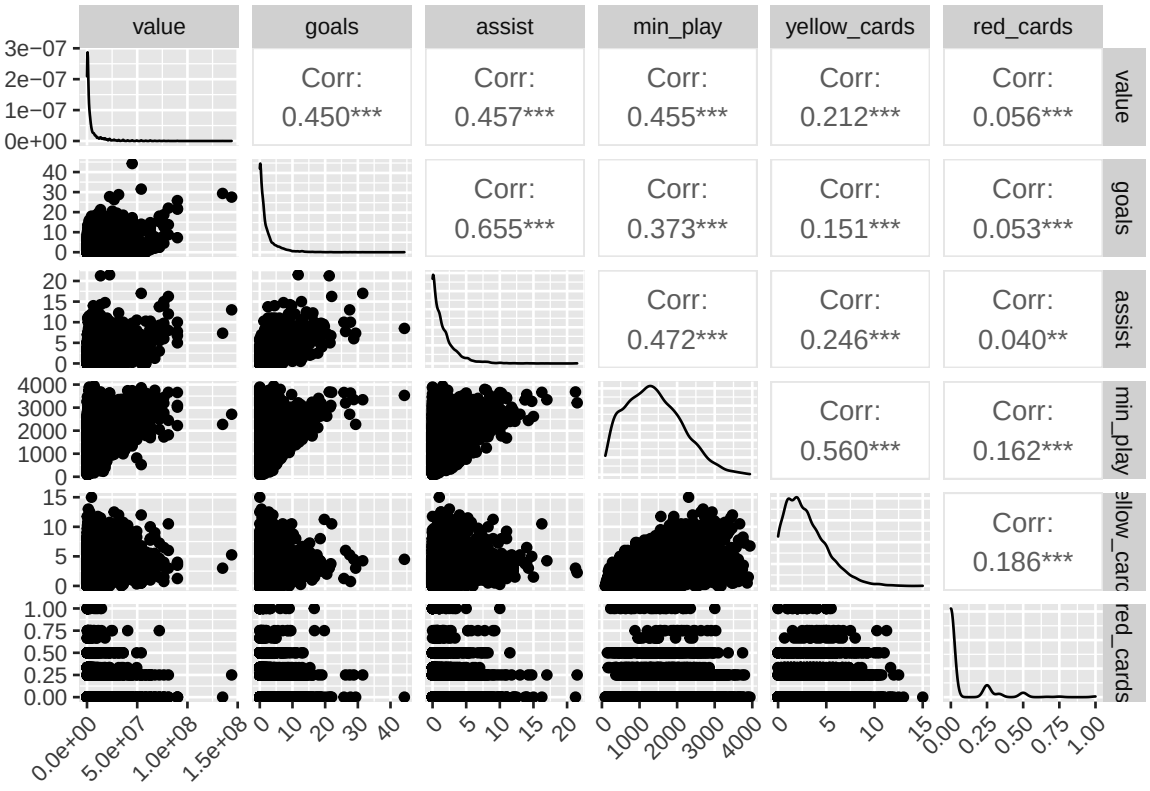
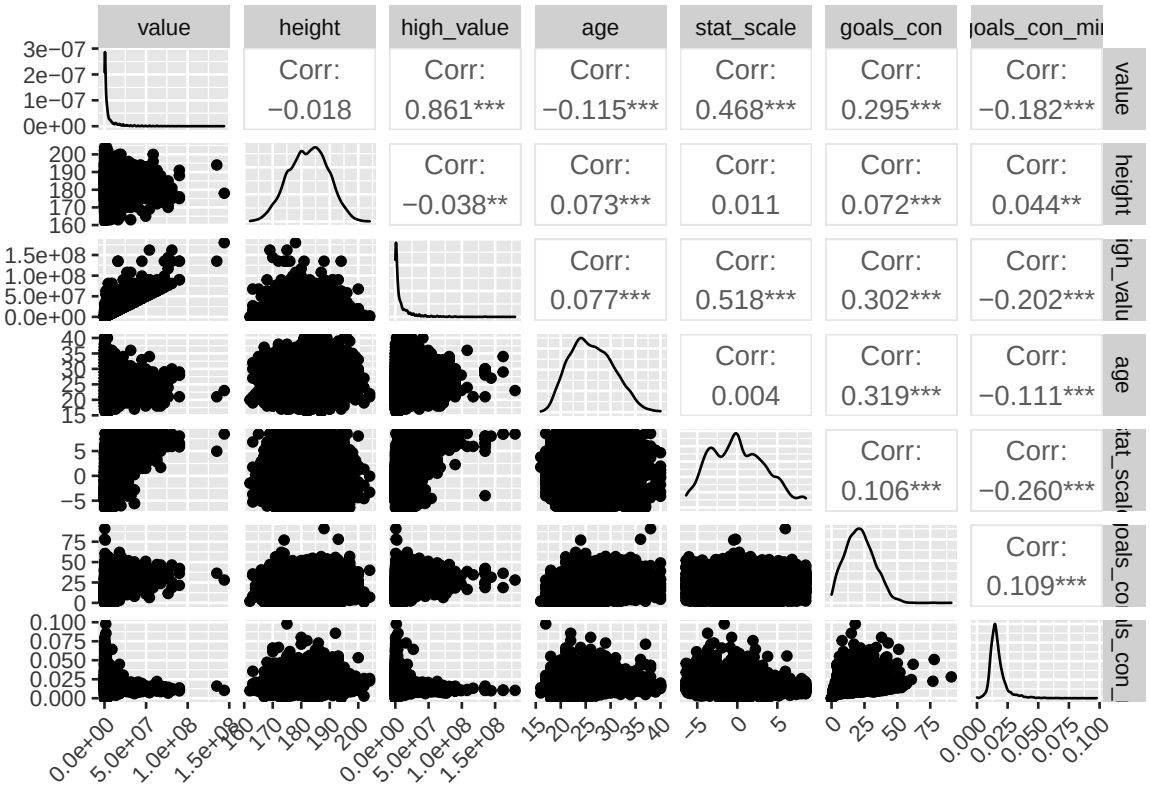


Figura 3.3: Correlaciones



La correlación de las variables con la objetivo se resumen en la siguiente tabla 3.12:

Tabla 3.12: Correlaciones

Variable	Correlacion	Porteros	Defensas	Centrocampistas	Delanteros
goals_average	0.4495175	0.1852426	0.3489489	0.4387393	0.5998325
assists_average	0.4574996	0.1653841	0.3085935	0.4628677	0.5424197
min_played_average	0.4550412	0.4875291	0.4682495	0.4878286	0.5092502
yellow_cards_average	0.2118124	0.1794264	0.2460248	0.2474707	0.1834971
red_cards_average	0.0555784	0.0809536	0.0711716	0.0119905	0.1012784
height_in_cm	-0.0177828	0.0728891	0.0498385	0.0450912	-0.0034654
highest_market_value_in_gbp	0.8611106	0.8741742	0.8640215	0.8613912	0.8577856
age	-0.1149719	-0.1135072	-0.1252273	-0.1117589	-0.0776586
statistics_scale	0.4677396	0.3957115	0.4711371	0.4985083	0.4871217
goals_conceded_season_average	0.2951457	0.3170395	0.2805539	0.3354974	0.3483725
goals_conceded_minute	-0.1824236	-0.2206142	-0.2120501	-0.1697885	-0.1703427

Las variables **edad**, **goles encajados por minuto** y **altura** tienen un índice de correlación negativo con respecto a la variable objetivo (no se tendrá en cuenta la altura ya que es prácticamente cero). Esto indica, que cuanto más mayor sea un futbolista y mayor número de goles encaje por minuto jugado, menor va a ser su *valor de mercado*. Estas dos variables serán analizadas con más exactitud más adelante.

Las demás variables tienen correlación positiva con el *valor de mercado*, por lo que cuanto mayor sea el valor de estas en los futbolistas, mayor valor tendrán. Las variable con mayor correlación es **mayor valor de mercado** (como era de esperar), seguida de las variables **media de goles**, **media de minutos jugados**, **estadística** y **media de asistencias**. A **media de goles** y **media de minutos jugados** se les va a realizar un análisis más específico respecto a la variable objetivo.

Ya se ha contestado parcialmente a la pregunta de **qué** variables cuantitativas influyen

positiva o negativamente en la variable objetivo. A continuación, se va a tratar de responder a la pregunta de **cómo** influyen dichas variables sobre el valor de los jugadores.

También vamos a ver cómo influyen estas variables en función de la posición del futbolista. Se analiza en función de esta variable porque en la práctica, los equipos de *scouting* de los clubes suelen ir buscando un jugador de una posición concreta, para reforzar esa zona del campo, por lo que es interesante evaluar por separado a cada una de las posiciones. Por eso, en la tabla anterior también se muestran los coeficientes de correlación entre las variables en función de la posición que ejerce cada jugador dentro del terreno de juego.

Los gráficos con los que se realiza este proceso están generados mediante las funciones *ggplot*, *geom_point* (representación de puntos) y *geom_smooth* (representa una estimación de la función de densidad) del paquete ggplot2. También se analizan cómo varían estas variables según las posiciones de los jugadores y que se incluye una representación en escala logarítmica para mejorar la visualización de los gráficos.

3.4.3.1. Media de goles

En la figura 3.4 se representa cómo varía el *valor de mercado* en función del número de goles medio que han marcado los jugadores por temporada junto con una estimación de la función de densidad y su intervalo de confianza. En el segundo gráfico se ha realizado una transformación logarítmica a la variable *valor de mercado* para mejorar la visualización del mismo.

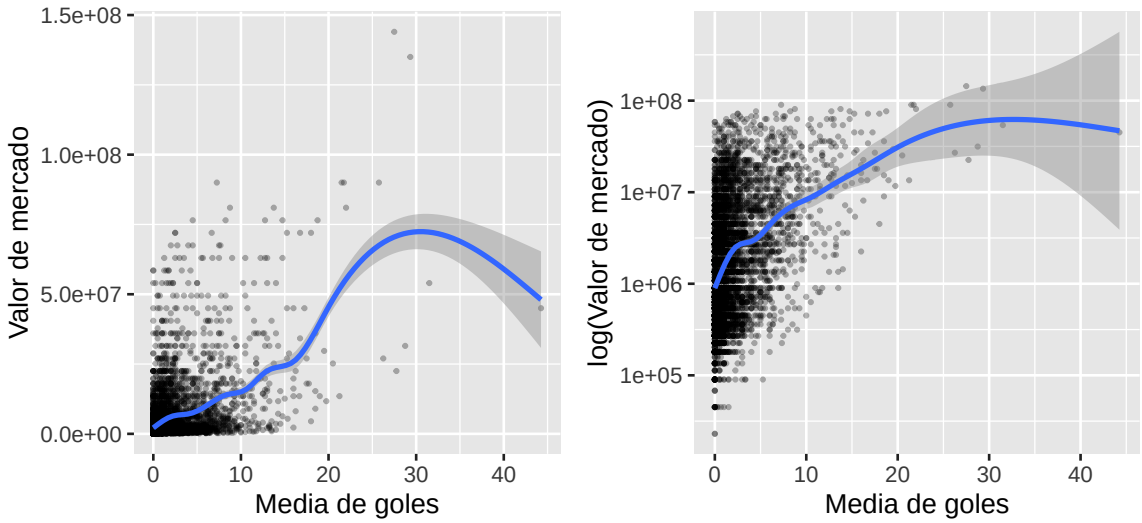


Figura 3.4: Media de goles y valor de mercado

Estos gráficos muestran que el *valor de mercado* aumenta cuando se marcan más goles (correlación positiva como se había comentando anteriormente), pero se observa claramente que cuando más aumenta el valor del jugadores es cuando marca más de 20 goles, hasta ese número, el valor subía pero no de forma tan pronuncidad. También es destacable que cuando se marcan más de 30 goles, el *valor de mercado* disminuye pero hay que tener en cuenta que en ese intervalo sólo hay dos jugadores, de modo que es insuficiente para hacer extrapolaciones. Por último, los dos jugadores que mayor *valor de mercado* tienen (más de 100M) están en el intervalo de 20-30 goles de media por temporada, por lo que se puede concluir que el gol en el fútbol "se paga".

A continuación, en la figura 3.5 se representan los mismos puntos que en los gráficos de la figura 3.4 pero cada posición se dibuja con un color diferente y además, cada una tiene su propia estimación de la función de densidad.

El gráfico posterior (figura 3.6) representa un facetado del *valor de mercado* de los jugadores en función de sus posición. De este han sido eliminados los porteros debido a la correlación tan baja que tienen con esta variable, ya que los porteros no suelen anotar goles y estos no afectan apenas a su *valor de mercado*, como es normal.

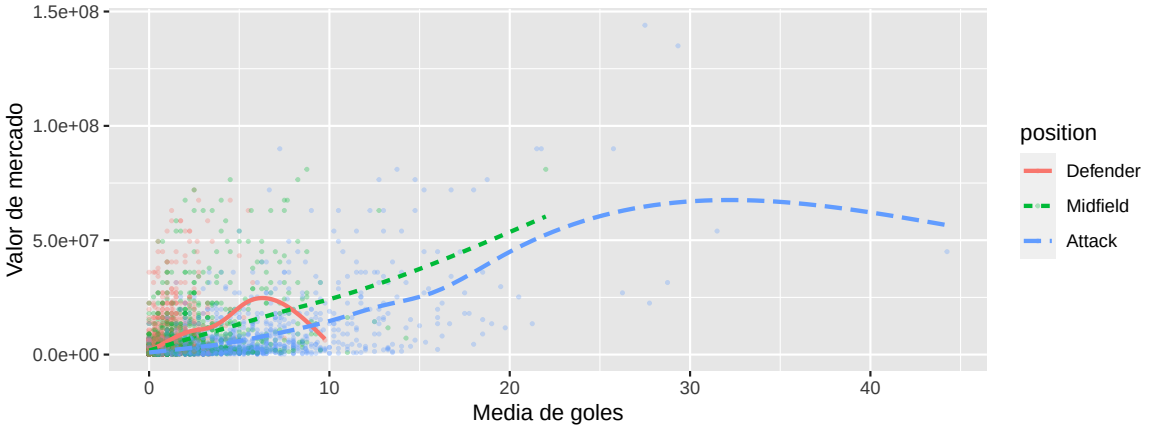


Figura 3.5: Media de goles por posición y valor de mercado

```
jugadores_con_valor %>%
  filter(position != "Goalkeeper") %>%
  ggplot() +
  geom_point(aes(goals_average,market_value_in_gbp), size = 0.5, alpha = 0.3) +
  facet_wrap(~ position, nrow = 2) +
  xlab("Media de goles") +
  ylab("Valor de mercado")
```

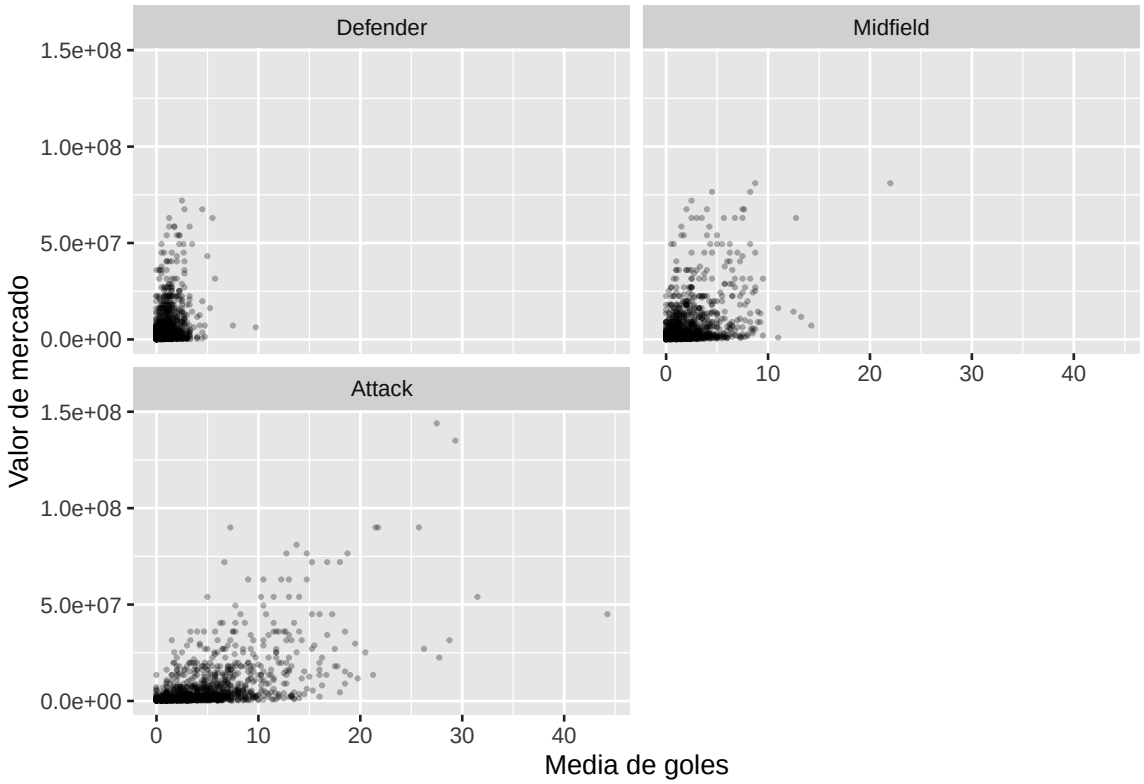


Figura 3.6: Facetado por posición (media de goles)

Se observa que en los centrocampistas nunca disminuye el *valor de mercado* cuando aumenta el número de goles marcados, lo que nos indica que los centrocampistas con llegada al área están muy cotizados en el mercado. En los delanteros, siendo la posición que más relaciona los goles con el valor por su coeficiente de correlación, ocurre algo similar al caso general, que si se superan los 30 goles, el *valor de mercado* baja (sólo superan los 30 goles de media dos futbolistas), esto indica que no sólo influye el gol para su valor, incluso para los delanteros. De hecho, estos jugadores son Lionel Messi y Robert Lewandowski, que tienen cierta edad y esto podría influir en que aunque sean los más goleadores no sean los más valorados monetariamente (recordemos que la edad tiene un coeficiente de correlación negativo). En los defensas, esta variable no influye mucho.

Tabla 3.13: Mayores goleadores

player_id	pretty_name.x	market_value_in_gbp	goals_average
28003	Lionel Messi	5.4e+07	31.50
38253	Robert Lewandowski	4.5e+07	44.25

La media de asistencias tiene un comportamiento muy parecido al que se ha explicado para los goles y por eso no la incluimos en el estudio.

3.4.3.2. Media de Minutos Jugados

En los siguientes dos gráficos (3.7) se muestra como varía el *valor de mercado* en función de la media de minutos jugados por temporada junto con una estimación de la

función de densidad y su intervalo de confianza. En el segundo gráfico se ha realizado una transformación logarítmica a la variable *valor de mercado*. En este caso, no se muestra la evolución por posiciones ya que el comportamiento apenas varía entre ellas y es muy semejante al que aparece a continuación.

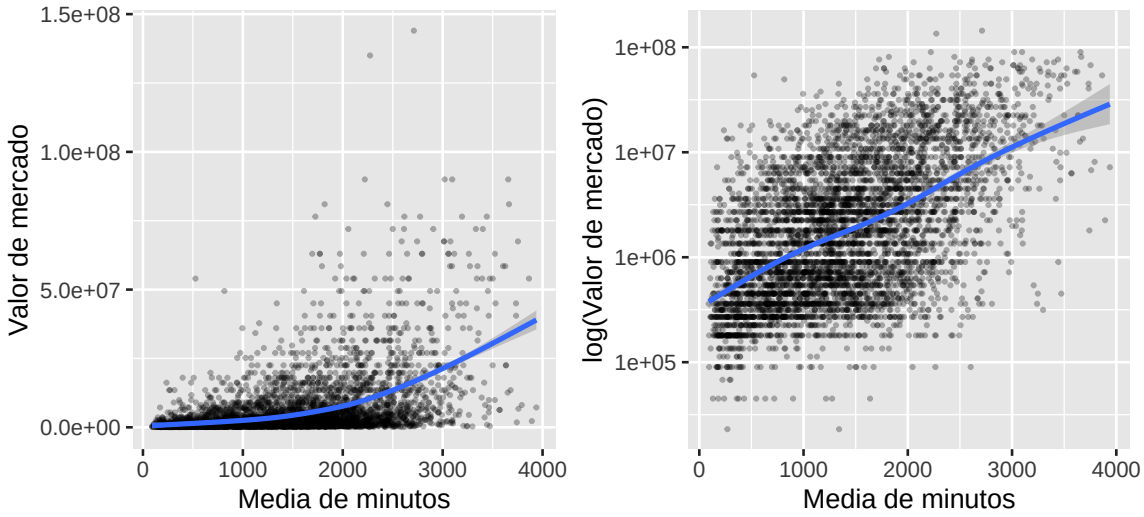


Figura 3.7: Media de minutos jugados y valor de mercado

Los gráficos indican que las zonas más oscuras, es decir, donde hay más jugadores, están entre los 500 y los 2000 minutos medios por temporada.

Claramente el valor de los jugadores crece si juegan más minutos. Así mismo, se observa una subida más rapida a partir de los 2000 minutos, lo que indica que acumular muchos minutos de media por campaña provoca un mayor precio de los futbolistas (suele ser reflejo de la relevancia de los futbolistas en sus equipos). Por lo tanto, en un momento de dificultad económica del club, se puede optar por la compra de jugadores que gocen de menos minutos, darle minutos, que suba su valor y luego vender a dicho futbolista, siempre que otros indicadores hagan pensar que su potencial es bueno su fichaje.

3.4.3.3. Edad

Antes de realizar el análisis de la variable edad junto con el *valor de mercado* vamos observar cómo se comporta la distribución de esta variable por separado. El diagrama de violín (3.8) de esta variable proporciona información sobre qué edad es la mayoritaria entre los jugadores presentes en la base de datos del estudio. Se observa que la mayoría se encuentran en la franja de edad entre 20 y 30 años. No hay ningún futbolista menor de 15 años, ni mayor de 40 años (a fecha del estudio, porque en unos días llegará a los 41 Joaquín, jugador del Real Betis Balompié).

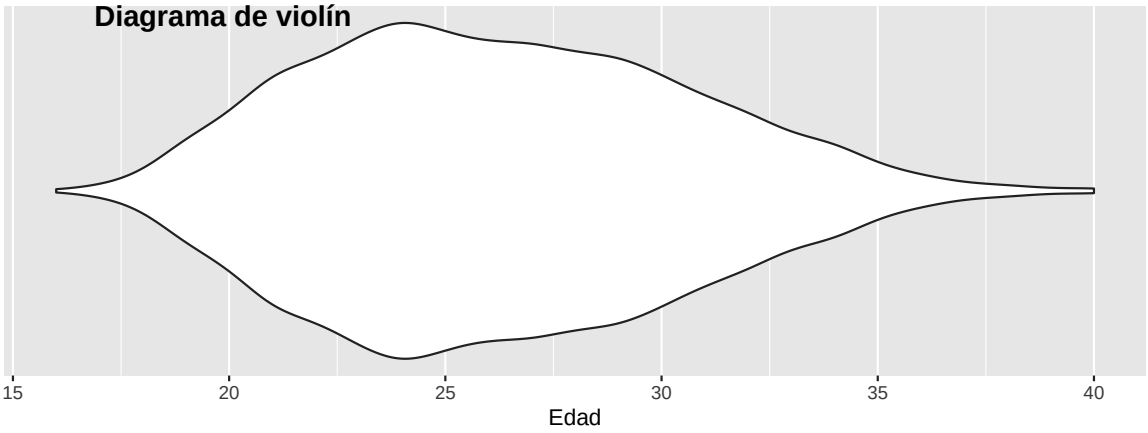


Figura 3.8: Diagrama de violín edad

Los gráficos de la figura 3.9 muestran cómo varía el *valor de mercado* en función de la edad de los jugadores junto con una estimación de la función de densidad y su intervalo de confianza. En el segundo gráfico se ha realizado una transformación logarítmica.

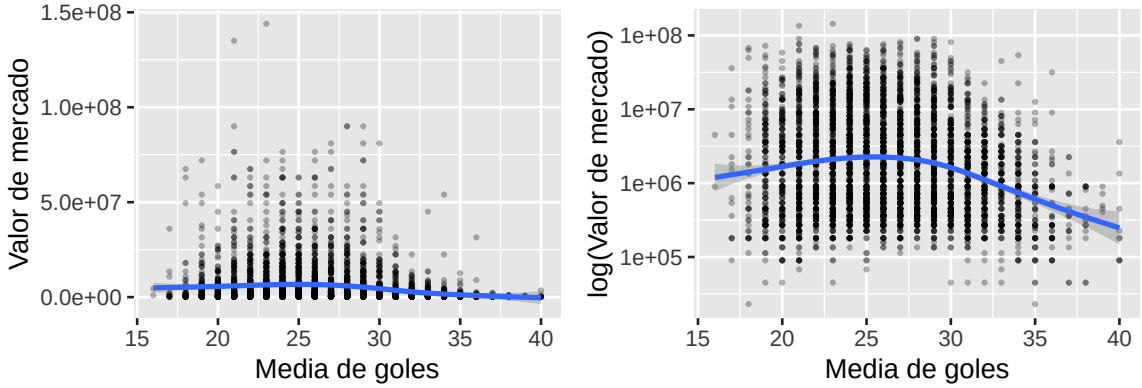


Figura 3.9: Edad y valor de mercado

Según los gráficos anteriores, el *valor de mercado* aumenta tímidamente desde el inicio de la carrera del futbolista (al ser joven se tiene un buen *valor de mercado* por lo que puede prometer) hasta la edad de 26, donde empieza el declive del valor que no cesa de disminuir hasta acabar su carrera. Que haya una pequeña subida en el valor, hasta la mitad de la carrera de los deportistas, hace que aunque la correlación con la edad sea negativa, el coeficiente tenga un valor muy pequeño. Por lo tanto, lo ideal para una secretaría técnica sería comprar futbolistas jóvenes (17-19 años) y venderlos antes de que cumplan treinta años, para así obtener el máximo beneficio posible.

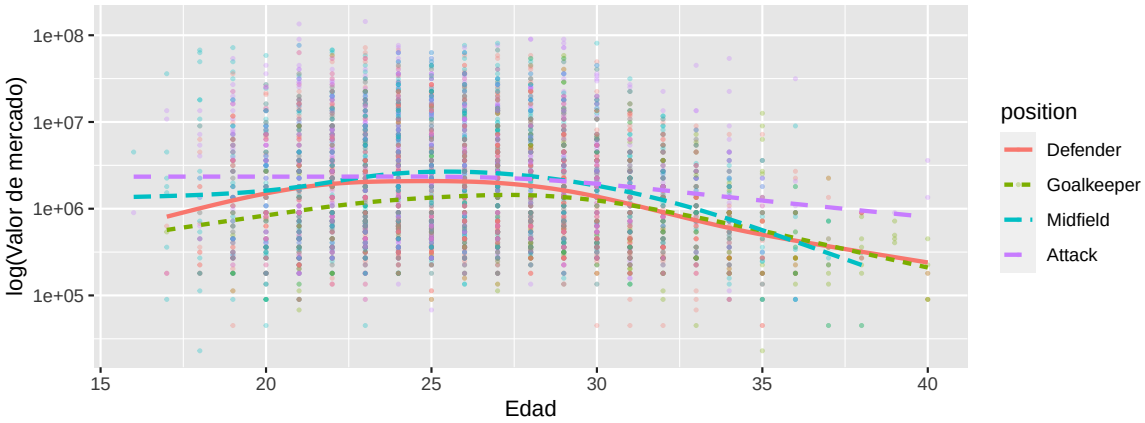


Figura 3.10: Edad por posición y valor de mercado

Analizando por separado las posiciones, en la figura 3.10 (con la transformación logarítmica realizada), se podría destacar que los que antes empiezan a perder valor son los defensas, pero tampoco con una gran diferencia, ya que el comportamiento es similar para las cuatro posiciones.

Hay que destacar que como la edad se ha calculado manualmente y no se va actualizando, es una variable cuantitativa discreta, ya que sólo puede tomar valores enteros y por eso en el gráfico de puntos aparecen esas bandas en cada valor de la edad.

3.4.3.4. Goles encajados por minuto

Las siguientes gráficos (figura 3.11) representan cómo varía el *valor de mercado* en función del número de goles encajados por minuto jugado por parte de los jugadores, junto con una estimación de la función de densidad y su intervalo de confianza. En el segundo gráfico se ha realizado una transformación logarítmica a la variable *valor de mercado* para mostrar de forma más clara la tendencia.

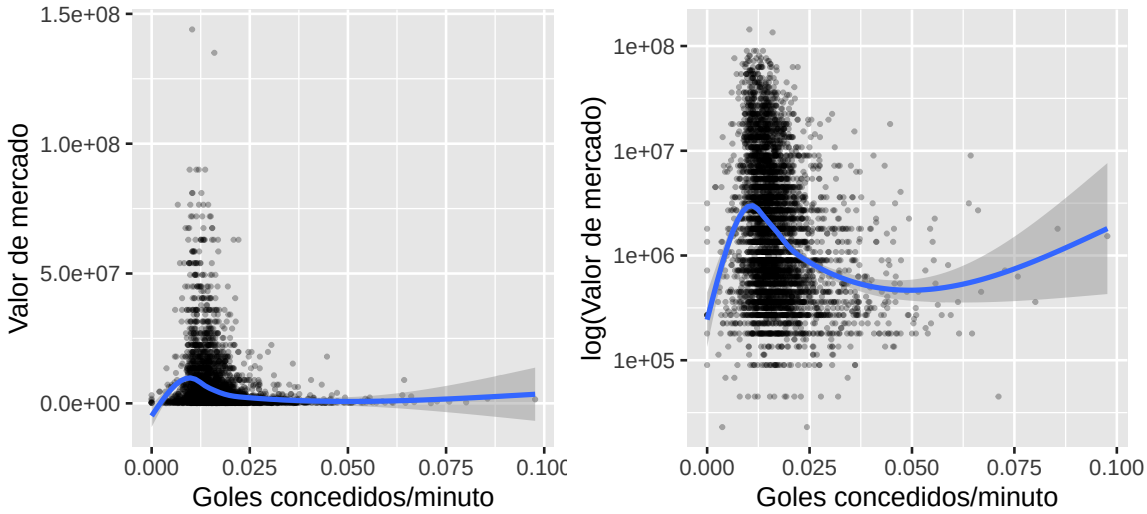


Figura 3.11: Goles concedidos por minuto y valor de mercado

Al principio el *valor de mercado* crece, ya que los jugadores que tienen muy pocos minutos jugados tendrán pocos goles encajados (se ha rectificado esta variable gracias a

dividirla por el número medio de minutos jugados). Sin embargo, después de una pequeña subida donde hay pocas observaciones, donde se encuentran la mayoría de estas, el *valor de mercado* baja. Por lo tanto, se puede concluir que para la mayoría de los jugadores tener una tasa alta de goles encajados por minuto es perjudicial para su *valor de mercado*. También destaca que la bajada de los centrocampistas es más brusca.

A continuación, en la figura 3.12 se representan los goles encajados por minuto junto con el logaritmo del *valor de mercado*, pero con un color diferente para cada posición.

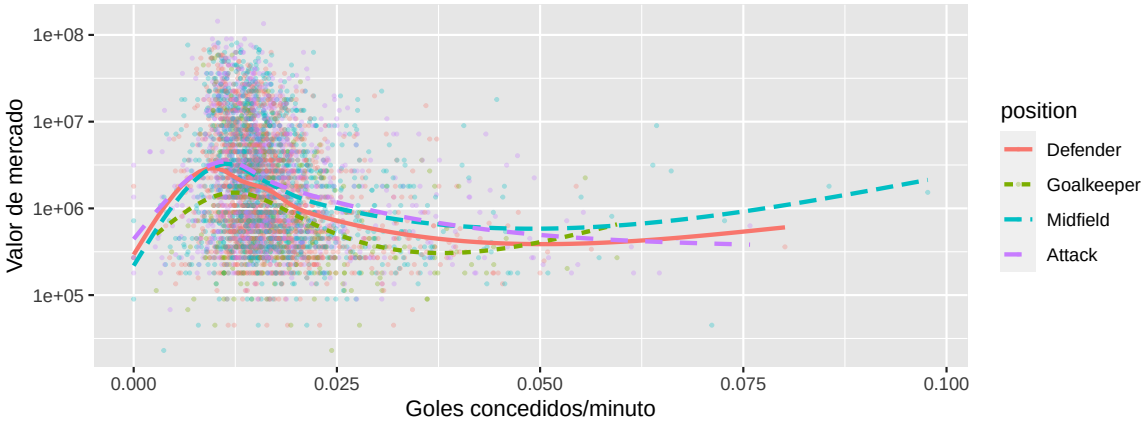


Figura 3.12: Goles concedidos por minuto y valor de mercado por posición

El gráfico 3.13 representa un facetado del *valor de mercado* de los jugadores en función de su posición.

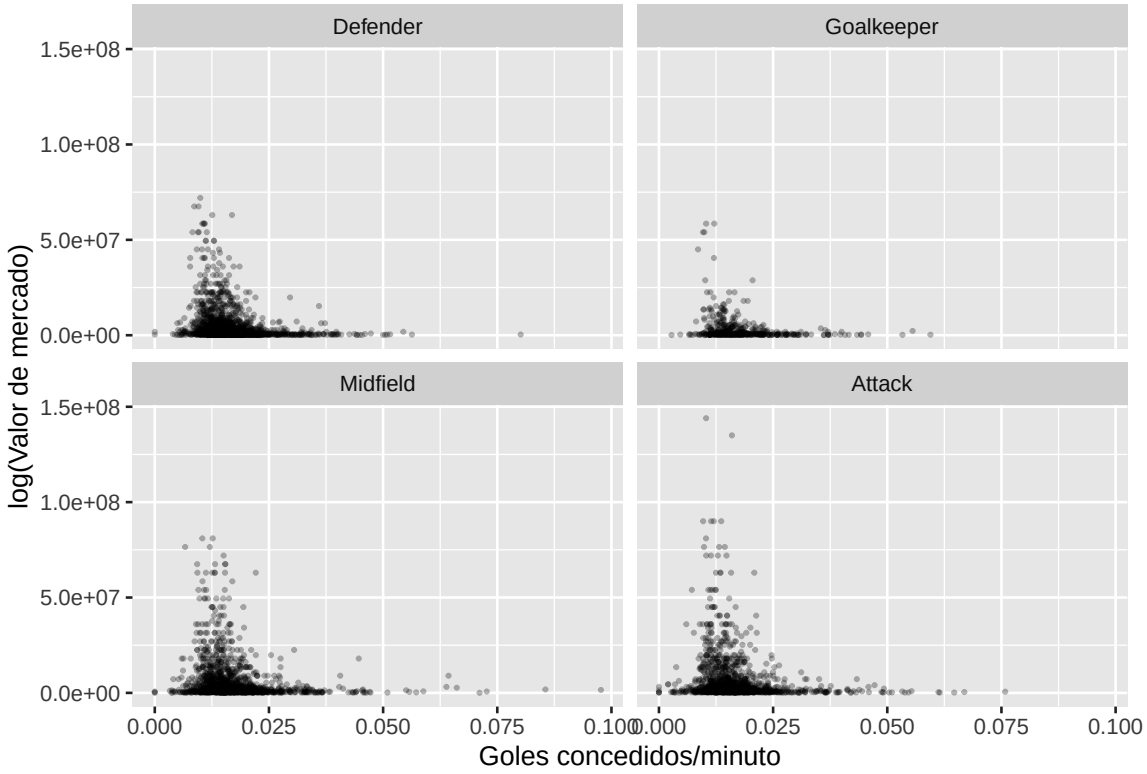


Figura 3.13: Facetado por posición (goles concedidos por minuto)

Las dos últimas representaciones de la figura 3.13 indican que influyen más los goles encajados en los porteros y los defensas, debido a que se dedican a la faceta defensiva del juego. Esto también se puede observar en la tabla de correlaciones, es decir, que la correlación será más negativa para los porteros y defensas.

3.4.4. Estudio de las variables cualitativas

Una vez realizado el estudio de las variables cuantitativas, se debe comprobar si pertenecer a una clase o a otra dentro de las variables cualitativas influye en el *valor de mercado* del futbolista. En esta sección se va a dar respuesta a preguntas como: ¿qué vale más, un delantero o un defensa?, ¿un diestro o un zurdo?, ¿un europeo o un americano?. Así, podemos aconsejar a las secretarías técnicas de los clubes qué características deben tener los jugadores para que el fichaje sea más barato.

3.4.4.1. Posición

Empezamos por estudiar si la posición que suele ocupar el jugador en el campo influye en su valor (portero, defensa, centrocampista o delantero). Para ello, se realiza una tabla resumen con la media del *valor de mercado* por posición (3.14) y tres gráficos (medias, "cajas y bigotes" y de violín).

Tabla 3.14: Valor por posición

position	Cantidad	valor_medio
Defender	1833	4540865
Goalkeeper	480	3202660
Midfield	1498	6009555
Attack	1353	6736254

En media, los delanteros tienen mayor *valor de mercado* que todos lo demás, lo que está relacionado con lo dicho anteriormente, que la media goleadora aumentaba el *valor de mercado*, por lo tanto, se vuelve a concluir que en el fútbol, el gol "se paga". Sin embargo, es un poco contradictorio con el sentido del juego que los porteros sean los que menor valor medio tengan, ya que es tan importante meter goles como que no te los metan.

En el primer gráfico de la figura 3.14, no se puede distinguir si las medias son muy diferentes. Por lo tanto, los dos siguientes van a ser representados con la transformación logarítmica realizada previamente a la variable *valor de mercado*.

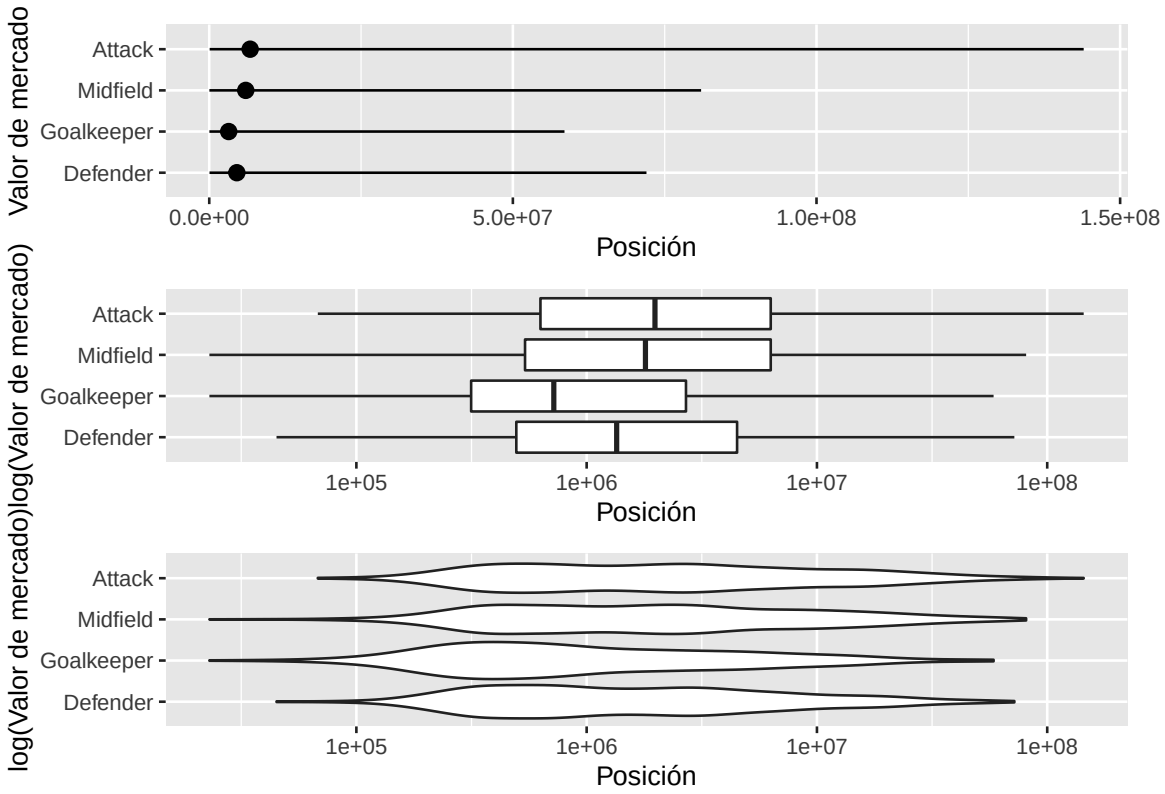


Figura 3.14: Valor de mercado por posición

De las dos últimas representaciones del la figura 3.14, los delanteros son los que mayor media de *valor de mercado* tienen, como se ha dicho antes, seguidos muy de cerca por los centrocampistas. Una cosa reseñable sobre los delanteros es que su valor extremo inferior es considerablemente mayor que el de los centrocampistas. Es destacable que los porteros tienen un *valor de mercado* por debajo de las tres posiciones restantes y que el valor más alto de un portero es cercano a la mediana del de los delanteros.

Por lo tanto, la conclusión es que el *valor de mercado* de un futbolista **sí** depende de la demarcación habitual que ocupe y esta variable será tratada como una variable *dummy* a la hora de realizar la modelización de la variable objetivo.

3.4.4.2. Pie

La variable factor **pie** indica si el jugador es diestro, zurdo o ambidiestro (ver tabla 3.15). Vamos a estudiar mediante el gráfico de la figura 3.15 si existe alguna diferencia entre estas tres categorías a la hora de evaluar el valor de un jugador.

Tabla 3.15: Valor por 'pie bueno'

foot	Cantidad	valor_medio
Both	219	5394767
Left	1298	5564934
Right	3559	5481761

El valor medio de mercado de las tres categorías es muy parecido.

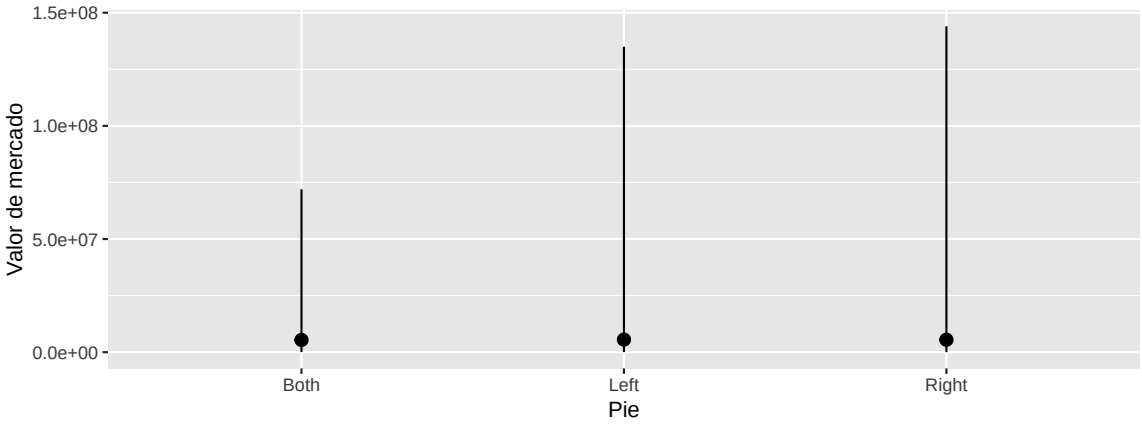


Figura 3.15: Pie y valor de mercado

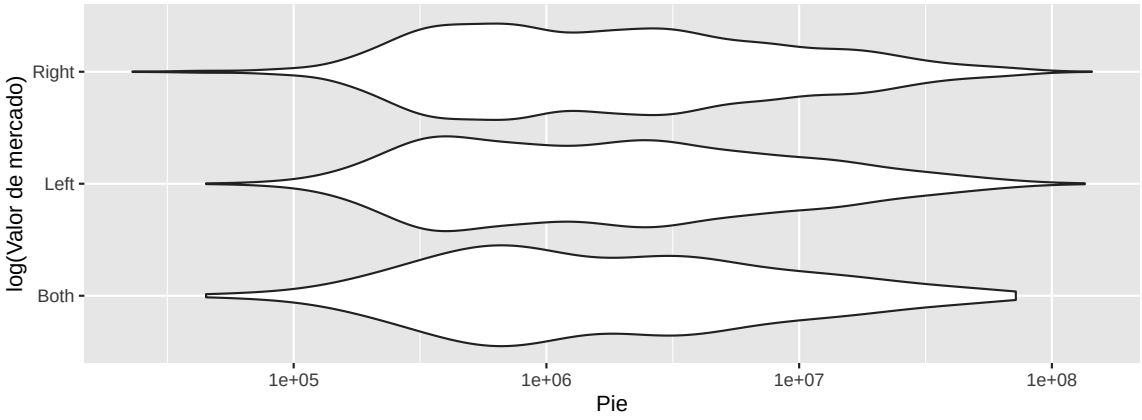


Figura 3.16: Pie y valor de mercado

Se observa claramente en las figuras 3.15 y 3.16, que no hay diferencias significativas en el valor del mercado en función del "pie bueno" del futbolista. En este caso, no ha sido necesario realizar más gráficos, ya que con estos se aprecia perfectamente lo explicado. Lo único que se podría destacar, es que los jugadores con un altísimo valor, tienen un pie bien definido, es decir, todos los jugadores que superan los 100 millones no son ambidiestros.

3.4.4.3. Continente

En esta ocasión, vamos a estudiar si dependiendo del continente de procedencia del futbolista el *valor de mercado* puede cambiar sustancialmente. Para ello vamos a realizar el gráfico de cajas y bigotes con la transformación logarítmica aplicada para que se facilite la interpretación.

Tabla 3.16: Valor por continente

continent	Numero	valor_medio
Africa	531	5005388
Asia	270	3000381

Tabla 3.16: Valor por continente (continúa)

continent	Numero	valor_medio
Europe	3742	5384475
North America	89	6938348
Oceania	17	1413529
South America	515	7221235

En la tabla resumen de los valores medios 3.16, se observa que la mayoría de los jugadores son europeos, ya que los datos son de 14 ligas europeas, pero la cantidad de sudamericanos, africanos y asiáticos que hay es suficiente como para tenerlos en cuenta.

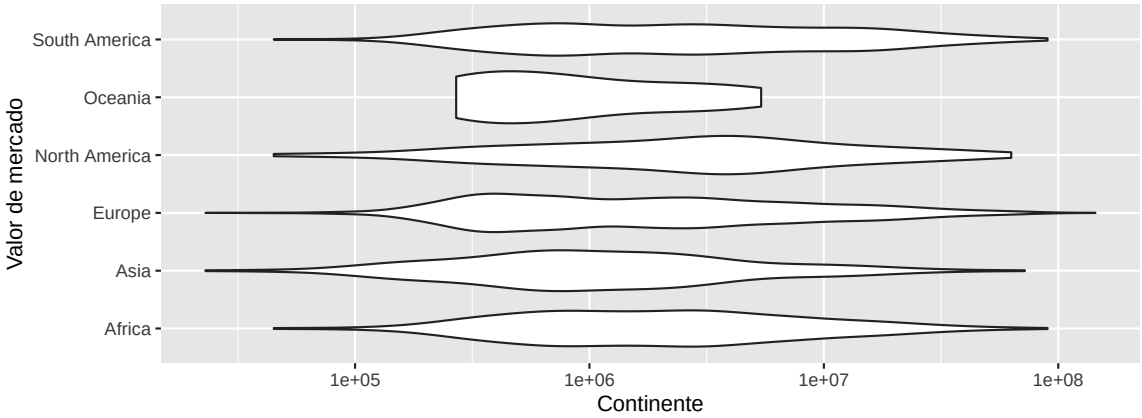


Figura 3.17: Pie y valor de mercado

En la figura 3.17 se aprecia que los americanos (tanto del norte como del sur) tienen valores medios de mercado superiores al resto. Hay que tener en cuenta el reducido número de norteamericanos que juegan en estas catorce ligas, sólo 89, lo que podría ser un número demasiado bajo, por lo que se intuye que todos los que han venido tienen cierto nivel.

Sin embargo, los de mayor *valor de mercado* medio son los americanos del sur y de ellos sí hay un número apreciable, lo cual se debe a que los fichajes extra continentales suelen ser de garantías y de jugadores con gran proyección. También hay que tener en cuenta, que la mayoría de ligas europeas solo permiten tres extra-comunitarios, lo que hace que los equipos si tienen un sudamericano, sea un jugador de alto valor y de calidad.

3.4.4.4. Ligas

Por último, se estudia si jugar en una liga u otra influye en el *valor de mercado*. Para ello se realiza un diagrama de puntos con la media de los valores de cada liga (figura 3.18) con transformación logarítmica sobre la variable objetivo.

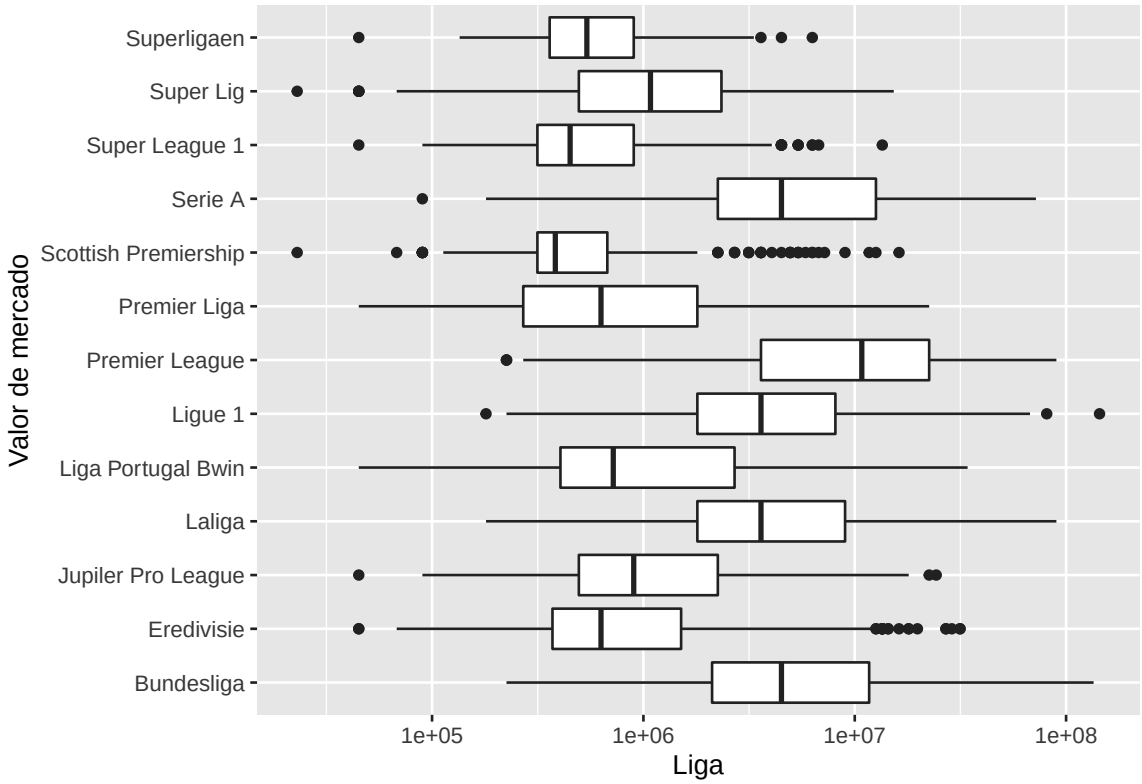


Figura 3.18: Ligas y valor de mercado

De este gráfico se puede extraer una conclusión clara, que los jugadores de las cinco grandes ligas, que son Bundesliga, Laliga, Ligue 1, Premier League y Serie A, tienen un *valor de mercado* medio mayor que el de las demás, por lo que a la hora de cuantificar el valor de un jugador, si se quiere realizar una incorporación a una plantilla, es importante tener en cuenta en qué liga juegan los posibles candidatos, ya que la liga influye claramente en el *valor de mercado*. Esta variable no va a ser incluida en el modelo de predicción ya que se va a incluir la variable **estadistica**, que es cuantitativa y está directamente correlacionada con la liga en la que milita cada futbolista.

NOTA: Si se desea ver el código completo de la sección de análisis de los datos, ir al apartado A.3 (parte del apéndice A).

Capítulo 4

Modelización

En este apartado se va a tratar de obtener un modelo que prediga adecuadamente los valores de mercado de los futbolistas. Para ello, se realizan evaluaciones con varios modelos y se decide el mejor teniendo en cuenta el R^2 y la raíz del error cuadrático medio (definición en la sección 2.1.3 de premilinares).

4.1. Diseño

Para poner en contexto lo que se va a realizar en esta sección, primero se incluye una breve descripción de cada uno de los modelos que se usan. Todos los modelos son de regresión, ya que el objetivo es predecir el valor numérico de una variable cuantitativa, no de clasificarla. Los modelos que van a ser utilizados son: árbol de decisión, bosques aleatorios, vecino más cercano, regresión lineal y redes neuronales. Posteriormente, se analizará la ingeniería de características utilizada, se explicará como se ha realizado el entrenamiento de los datos (validación cruzada) y cómo se han obtenido los parámetros auxiliares de los modelos que se realizan (tuning).

Antes de comenzar la descripción de los modelos, hay que destacar que su entrenamiento y test se ha realizado con funciones de la librería *tidymodels* de R. El procedimiento a seguir, aunque posteriormente se explicará con más exactitud, se resume así:

1. Definición del modelo, indicando los parámetros y el motor correspondientes. En nuestro caso, como se va a utilizar tuning, no se indican los parámetros del modelo si no que le pide que devuelva los parámetros necesarios para que el modelo tenga las mejores métricas que se desean.
2. División de la muestra en muestra de entrenamiento y test.
3. Obtención de particiones para realizar la validación cruzada.
4. Ingeniería de características (descripción de la receta).
5. Creación de un workflow que integre todos los pasos anteriores.
6. Entrenar el modelo partiendo del workflow. Se obtienen las predicciones y métricas.
7. Guardar el que proporciona mejores métricas y crear un nuevo workflow con él.

8. Aplicar el modelo seleccionado a la muestra test, obtener las predicciones y sus métricas (R^2 y la raíz de error cuadrático medio). También es interesante comprobar que no han salido predicciones negativas (evaluación).

Nota: Se va a incluir el código correspondiente al modelo de árboles de decisión, para los demás sería análogo cambiando el modelo, su motor y el nombre de los parámetros, excepto para redes neuronales que se realizará con la librería *Keras*. Para ver el código de todos los modelos ir a la sección A.4 del apéndice A.

4.2. Descripción de los modelos.

A continuación, se va a realizar una breve descripción de los modelos que se van a utilizar para predecir el *valor de mercado*.

4.2.1. Árbol de decisión (decision tree)

Un árbol de regresión consiste en una serie de reglas de "reparto" o división del espacio definido por las variables predictoras.

Estos conducen a estratificar o segmentar el espacio predictor en un número de regiones simples, de forma que para predecir una observación determinada se utiliza la media o moda de la región a la que pertenece.

Un árbol de decisión se construye en dos pasos:

- **Paso 1:** Se construye una partición del espacio predictor en J regiones, R_j ($j = 1$,.., J). En nuestro caso, como es una regresión, las diferentes particiones serán intervalos de *valor de mercado*.
- **Paso 2:** Para cada observación que cae en la región R_j , hacemos la misma clasificación (predicción): la clase más frecuente o clase modal (media de los valores) de la respuesta para las observaciones de la muestra incluidas en R_j.

La división del espacio predictor se realizará de forma que estas divisiones optimicen la función objetivo.

Como divide el espacio predictor en intervalos, cabe esperar que haya varios jugadores con el mismo valor de predicción de *valor de mercado*.

Los parámetros de los árboles de decisión son: la profundidad máxima del árbol (tree_depth), el número mínimo de datos en un nodo que se requieren para que el nodo se divida más (min_n) y el coste de complejidad (cost_complexity). En nuestro modelo sólo se van a usar los dos primeros.

Definimos el modelo:

```
modelo_tree <- decision_tree(tree_depth = tune(),
                             min_n = tune()) %>%
  set_engine("rpart") %>%
  set_mode("regression")
```

Como se ha comentado antes, se va a realizar tuning de hiperparámetros para tomar los mejores según diferentes métricas como será explicado más adelante.

4.2.2. Bosques aleatorios (random forest)

Los bosques aleatorios consisten en construir una serie de árboles sobre muestras de entrenamiento bootstrap (con reemplazamiento). En la construcción de estos árboles, cada vez que se va a realizar una división de un nodo, se utiliza una muestra aleatoria de m (m $<$ p) predictores, donde p es es número de variables predictoras (en nuestro caso p = 13).

El método de bosques aleatorios consiste en aplicar bagging (concepto explicado en la sección 2.1.3 de los preliminares), usando árboles de decisión como predictores, pero construidos eligiendo en cada nodo, para bifurcar o dividir, el mejor predictor entre una muestra aleatoria de los atributos o predictores.

La razón de realizar este modelo en lugar de bagging es que si hay un predictor muy fuerte en el conjunto de datos, junto con un número de otros predictores moderadamente fuertes, en la colección de árboles bagging, la mayoría o la totalidad de los árboles utilizarían este predictor en la primera división y luego ya usaría los otros predictores moderadamente fuertes.

En consecuencia, todos los árboles que se crean serían muy similares entre sí y provocaría que las predicciones de los árboles estarían muy correladas entre ellas y no provocaría mucha disminución y por tanto en la raíz del error cuadrático medio.

Por esto, el uso de los bosques soluciona este problema, contando con un conjunto de árboles donde cada uno de ellos puede ir empleando distintas variables predictoras en cada nodo, de modo que esa aleatoriedad en la selección de variables de partida introduce más diversidad, y del consenso entre todos los árboles saldrá una predicción más robusta.

Los parámetros de este modelo son: el número de predictores que se muestrearán aleatoriamente en cada división al crear los modelos de árbol (mtry), el número mínimo de datos en un nodo que se requieren para que el nodo se divida más (min_n) y el número de árboles contenidos en el conjunto (trees). Se hará uso de los dos primeros.

Definimos el modelo:

```
rf_tuner <-
  rand_forest(mtry = tune(),
              min_n = tune()) %>%
    set_engine("ranger") %>%
    set_mode("regression")
```

Los hiperparámetros se van a obtener de la misma forma que antes, en el proceso lanzado por tune_grid (por defecto, los hiperpárametros se toman de las funciones del modelo subyacente).

4.2.3. Vecinos más cercanos (knn)

KNN es un modelo sencillo, perezoso y muchas veces efectivo que clasifica nuevas instancias como la clase mayoritaria de entre los k vecinos mas cercanos de entre los datos de entrenamiento. Durante el entrenamiento, sólo guarda las instancias, no construye ningún modelo (a diferencia de los árboles de decisión).

KNN es local ya que asume que la clase de un dato depende sólo de los k vecinos mas cercanos y como en este caso estamos haciendo un modelo de regresión calcula la media de los k vecinos.

Sin embargo, para aplicar kNN necesitamos elegir un valor apropiado para k, y el éxito de la regresión depende mucho de este valor. En cierto sentido, el método kNN está sesgado por k.

Para que kNN sea menos dependiente de la elección de k, se propuso observar múltiples conjuntos de vecinos más cercanos en lugar de solo un conjunto de k-vecinos más cercanos.

Para ello, vamos a definir una malla que va desde 1 a 30 vecinos para que por el método tuning tome el mejor valor de vecinos en función del R^2 y de la raíz del error cuadrático medio. El parámetro k (neighbors) es el único que vamos a utilizar en este modelo.

Definimos el modelo y la red de vecinos:

```
k1_30 <- expand_grid(neighbors = 1:30)

knn_tuner <-
  nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn") %>%
    set_mode("regression")
```

4.2.4. Regresión lineal

Al realizar la regresión con trece variables predictoras estamos utilizando la regresión lineal multiple. Con ella, pretendemos explicar el comportamiento de la variable dependiente (variable objetivo).

Un modelo de regresión lineal se define teóricamente de la siguiente forma:

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k + \epsilon$

Tenemos que tener en cuenta que tenemos variables cualitativas y serán transformadas como variables *dummy* para poder incluirlas en el modelo. La estrategia es que si la variable cualitativa tiene p categorías se utilizarán $p - 1$ variables dicotómicas y la última variable creada será combinación lineal de las demás.

Definimos el modelo:

```
modelo_lm <-
   linear_reg() %>%
   set_engine(engine = "lm")
```

4.2.5. Redes neuronales

Las redes neuronales artificiales son un un modelo matemático que se basa en potentes algoritmos de aprendizaje automático inspirado en procesos biológicos a partir de neuronas. Esos modelos matemáticos están basados en una estructura de grafo dirigido cuyos nodos son neuronas artificiales.

Cada neurona artificial se conecta a otras unidades a través de arcos dirigidos. Cada arco $j \to i$ sirve para enviar la salida a_j de la unidad j a la unidad de entrada i. Cada arco $j \to i$ tiene asociado un peso w_{ji}.

Las neuronas calculan sus salidas en función de las entradas que le llegan, además, cada salida sirve como entrada para otras neuronas. Hay que destacar que la red recibe una

serie de entradas externas y devuelve una serie de unidades de salida al exterior, que son salidas de algunas neuronas.

La salida de las neuronas se calcula de la siguiente forma:

$$a_i = g(\sum_{j=0}^{n} w_{ji} a_j),$$

donde n es el número de entradas hacía la neurona a_i y g es la función de activación.

Hay que destacar que para $j = 0$ se considera una entrada ficticia $a_0 = -1$ y un peso w_{0i} (umbral para que se active la neurona).

La función de activación g normaliza la salida cuando el umbral de entrada se supera y provoca que la red no se comporte siempre como un función lineal. Las funciones de activación más importantes son la bipolar, la umbral y la sigmoide (siendo esta derivable).

En nuestro caso, se usan redes neuronales predictoras y hacia delante. Las unidades (neuronas) en este caso, suelen estructurarse en capas (capa de entrada, capas ocultas y capa de salida).

4.3. Muestra de entrenamiento y test

Para la división de la muestra, en conjunto de entrenamiento y test se va a tomar cuatro estratos, en función de la variable objetivo, para que estén representadas en las dos divisiones jugadores de todos los rangos de valores de mercado y no se produzca, por ejemplo, que en la muestra test no haya ningún jugador con valor superior a 5 millones, lo cual podría provocar errores en la predicción.

Para que el documento sea reproducible, se añade una semilla.

```
set.seed(46072925)
```

```
jugadores_split_estratos <- jugadores_con_valor %>%
  initial_split(strata = market_value_in_gbp,
                breaks = 4)
```

A continuación, realizamos el entrenamiento y test, respectivamente.

```
jugadores_train <- training(jugadores_split_estratos)
jugadores_test <- testing(jugadores_split_estratos)
```

4.4. Validación cruzada

La validación cruzada es un método de remuestreo de datos para evaluar la capacidad de generalización de los modelos predictivos y para evitar el sobreajuste. La validación cruzada es uno de los métodos más usados para estimar el verdadero error de predicción de los modelos y ajustar los parámetros del modelo.

En el caso actual, se utilizan diez pliegues y cuatro estratos, para mantener la proporción en cada uno de ellos. Además, para el cálculo de las métricas y de las predicciones se realizará la media de lo calculado en cada estrato.

```
cv_folds <-
    vfold_cv(jugadores_train,
             v = 10,
             strata = market_value_in_gbp,
             breaks = 4)
```

4.5. Ingeniería de características

La ingeniería de características consiste en preparar los datos de forma correcta para su posterior modelización. Esta preparación se realiza por medio de las recetas con las que se pueden usar secuencias canalizables similares a dplyr.

En la receta definida a continuación hemos realizado las siguientes transformaciones sobre las variables predictoras:

- Tratar a las variables cualitativas (nominales) como variables dummy (step_dummy).
- Eliminar algunas de las variables numéricas que estén muy correladas (step_corr).
- Normalizar los datos numéricos para tener una media de cero (centrar).
- Normalizar los datos numéricos para que tengan una desviación estándar de uno (escalar).

```
receta <- recipe(market_value_in_gbp ~. , data = jugadores_con_valor) %>%
  step_dummy(all_nominal()) %>%
  step_corr(all_numeric(),-all_outcomes()) %>%
  step_center(all_numeric(),-all_outcomes()) %>%
  step_scale(all_numeric(),-all_outcomes())
receta
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         13
##
## Operations:
##
## Dummy variables from all_nominal()
## Correlation filter on all_numeric(), -all_outcomes()
## Centering for all_numeric(), -all_outcomes()
## Scaling for all_numeric(), -all_outcomes()
```

A continuación, se crea un workflow con la receta y el modelo. Este workflow crera flujos de trabajo confiables que generan predicciones precisas para datos futuros.

```
tree_wf <-
  workflow() %>%
```

```
  add_recipe(receta) %>%
  add_model(modelo_tree)
```

4.6. Tuning

Como se ha comentado anteriormente, los hiperparámetros de todos los modelos no se van a establecer manualmente, si no que van a tomar los que sean los más adecuados para mejorar las métricas.

El objetivo del tuning es facilitar el ajuste de hiperparámetros para los paquetes tidymodels. Se basa en gran medida en la receta anteriormente definida y es lo que vamos a utilizar para calcular los hiperparámetros.

```
tree_results <- tree_wf %>%
  tune_grid(resamples = cv_folds)
```

Una vez calculados, se van a guardar dos modelos finales (para cada modelo definido al principio), uno el que mayor R^2 tenga y otro el que menor error cuadrático tenga. Si ambos son el mismo, sólo se guardará uno.

En la tabla 4.1 se muestran los mejores modelos en función de la raíz del error cuadrático medio.

```
tree_rmse_best <- tree_results %>%
  show_best(metric = "rmse")
tree_rmse_best %>%
  kable(caption = "\\label{tabla42}Mejores modelos por rmse", booktabs = T,
        longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 10,
                repeat_header_text = "(continúa)")
```

Tabla 4.1: Mejores modelos por rmse

tree_depth	min_n	.metric	.estimator	mean	n	std_err	.config
14	35	rmse	standard	4785269	10	258454.1	Preprocessor1_Model06
13	32	rmse	standard	4785269	10	258454.1	Preprocessor1_Model10
6	23	rmse	standard	4785411	10	249053.9	Preprocessor1_Model02
9	20	rmse	standard	4785411	10	249053.9	Preprocessor1_Model08
11	16	rmse	standard	4806851	10	248935.4	Preprocessor1_Model05

Para que se minimice la raíz del error cuadrático medio, la profundidad máxima del árbol deberá ser de 14 y el número mínimo de datos en un nodo que se requieren para que el nodo se divida más, deberá ser de 35.

En la tabla 4.2 se muestran los mejores modelos en función del R^2.

```
tree_rsq_best <- tree_results %>%
  show_best(metric = "rsq")
tree_rsq_best %>%
  kable(caption = "\\label{tabla41}Mejores modelos por rsq", booktabs = T,
        longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 10,
                repeat_header_text = "(continúa)")
```

Tabla 4.2: Mejores modelos por rsq

tree_depth	min_n	.metric	.estimator	mean	n	std_err	.config
6	23	rsq	standard	0.7899044	10	0.0208568	Preprocessor1_Model02
9	20	rsq	standard	0.7899044	10	0.0208568	Preprocessor1_Model08
14	35	rsq	standard	0.7891708	10	0.0211144	Preprocessor1_Model06
13	32	rsq	standard	0.7891708	10	0.0211144	Preprocessor1_Model10
11	16	rsq	standard	0.7888970	10	0.0205838	Preprocessor1_Model05

Para que se maximice el $R^2(rsq)$, la profundidad máxima del árbol deberá ser de 6 y el número mínimo de datos en un nodo que se requieren para que el nodo se divida más deberá ser de 23.

4.7. Evaluación

Hasta este momento, sólo se ha trabajado sobre la muestra de entrenamiento, y ahora que ya tenemos cuáles son los hiperparámetros que nos proporciona los mejores modelos, vamos a utilizar la muestra test para evaluarlos y determinar cuál es el modelo que mejor predice el *valor de mercado* de los futbolistas (mayor R^2, menor rmse).

Se muestra el código de la evaluación del modelo de árbol de decisión con mejor R^2.

```
tree_best_rsq <-
  tree_results %>%
    select_best(metric = "rsq")
```

Creamos un nuevo workflow que incluya el modelo seleccionado.

```
tree_wfl_final_rsq <-
  tree_wf %>%
    finalize_workflow(tree_best_rsq)
```

Ahora, gracias al comando *last_fit* emulamos el proceso donde, después de determinar el mejor modelo, se necesita el ajuste final en todo el conjunto de entrenamiento y luego se evalúa en el conjunto de prueba.

```
tree_test_results_rsq <-
  tree_wfl_final_rsq %>%
    last_fit(split = jugadores_split_estratos)
```

Obtenemos las métricas procedentes de la muestra test.

```
tree_final_rsq <- tree_test_results_rsq %>%
  collect_metrics()
tree_final_rsq %>%
  kable(caption = "Métricas", booktabs = T,
        longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 12,
                repeat_header_text = "(continúa)")
```

Tabla 4.3: Métricas

.metric	.estimator	.estimate	.config
rmse	standard	4.955071e+06	Preprocessor1_Model1
rsq	standard	7.834996e-01	Preprocessor1_Model1

Obtenemos un modelo con un R^2 relativamente alto, de 0.7834996, y una raíz del error cuadrático medio de 4.9550709 millones, que como es un valor medio, para los valores muy altos es un buen valor pero tal vez para los bajos, no tanto. Para solucionar esta incógnita, vamos a dividir la muestra test en diferentes estratos, en función del *valor de mercado*, para determinar como es la raíz del error cuadrático medio en cada intervalo.

Hay que destacar que la raíz del error cuadrático medio se ha dividido entre 1 millón, por lo tanto, ahora se expresa en millones de libras, para que sea más fácil su interpretación.

Tabla 4.4: Rmse por estratos

Estratos	rmse
<100k	0.8997151
<500k	1.0771552
500k - 1.5M	1.4966236
1.5M - 5M	3.5286302
<1.5	1.3006198
<5M	2.3291609
>5M	9.1585521
5M - 20M	5.0632409
>20M	15.5395265
5M - 30M	6.6309978
>30M	18.5280117
>40M	22.0343878
20 - 60M	12.2776095
>60M	28.1684477

Tabla 4.4: Rmse por estratos (continúa)

Estratos	rmse
>70M	44.3984917

Dependiendo del intervalo, la calidad de las predicciones varía claramente, ya que los que tienen valores mayores de 5 millones tienen errores relativamente menores que los que tienen un *valor de mercado* inferior, exceptuando aquellos que tienen un *valor de mercado* mayor de 70 millones, que sólo son tres. Este proceso se realiza a cada uno de los modelos y al final se mostrará una tabla similar correspondiente al modelo elegido como el mejor.

4.8. Redes neuronales

La predicción del *valor de mercado* mediante redes neuronales se va a realizar de forma diferente, ya que no se van a usar las funciones del paquete *tidymodels*, si no el paquete *keras*.

Primero dividimos la muestra en muestra entrenamiento y test.

Vamos a explicar cómo se define la muestra entrenamiento y test. Se define una receta parecida a la utilizada antes, ya que se recomienda normalizar las características que usan diferentes escalas y rangos. A continuación, guardamos la muestra como una matriz con las transformaciones realizadas y en un vector la variable objetivo.

```
receta_train <- recipe(market_value_in_gbp ~. ,
                       data = jugadores_con_valor) %>%
  step_dummy(all_nominal(), one_hot = TRUE) %>%
  step_corr(all_numeric(),-all_outcomes()) %>%
  step_center(all_numeric(),-all_outcomes()) %>%
  step_scale(all_numeric(),-all_outcomes()) %>%
  prep()


jugadores_train <- bake(receta_train, new_data = NULL)
train_labels <- as.vector(as.matrix(jugadores_train[,11]))
jugadores_train <- as.matrix(jugadores_train[,-11])

jugadores_test <- receta_train %>%
                        bake(testing(jugadores_split_estratos))
test_labels <- jugadores_test[,11]
test_labels <- as.vector(as.matrix(test_labels))
jugadores_test <- jugadores_test[,-11]
jugadores_test <- as.matrix(jugadores_test)
```

Una vez hemos se ha dividido la muestra original, se define y compila el modelo. El modelo construido se trata una red de tres capas interiores, la primera de ella compuesta por 64 neuronas, la segunda por 32 neuronas y la tercera por 16. Este modelo es el que mejor métricas obtenía de los que se han probado, como son los de las redes de una

y dos capas y los que incluyen **dropout** (comentado en el código). El **dropout** es un método que desactiva un número de neuronas de una red neuronal de forma aleatoria. En cada iteración de la red neuronal **dropout** desactivará algunas y no se toman en cuenta para el forwardpropagation, ni para el backwardpropagation lo que obliga a las neuronas cercanas a no depender tanto de las neuronas desactivadas. Este método ayuda a reducir el overfitting. Sin embargo, para los datos del estudio, aplicar el **dropout** empeoraba los resultados.

Hay que tener en cuenta que el modelo está definido para el error cuadrático medio, no para su raíz, por tanto basta aplicarsela al resultado de las métricas para obtener lo que se desea. Además, en este caso los gráficos serán representados para el error cuadrático medio y no para su raíz.

```
build_model <- function() {

  model <- keras_model_sequential() %>%
    layer_dense(units = 64, activation = "relu",
                input_shape = dim(jugadores_train)[2]) %>%
#    layer_dropout(0.6) %>%
    layer_dense(units = 32, activation = "relu") %>%
#    layer_dropout(0.4) %>%
    layer_dense(units = 16, activation = "relu") %>%
#    layer_dropout(0.2) %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_squared_error")
  )

  model
}

model <- build_model()
model %>% summary()
```

```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  dense_3 (Dense)                    (None, 64)                      1600
##  dense_2 (Dense)                    (None, 32)                      2080
##  dense_1 (Dense)                    (None, 16)                      528
##  dense (Dense)                      (None, 1)                       17
## =========================================================================
## Total params: 4,225
## Trainable params: 4,225
## Non-trainable params: 0
## _________________________________________________________________________
```

El modelo está entrenado para 500 epochs, registrando la precisión del entrenamiento y la validación.

En la figura 4.1 se observa cómo va evolucionando el error cuadrático medio a lo largo de la ephocs y se tiene que la muestra de validación tiene una mejor evolución que la de entrenamiento, por lo que se está cumpliendo el objetivo del modelo.

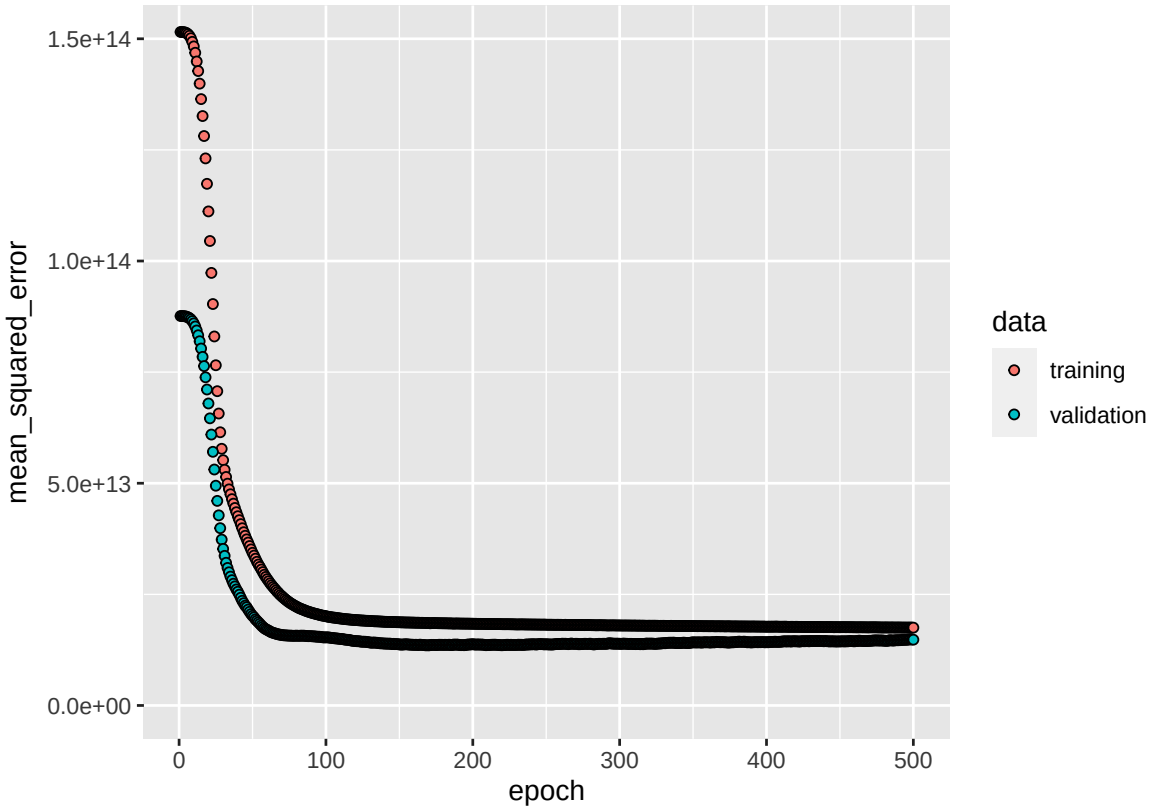


Figura 4.1: Evolución del MSE en entrenamiento y test

En el gráfico de la figura 4.2 nos muestra que el error cuadrático medio se estabiliza en las 200 ephocs.

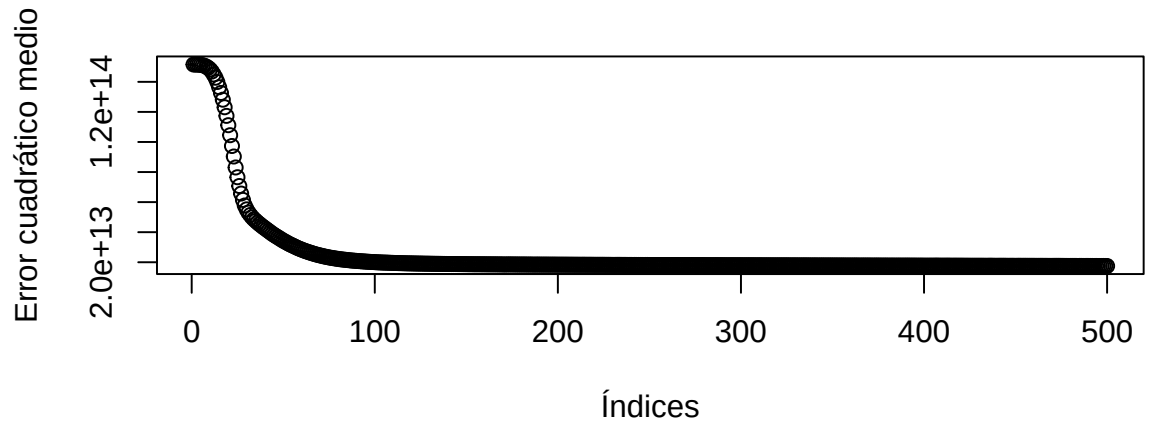


Figura 4.2: Evolución del MSE

Ahora, se realiza la predicción del *valor de mercado* y obtenemos las métricas del modelo.

```
test_predictions <- model %>% predict(jugadores_test)
```

Por último, se obtienen las siguientes métricas para el modelo.

- $R^2 = 0.8381405$.
- $RMSE = 3.999835$.

Se concluye que el modelo obtenido es fiable para la predicción del *valor de mercado* de los jugadores.

4.9. Tabla resumen

El proceso anterior al de redes neuronales se aplica a todos los modelos descritos y se obtienen los siguientes resultados.

Tabla 4.5: Métricas de los modelos

Método	rsq	rmse
Árbol de decisión(rmse)	0.783499644908447	4.95507093460659
Árbol de decisión(rsq)	0.783499644908447	4.95507093460659
Random Forest(rmse y rsq)	0.87926958816451	3.74752026340686
knn(rmse)	0.755629650524639	5.58433286111724
knn(rsq)	0.771651705233554	5.6722532912086
Regresión lineal	0.784607720334334	4.83078946968983
Red neuronal	0.838140547253528	3.99983496875559

Con estos resultados, se puede concluir que el modelo con mejor métricas es el de bosques aleatorios (Random Forest), con un R^2 cercano a 0.9 (bastante alto) y una raíz del error cuadrático medio menor que 4 millones, seguido muy de cerca por el método de redes neuronales que proporciona también buenos resultados. Los hiperparámetros que nos brindan estos resultados son:

- Se muestrearán aleatoriamente 15 predictores en cada división al crear los modelos de árbol (mtry = 15).
- El número mínimo de datos en un nodo que se requieren para que el nodo se divida más es de 2 (min_n = 2).

Vamos a comprobar cómo varía la raíz del error cuadrático medio según los estratos definidos antes para el modelo de árbol de decisión.

Tabla 4.6: Rmse por estratos Random Forest

Estratos	rmse
<100k	0.3275765
<500k	0.3637628

Tabla 4.6: Rmse por estratos Random Forest (continúa)

Estratos	rmse
500k - 1.5M	0.8527389
1.5M - 5M	2.0679883
<1.5	0.6519829
<5M	1.3272058
>5M	7.2207303
5M - 20M	3.8278504
>20M	12.3924584
5M - 30M	4.6005661
>30M	15.9206859
>40M	18.5647696
20 - 60M	8.9386702
>60M	24.5306268
>70M	41.8442275

Los resultados que muestra la tabla indica que las predicciones con este modelo son mucho más alentadoras que las del anterior, sobre todo en los valores más pequeños. Por ejemplo, la raíz del error cuadrático medio para valores menores que 1.5 millones es cercano al medio millón y para valores menores de 5 millones cercano al millón. También se puede destacar que, para valores entre 5 y 30 millones, el error es de 4.5 millones (se toma este intervalo de valores para que haya un número representativo de futbolistas). Todo esto nos indica que hemos obtenido un modelo que predice con ciertas garantías el *valor de mercado*, que es el objetivo principal del estudio.

Capítulo 5

Conclusiones

En este último apartado se van a incluir las conclusiones más importantes que se pueden obtener del estudio realizado.

Antes de citar las conclusiones finales, hay que recordar que se ha afrontado un problema real de una gran dificultad e interés socioeconómico, teniendo en cuenta tanto el interés como el dinero que mueve el mundo del fútbol. A partir de ahí, también hay que recordar que se ha hecho un proceso exhaustivo de búsqueda de fuentes a nuestro alcance (fuentes de datos públicas), limpieza, tratamiento, análisis exploratorio y un proceso organizado de modelización, con los ecosistemas de paquetes *tidyverse* y *tidymodels* y que se han alcanzado resultados bastante buenos.

Del análisis descriptivo de la variable objetivo *valor de mercado* se puede concluir que el *valor de mercado* de los futbolistas suele ser bajo, en la mayoría de los casos por debajo de los dos millones de libras. De aquí, se puede notificar que los traspasos tan caros que están siempre en las portadas de prensa son la minoría realmente. También hay que tener en cuenta que la diferencia entre los valores más altos y los más bajos es muy grande; de hecho, estando más del 80 % de los jugadores por debajo de los dos millones de valor, la media de todas ellas es de algo más de cinco millones. Por lo tanto, podemos concluir que la variable objetivo se caracteriza por la acumulación de muchos valores en un rango de cero a dos millones y la gran diferencia entre los mayores y menores.

Con respecto a las variables predictoras cuantitativas, se puede concluir que el *valor de mercado* depende mucho del valor más alto que haya tenido el futbolista en el pasado. Este valor ha sido tomado de Transfermarkt, pero una vez esté implementada esta aplicación, este valor puede ser sustituido por otro que hayamos calculado y que haya superado al original, por lo que es una variable fácilmente actualizable. Con respecto a las demás variables, que dependen de la actuación del futbolista en sí, podemos obtener diferentes conclusiones.

Si un equipo está en disposición de vender un futbolista, tendría que tratar de darle minutos de juego, lo que provocaría que marcara más goles y diera más asistencias (estas variables están correlacionadas positivamente). Esto llevaría a que el *valor de mercado* suba y podrían pedir más dinero a la hora de tasarlo para un traspaso. Es verdad que jugar más provocaría que le marcaran más goles por minuto, pero esta variable está menos relacionada con el *valor de mercado*, por lo que es preferible darle minutos si la finalidad es el traspaso de un jugador determinado.

Además, si el club se encuentra en una situación económica difícil quizás tendría que vender a más delanteros, ya que su *valor de mercado* suele ser mayor; por tanto, tendrían que pagar más por el traspaso.

Por otro lado, si estamos en disposición de comprar futbolistas y tenemos un presupuesto limitado (no somos el PSG y por lo que se observa en las noticias parece tener dinero ilimitado y no estar sujeto a las llamadas reglas del fair-play financiero) tendríamos que tener en cuenta lo siguiente: comprar un jugador menor de 20 años, o mayor de 27, que no juegue en una de las cinco grandes ligas y que no esté disfrutando de muchos minutos. Este último parámetro es más difícil de utilizar, ya que está más en el ojo clínico del director y en la confianza que ponga en el mismo jugador, ya que siempre es difícil añadir a tu equipo a un jugador con pocos minutos.

Una estrategia de mercado que se puede extraer del estudio sería la de comprar jugadores sudamericanos a los clubes de allí directamente, antes de que lleguen a Europa, y luego venderlos al tiempo, ya que el valor de los jugadores sudamericanos tienen la mayor media del *valor de mercado*.

Sobre los modelos planteados, se puede apuntar que todos ellos menos el de redes neuronales (rechazamos este modelo, aunque quedaría para un trabajo futuro un análisis más profundo específico de estas redes para buscar configuraciones más prometedoras), predicen los valores de mercado con ciertas garantías, todos con valores de R^2 por encima de 0.75, teniendo en cuanta que alguno de ellos tienen una raíz del error cuadrático medio alta, lo que provoca que no sean tan prometedoras estas predicciones. El modelo de regresión lineal se va a rechazar porque da lugar a predicciones negativas.

De los tres modelos restantes, concluimos que el mejor modelo es Random Forest, con un R^2 cercano a 0.9 y un error cuadrático medio menor que 4 millones de libras, con los hiperparámetros calculados por el tunning.

Por lo tanto, para los nuevos datos o para darle a estos datos nuestra valoración de mercado propia, se utilizará un modelo de bosques aleatorios (Random Forest) donde se muestrearán aleatoriamente 15 predictores en cada división al crear los modelos de árbol (mtry = 15) y el número mínimo de datos en un nodo que se requieren para que el nodo se divida más es de 2 (min_n = 2). Este es nuestro modelo final.

Como se acaba de comentar, no sólo se va a utilizar para predecir valores nuevos, si no para tener una valoración propia de los jugadores, distinta a la de Transfermarkt. Además, si se añaden nuevas variables al la base de datos, este modelo seguiría siendo estable y se podría usar.

Para finalizar y concluir el estudio, se debe destacar que todo lo que hemos analizado y calculado puede ser utilizado en la práctica por una dirección técnica de un club de élite, ya que los resultados provienen de datos reales y el desarrollo se ha realizado con la mayor exactitud posible.

Por lo tanto, con los medios económicos, de infraestructura y tecnológicos de los clubes profesionales se podría avanzar aún más en este estudio, añadiendo variables y conjunto de datos para ver cómo fluctúa el *valor de mercado* a lo largo del tiempo. Además, este problema del *valor de mercado* a lo largo del tiempo, se soluciona una vez hemos implementado el modelo de predicción, ya que si actualizamos las variables, podemos actualizar el *valor de mercado* y poder implementar nuevos modelos y estudios a partir de este.

Apéndice A

Códigos

En este apéndice se incluye todostodo el desarrollo de software diseñado para la realización del estudio, muchos de ellos no se han incluido en la explicación y se reverenciarán en este apartado.

A.1. Importación

Primero, se incluye el código de la importación de los conjuntos de datos necesarios para el estudio.

```
jugadores <-read_csv("Datos/jugadores.csv",
  col_types = cols(
    last_season = col_factor(),
    current_club_id = col_integer(),
    country_of_birth = col_factor(),
    country_of_citizenship = col_factor(),
    position = col_factor(),
    sub_position = col_factor()
  ))
```

```
jugadores[c(2,555,7777,8888,9993,10000),c(8,9,10,11,12,13,14)] %>%
  head() %>%
  kable(caption = "\\label{tabla72}Jugadores (segunda parte)",
        booktabs = T,longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 6,
                repeat_header_text = "(continúa)")
```

```
clubs <- read_csv("Datos/clubs.csv")
```

```
clubs[c(1,2,3,5,6,9),1:5] %>%
  kable(caption = "\\label{tabla01}Clubs", booktabs = T,
        longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
```

```
                position = "center",
                full_width = FALSE,
                font_size = 9,
                repeat_header_text = "(continúa)")
```

A continuación, se incluye cómo se importan conjuntos de datos sobre los que se han realizado algunas modificaciones en los nombres de los elementos de las columnas.

```
competiciones <- read_csv("Datos/competiciones.csv")

separacion <- str_split(competiciones$type, pattern = "_")


# Aquí tenemos un vector con los nombres de
# los tipos de competiciones.

uniones <- c()
for (i in 1:length(competiciones$type)) {
  uniones[i] <- str_c(separacion[[i]], collapse =" ")
}



# Lo convertimos a frase para que la primera esté en mayúsculas.

competiciones$type <- str_to_title(uniones)

# Vamos a realizar el mismo procedimiento con la varaible
# "name" que se refiere al nombre de la competición.


separacion <- str_split(competiciones$name, pattern = "-")



# Aquí tenemos un vector con los nombres de
# los tipos de competiciones.

uniones <- c()
for (i in 1:length(competiciones$name)) {
  uniones[i] <- str_c(separacion[[i]], collapse =" ")
}

competiciones$name <- uniones

competiciones$name <- str_to_title(competiciones$name)
competiciones$confederation <- str_to_title(competiciones$confederation)
```

```
competiciones$country_name <- str_replace(
  str_replace_na(competiciones$country_name),
  pattern="NA", replacement="Europe")


competiciones$type <- as.factor(competiciones$type)
competiciones$country_name <- as.factor(competiciones$country_name)
competiciones$confederation <- as.factor(competiciones$confederation)
```

```
competiciones[,1:7] %>%
  head() %>%
  kable(caption = "\\label{tabla02}Competiciones",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options =
                  c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 6,
                repeat_header_text = "(continúa)")
```

```
ligas <- read_csv("Datos/ligas.csv")

separacion <- str_split(ligas$name, pattern = "-")


# Aquí tenemos un vector con los nombres
# de los tipos de competiciones.

uniones <- c()
for (i in 1:length(ligas$name)) {
  uniones[i] <- str_c(separacion[[i]], collapse =" ")
}

ligas$name <- uniones


ligas$name <- str_to_title(ligas$name)
ligas$confederation <- str_to_title(ligas$confederation)


ligas$name <- as.factor(ligas$name)
ligas$confederation <- as.factor(ligas$confederation)
```

```
ligas %>%
  kable(caption = "\\label{tabla03}Ligas", booktabs = T,
        longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
```

```
                full_width = FALSE,
                font_size = 10,
                repeat_header_text = "(continúa)")
```

```
aparaciones <- read_csv("Datos/apariciones.csv")
```

```
aparaciones %>%
  head() %>%
  kable(caption = "\\label{tabla04}Apariciones",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 5,
                repeat_header_text = "(continúa)")
```

```
partidos <- read_csv("Datos/partidos.csv")

partidos <- read_csv("Datos/partidos.csv",
    col_types = cols(
      round = col_factor(),
                    ))
```

```
partidos[6:9,1:9] %>%
  head() %>%
  kable(caption = "\\label{tabla55}Partidos", booktabs = T,
        longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 5,
                repeat_header_text = "(continúa)")
```

```
continent <- read_csv(
 "https://raw.githubusercontent.com/dbouquin/IS_608/master
 /NanosatDB_munging/Countries-Continents.csv",
 col_names = c("continent","country_of_citizenship"))
```

```
continent[c(8,77,130,155),] %>%
  head() %>%
  kable(caption = "Continente", booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 10,
                repeat_header_text = "(continúa)")
```

A.2. Tratamiento

A continuación, se muestra el código donde se crea el conjunto de datos sobre el que se ha realizado el estudio.

A.2.1. Variable edad

Código de la creación de la variable **age**.

```
jugadores$foot <- str_replace(str_replace_na(jugadores$foot),
                              pattern="NA", replacement="No Data")

jugadores$foot <- as.factor(jugadores$foot)

jugadores <- jugadores %>%
  mutate(age = 2021 - as.numeric(format(jugadores$date_of_birth,'%Y')))
```

```
jugadores_con_valor <- jugadores %>%
  select( pretty_name,player_id,current_club_id,last_season,
          country_of_birth, country_of_citizenship, position,
          sub_position,foot, height_in_cm,market_value_in_gbp,
          highest_market_value_in_gbp,age) %>%
  drop_na(market_value_in_gbp) %>%
  drop_na(age) %>%
  drop_na(country_of_birth) %>%
  filter(last_season == 2021) %>%
  filter(height_in_cm > 0) %>%
  filter(last_season == 2021) %>%
  filter(position != 0)
```

A.2.2. Continente

Código de la creación de la variable **Continent**.

```
conts <- distinct(jugadores_con_valor %>%
  select(country_of_citizenship))

datos_continent <- conts %>%
  left_join(continent) %>%
  arrange(continent,country_of_citizenship)

datos_continent$continent[c(111,114,115,117,118,121,124,125,127,128,131)]
    <- "Europe"
datos_continent$continent[c(112,113,116,129)] <- "Africa"
datos_continent$continent[c(119,120,122,130)] <- "North America"
datos_continent$continent[123] <- "Oceania"
datos_continent$continent[126] <- "Asia"
```

```
# Comprobación
jugadores_con_valor %>%
  inner_join(datos_continent) %>%
  nrow() == nrow(jugadores_con_valor)

continent <- datos_continent
```

Código de la creación de la variable **Continent**.

```
jugadores_con_valor <- jugadores_con_valor %>%
  inner_join(continent)
```

A.2.3. Media de goles, asistencias, minutos, amarillas y rojas

Código de la creación de la variables **goals_average** (media de Goles), **assists_average** (media de Asistencias), **min_played_average** (media de Min.Jug), **yellow_cards_average** (media de amarillas) y **red_cards_average** (media de rojas).

```
estats_desde_2017 <- jugadores_con_valor %>%
  inner_join(aparaciones) %>%
  inner_join(partidos) %>%
  group_by(player_id,season,pretty_name) %>%
  summarise(Goles = sum(goals),
            Asistencias = sum(assists),
            `Minutos Jugados` = sum(minutes_played),
            `Tarjetas amarillas` = sum(yellow_cards),
            `Tarjetas rojas` = sum(red_cards)) %>%
  filter(season > 2017) %>%
  inner_join(jugadores_con_valor[,c(2,11)])

statisticss_jugadores <- estats_desde_2017 %>%
  group_by(player_id,pretty_name) %>%
  summarise(goals_average = mean(Goles),
            assists_average = mean(Asistencias),
            min_played_average = mean(`Minutos Jugados`),
            yellow_cards_average = mean(`Tarjetas amarillas`),
            red_cards_average = mean(`Tarjetas rojas`),
            market_value_in_gbp = mean(market_value_in_gbp))
```

```
jugadores_con_valor <- statisticss_jugadores[,-c(2,8)] %>%
  inner_join(jugadores_con_valor, by = 'player_id')
```

```
estats_antes_2017 <- jugadores_con_valor %>%
  inner_join(aparaciones) %>%
  inner_join(partidos) %>%
  group_by(player_id,season,pretty_name) %>%
  summarise(Goles = sum(goals),
            Asistencias = sum(assists),
            `Minutos Jugados` = sum(minutes_played),
```

```
            `Tarjetas amarillas` = sum(yellow_cards),
            `Tarjetas rojas` = sum(red_cards)) %>%
  filter(season < 2019) %>%
  inner_join(jugadores_con_valor[,c(2,11)])

statisticss_jugadores_a2017 <- estats_antes_2017 %>%
  group_by(player_id,pretty_name) %>%
  summarise(goals_average = mean(Goles),
            assists_average = mean(Asistencias),
            min_played_average = mean(`Minutos Jugados`),
            yellow_cards_average = mean(`Tarjetas amarillas`),
            red_cards_average = mean(`Tarjetas rojas`),
            market_value_in_gbp = mean(market_value_in_gbp))
}
jugadores_con_valor_a2017 <- statisticss_jugadores_a2017[,-c(2,8)] %>%
  inner_join(jugadores_con_valor, by = 'player_id')
cor(jugadores_con_valor_a2017$assists_average,jugadores_con_valor_a2017$market_value_
```

A.2.4. Variable propia estadística

A continuación, se muestra todo el proceso de creación de las variables **Statistics** y **statistics.scale**, que es una variable propia como se explica en el documento.

```
### Puntos por partido de cada equipo

puntajeLocal <- function(a,b) {
  puntos_local <- c()
  for (i in 1:length(a)) {
  if (a[i] > b[i]) {
    puntos_local[i] <- 3}
   else {
     if
    (a[i] < b[i]) {
    puntos_local[i] <- 0}
   else puntos_local[i] <- 1
   }
  }
  puntos_local
}

puntajeVisitante <- function(a,b) {
  puntos_visitante <- c()
  for (i in 1:length(a)) {
  if (a[i] < b[i]) {
    puntos_visitante[i] <- 3}
   else {
     if
    (a[i] > b[i]) {
```

```
    puntos_visitante[i] <- 0}
   else puntos_visitante[i] <- 1
   }
  }
  puntos_visitante
}

partidos <- partidos %>%
  mutate(
    puntos_local = puntajeLocal(home_club_goals,
                                away_club_goals),
    puntos_visitante = puntajeVisitante(home_club_goals,
                                        away_club_goals),
    goles_encajados_local = away_club_goals,
    goles_encajados_visitante = home_club_goals
  )


colnames(partidos)[2] <- "competition_id"

Locales <- partidos %>%
  inner_join(competiciones,by = "competition_id") %>%
  filter(type == "First Tier") %>%
  group_by(competition_id,season,home_club_id) %>%
  summarise(Puntos_Local_Temporada = sum(puntos_local))
colnames(Locales)[3] <- "club_id"


Visitantes <- partidos %>%
  inner_join(competiciones,by = "competition_id") %>%
  filter(type == "First Tier") %>%
  group_by(competition_id,season,away_club_id) %>%
  summarise(Puntos_Visitante_Temporada = sum(puntos_visitante))
colnames(Visitantes)[3] <- "club_id"

Clasificaciones <- Locales %>%
  inner_join(Visitantes,
             by = c("club_id","season","competition_id")) %>%
  mutate(Puntos = Puntos_Local_Temporada+Puntos_Visitante_Temporada)


Clasificaciones <- Clasificaciones %>%
  inner_join(clubs, by = "club_id")

Clasificaciones <- Clasificaciones %>%
  inner_join(competiciones, by = "competition_id")
```

```
Clasificaciones <- Clasificaciones %>%
  select(c("season","Puntos","pretty_name","name.y","country_name"),
         "club_id")
Clasificaciones %>%
  filter(name.y == "Liga Portugal Bwin")

Clasificacion <- function (Liga,Temporada) {
  Clasificaciones %>%
    filter(season == Temporada) %>%
    filter(name.y == Liga) %>%
    arrange(desc(Puntos)) %>%
    mutate(Posicion = 1:length(club_id),
           PosicionInversa = length(club_id):1)
}

Clasificacion("Liga Portugal Bwin",2015)


ClasificacionesTotales <- function(Liga) {
  clasificacionesT <- rbind(
    Clasificacion(Liga,2014),
    Clasificacion(Liga,2015),
    Clasificacion(Liga,2016),
    Clasificacion(Liga,2017),
    Clasificacion(Liga,2018),
    Clasificacion(Liga,2019),
    Clasificacion(Liga,2020))

  clasificacionesT %>%
    group_by(club_id,pretty_name) %>%
    summarise(MediaPuntosCompleta = mean(Puntos),
              MediaPuntosReal = sum(Puntos)/7,
              SumaPuntos = sum(Puntos),
              MediaClasificatoria = mean(PosicionInversa),
              SumaClasifiación = sum(PosicionInversa)) %>%
    mutate(name = Liga)
}
ClasificacionTotal <- tibble()
LigasNombres <- as.character(ligas$name)
LigasNombres[5] <- "Liga Portugal Bwin"

for (i in 1:nrow(ligas)) {
  ClasificacionTotal <-
    rbind(ClasificacionTotal,ClasificacionesTotales(LigasNombres[i]))
}

colnames(ClasificacionTotal)[1] <- "current_club_id"
```

```
ClasificacionTotal$current_club_id <-
  as.integer(ClasificacionTotal$current_club_id)

jugadores_con_valor <- jugadores_con_valor %>%
  inner_join(ClasificacionTotal, by = "current_club_id")

jugadores_con_valor <-
  jugadores_con_valor[!duplicated(jugadores_con_valor), ]

LigasNombres <- as_tibble(LigasNombres)
LigasNombres <- LigasNombres %>%
  mutate(Indices = c(2,1.5,1.5,2,1.5,2,2,1.5,1.5,1.5,2,1.5,1.5,1))


colnames(LigasNombres)[1] = "name"

jugadores_con_valor <- jugadores_con_valor %>%
  inner_join(LigasNombres, by = "name")

jugadores_con_valor <-
  jugadores_con_valor[!duplicated(jugadores_con_valor), ]

jugadores_con_valor <- jugadores_con_valor %>%
  mutate(statistics = (SumaPuntos + SumaClasifiación)*Indices)
jugadores_con_valor

statistics.scale <- scale(jugadores_con_valor[,c(23,25)],center=T,scale=T)
statistics.scale <- as_tibble(statistics.scale)
statistics.scale <- statistics.scale %>%
  mutate(statistics_scale =
           ((SumaPuntos + SumaClasifiación))*jugadores_con_valor$Indices)
head(statistics.scale)
colnames(statistics.scale) <- c("SumaPuntos_escalasa",
                                "SumaClasifiación_scale",
                                "statistics_scale")
head(statistics.scale)

jugadores_con_valor <-  jugadores_con_valor %>%
  bind_cols(statistics.scale)
jugadores_con_valor
```

A.2.5. Goles encajados

Se muestra como se definen las variables **goals_conceded_season_average (media de goles encajados)** y **goals_conceded_minute (media de goles encajados por minuto)**.

```
## Valor de mercado de los fútbolistas en función de los goles encajados

partidos_locales <- partidos %>%
  select(c("game_id","competition_id",
           "season","home_club_id","goles_encajados_local"))
colnames(partidos_locales)[4] <- "player_club_id"

partidos_visitantes <- partidos %>%
  select(c("game_id","competition_id","season","away_club_id",
           "goles_encajados_visitante"))
colnames(partidos_visitantes)[4] <- "player_club_id"

partidos_locales_goles <- partidos_locales %>%
  inner_join(aparaciones, by = c("game_id","player_club_id")) %>%
  filter(season > 2017) %>%
  filter(minutes_played > 45) %>%
  group_by(player_id,season) %>%
  summarise(goles_encajados_temporada_local = sum (goles_encajados_local))

partidos_visitantes_goles <- partidos_visitantes %>%
  inner_join(aparaciones, by = c("game_id","player_club_id")) %>%
  filter(season > 2017) %>%
  filter(minutes_played > 45) %>%
  group_by(player_id,season) %>%
  summarise(
    goles_encajados_temporada_visitante = sum (goles_encajados_visitante))

goles_encajados <- partidos_locales_goles %>%
  inner_join(partidos_visitantes_goles, by= c("player_id","season")) %>%
  mutate(
    goles_encajados_temporada =
      goles_encajados_temporada_local +
      goles_encajados_temporada_visitante) %>%
  group_by(player_id) %>%
  summarise(goals_conceded_season_average =
              mean(goles_encajados_temporada))

goles_encajados

jugadores_con_valor <- jugadores_con_valor %>%
  inner_join(goles_encajados, by = "player_id")

jugadores_con_valor <-
  jugadores_con_valor[!duplicated(jugadores_con_valor),]

#### Goles encajados por minuto jugado
```

```
jugadores_con_valor <- jugadores_con_valor %>%
  mutate(goals_conceded_minute =
           goals_conceded_season_average/min_played_average)
```

A.2.6. Conjunto de datos final

Se termina de definir el conjunto de datos final para el desarrollo del proyecto.

```
jugadores_con_valor <- jugadores_con_valor %>%
  select(player_id,pretty_name.x,market_value_in_gbp,goals_average,
         assists_average,min_played_average,yellow_cards_average,
         red_cards_average,position,sub_position,foot,height_in_cm,
         country_of_birth,highest_market_value_in_gbp,age,statistics,
         statistics_scale,goals_conceded_season_average,
         goals_conceded_minute,name,continent)
```

```
jugadores_con_valor <- jugadores_con_valor %>%
  droplevels()
jugadores_con_valor$continent <- as.factor(jugadores_con_valor$continent)
```

```
jugadores_con_valor[,1:7] %>%
  head() %>%
  kable(caption = "\\label{tabla05}Jugadores con valor (primera parte)",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 5,
                repeat_header_text = "(continúa)")
```

```
jugadores_con_valor[,8:14] %>%
  head() %>%
  kable(caption = "\\label{tabla06}Jugadores con valor (segunda parte)",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 6,
                repeat_header_text = "(continúa)")
```

```
jugadores_con_valor[,15:20] %>%
  head() %>%
  kable(caption = "\\label{tabla07}Jugadores con valor (tercera parte)",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 6,
```

```
                repeat_header_text = "(continúa)")
```

A.3. Análisis

Una vez obtenido el conjunto de datos final con las variables necesarias para el estudio, en este apartado se muestra el código con el que se realiza el análisis descriptivo de las variables que van a ser útiles para predecir el *valor de mercado* de los futbolistas.

En este apartado también se muestran los códigos de otras formas de representación que no se han incluido en el estudio.

A.3.1. Descripción de las variable objetivo

A continuación, se muestra el código donde se definen el diagrama de cajas y bigotes, el diagrama de violín, el histograma, la estimación de densidad y el gráfico Q-Q de la variable objetivo, también junto a la transformación logarítmica.

```
# Objeto gráfico base para los 4 gráficos siguientes
gb <- jugadores_con_valor %>%
  ggplot(aes(market_value_in_gbp)) +
  xlab('Valor de mercado') + ylab('')

# Diagrama de caja y bigotes:
g0 <- gb +
  geom_boxplot(aes(y=1),width=.5) +
  geom_jitter(aes(y=1),width=0.1, height = .25, size = 0.5, alpha = 0.3) +
  scale_y_continuous(breaks=NULL)+
  ylab('')

# Diagrama de violín:
g0b <- gb +
  geom_violin(aes(y=1),width=.5) +
  scale_y_continuous(breaks=NULL)+
  ylab('')

## No utilizado.
g1 <- gb +
  geom_dotplot(binwidth=.05) +
  ylim(0, 20) +
  coord_fixed(ratio=.05)

# Histograma:
g2 <- gb +
  geom_histogram(bins=30)

# Gráfico de densidad:
g3 <- gb +
  geom_density()
```

```
g4 <- faithful %>%
  ggplot(aes(sample=eruptions)) +
  geom_qq() +
  xlab('Cuantiles de una N(0, 1)')
```

```
ggarrange(g0, g0b,g2,g4,
          labels = c("Caja y bigotes", "Violín" ),
          ncol = 2, nrow = 2)
```

```
# Objeto gráfico base para los 4 gráficos siguientes
gb <- jugadores_con_valor %>%
  ggplot(aes(market_value_in_gbp)) +
  xlab('log(Valor de mercado)') + ylab('') +
  scale_x_log10()

# Diagrama de caja y bigotes:
g0 <- gb +
  geom_boxplot(aes(y=1),width=.5) +
  geom_jitter(aes(y=1),width=0.1, height = .25, size = 0.5, alpha = 0.3) +
  scale_y_continuous(breaks=NULL)+
  ylab('')

# Diagrama de violín:
g0b <- gb +
  geom_violin(aes(y=1),width=.5) +
  scale_y_continuous(breaks=NULL)+
  ylab('')

## No utilizado.
g1 <- gb +
  geom_dotplot(binwidth=.05) +
  ylim(0, 20) +
  coord_fixed(ratio=.05)

# Histograma:
g2 <- gb +
  geom_histogram(bins=30)

# Gráfico de densidad:
g3 <- gb +
  geom_density()

g4 <- faithful %>%
  ggplot(aes(sample=eruptions)) +
  geom_qq() +
  xlab('Cuantiles de una N(0, 1)')
```

```
ggarrange(g0, g0b,g3,g4,
          labels = c("", "" ),
          ncol = 2, nrow = 2)
```

A.3.2. Estudio de las variables cuantitativas

A continuación, se describe cómo se ha programado el estudio de las variables cuantitativas del proyecto.

A.3.2.1. Correlaciones

Código sobre las correlaciones entre las variables.

```
ggpairs(jugadores_con_valor[,c(3,4,5,6,7,8)]) +
    theme(axis.text.x = element_text(angle = 45, hjust=1))
```

```
ggpairs(jugadores_con_valor[,c(3,12,14,15,17,18,19)]) +
    theme(axis.text.x = element_text(angle = 45, hjust=1))
```

```
porteros <- jugadores_con_valor %>%
  filter(position == "Goalkeeper")

defensas <- jugadores_con_valor %>%
  filter(position == "Defender")

centrocampistas <- jugadores_con_valor %>%
  filter(position == "Midfield")

delanteros <- jugadores_con_valor %>%
  filter(position == "Attack")



data.frame(
  `Variable` = names(jugadores_con_valor[c(4,5,6,7,8,12,14,15,17,18,19)]),

  Correlacion = as.vector(
    cor(jugadores_con_valor[,c(3,4,5,6,7,8,12,14,15,17,18,19)])[1,-1]),

  Porteros =  as.vector(
    cor(porteros[,c(3,4,5,6,7,8,12,14,15,17,18,19)])[1,-1]),

  Defensas =  as.vector(
    cor(defensas[,c(3,4,5,6,7,8,12,14,15,17,18,19)])[1,-1]),

  Centrocampistas =  as.vector(
    cor(centrocampistas[,c(3,4,5,6,7,8,12,14,15,17,18,19)])[1,-1]),

  Delanteros =  as.vector(
```

```
    cor(delanteros[,c(3,4,5,6,7,8,12,14,15,17,18,19)])[1,-1])



) %>%
  kable(caption = "\\label{tablacor}Correlaciones",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 8,
                repeat_header_text = "(continúa)")
```

A.3.2.2. Media de goles

Código sobre el análisis de la media de goles con la variable objetivo.

```
g_goles_mercado1 <- jugadores_con_valor %>%
  ggplot(aes(goals_average,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth() +
  ylab("Valor de mercado") +
  xlab("Media de goles")

g_goles_mercado2 <- jugadores_con_valor %>%
  ggplot(aes(goals_average,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth() +
  scale_y_log10() +
  ylab("log(Valor de mercado)") +
  xlab("Media de goles")

ggarrange(g_goles_mercado1, g_goles_mercado2 ,
          labels = c("", "" ),
          ncol = 2, nrow = 1)
```

```
g_goles_mercado_posicion2 <- jugadores_con_valor %>%
  filter(position != "Goalkeeper")%>%
  ggplot(aes(goals_average,market_value_in_gbp,color = position)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth(aes(linetype = position),se = FALSE) +
  xlab("Media de goles") +
  ylab("Valor de mercado")

g_goles_mercado_posicion2
```

```
jugadores_con_valor %>%
  filter(position != "Goalkeeper") %>%
  ggplot() +
```

```
  geom_point(aes(goals_average,market_value_in_gbp), size = 0.5,
             alpha = 0.3) +
  facet_wrap(~ position, nrow = 2) +
  xlab("Media de goles") +
  ylab("Valor de mercado")
```

```
jugadores_con_valor %>%
  filter(goals_average > 30) %>%
  select(pretty_name.x,market_value_in_gbp,goals_average) %>%
  kable(caption = "Mayores goleadores", booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 12,
                repeat_header_text = "(continúa)")
```

A.3.2.3. Media de Minutos Jugados

Código sobre el análisis de la media de minutos jugados con la variable objetivo.

```
g_goles_mercado1 <- jugadores_con_valor %>%
  ggplot(aes(min_played_average,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth() +
  xlab("Media de minutos") +
  ylab("Valor de mercado")

g_goles_mercado2 <- jugadores_con_valor %>%
  ggplot(aes(min_played_average,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth() +
  scale_y_log10() +
  xlab("Media de minutos") +
  ylab("log(Valor de mercado)")

ggarrange(g_goles_mercado1, g_goles_mercado2 ,
          labels = c("", "" ),
          ncol = 2, nrow = 1)
```

A.3.2.4. Edad

Código sobre el análisis de la edad con la variable objetivo.

```
# Objeto gráfico base para los 4 gráficos siguientes
gb <- jugadores_con_valor %>%
  ggplot(aes(age)) +
  xlab('') + ylab('age (años)')

# Diagrama de caja y bigotes:
```

```
g0 <- gb +
  geom_boxplot(aes(y=1),width=.5) +
  geom_jitter(aes(y=1),width=0.1, height = .25) +
  scale_y_continuous(breaks=NULL)+
  ylab('')
# Diagrama de violín:
g0b <- gb +
  geom_violin(aes(y=1),width=.5) +
  scale_y_continuous(breaks=NULL)+
  xlab('Edad') +
  ylab('')

## No utilizado.
g1 <- gb +
  geom_dotplot(binwidth=.05) +
  ylim(0, 20) +
  coord_fixed(ratio=.05)

# Histograma:
g2 <- gb +
  geom_histogram(bins=30)

# Gráfico de densidad:
g3 <- gb +
  geom_density()

g4 <- faithful %>%
  ggplot(aes(sample=eruptions)) +
  geom_qq() +
  xlab('Cuantiles de una N(0, 1)')
```

```
ggarrange(g0b,
          labels = c("Diagrama de violín" ),
          ncol = 1, nrow = 1)
```

```
g_age_mercado1 <- jugadores_con_valor %>%
  ggplot(aes(age,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth()+
  xlab("Media de goles") +
  ylab("Valor de mercado")


g_age_mercado2 <- jugadores_con_valor %>%
  ggplot(aes(age,market_value_in_gbp)) +
  scale_y_log10() +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth() +
```

```
  xlab("Media de goles") +
  ylab("log(Valor de mercado)")


ggarrange(g_age_mercado1,g_age_mercado2,
          labels = c("" ),
          ncol = 2, nrow = 2)
```

```
jugadores_con_valor %>%
  ggplot(aes(age,market_value_in_gbp,color = position)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth(aes(linetype = position), se = FALSE) +
  scale_y_log10() +
  xlab("Edad") +
  ylab("log(Valor de mercado)")
```

A.3.2.5. Goles encajados por minuto

Código sobre el análisis de goles encajados por minuto con la variable objetivo.

```
g_goles_mercado1 <- jugadores_con_valor %>%
  ggplot(aes(goals_conceded_minute,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth()+
  xlab("Goles concedidos/minuto") +
  ylab("Valor de mercado")

g_goles_mercado2 <- jugadores_con_valor %>%
  ggplot(aes(goals_conceded_minute,market_value_in_gbp)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth() +
  scale_y_log10()+
  xlab("Goles concedidos/minuto") +
  ylab("log(Valor de mercado)")

ggarrange(g_goles_mercado1, g_goles_mercado2 ,
          labels = c("", "" ),
          ncol = 2, nrow = 1)
```

```
g_goles_mercado_posicion2 <- jugadores_con_valor %>%
  ggplot(aes(goals_conceded_minute,market_value_in_gbp,
             color = position)) +
  geom_point(size = 0.5, alpha = 0.3) +
  geom_smooth(aes(linetype = position), se = FALSE) +
  scale_y_log10() +
  xlab("Goles concedidos/minuto") +
  ylab("Valor de mercado")

g_goles_mercado_posicion2
```

```
jugadores_con_valor %>%
  ggplot() +
  geom_point(aes(goals_conceded_minute,market_value_in_gbp),
             size = 0.5, alpha = 0.3) +
  facet_wrap(~ position, nrow = 2) +
  xlab("Goles concedidos/minuto") +
  ylab("log(Valor de mercado)")
```

A.3.3. Estudio de las variables cualitativas

A continuación, se describe como se programó el estudio de las variables cualitativas en el estudio.

A.3.3.1. Posición

Código sobre el análisis de la variable objetivo en función de la posición de los futbolistas.

```
jugadores_posicion <- jugadores_con_valor %>%
  group_by(position) %>%
  summarise(Cantidad = n(),
            valor_medio = mean(market_value_in_gbp, na.rm = TRUE)) %>%
  kable(caption = "\\label{tabla314}Valor por posición",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 15,
                repeat_header_text = "(continúa)")
jugadores_posicion
```

```
a <- jugadores_con_valor %>%
  ggplot() +
  stat_summary(aes(position,market_value_in_gbp),
               fun = mean,
               fun.min = min,
               fun.max = max) +
  coord_flip()  +
  xlab("Valor de mercado") +
  ylab("Posición")
```

```
jugadores_con_valor %>%
  ggplot(mapping = aes(x = position, y = market_value_in_gbp)) +
  geom_boxplot() +
  coord_flip()
```

```
jugadores_con_valor %>%
  ggplot(mapping = aes(x = position, y = market_value_in_gbp)) +
  geom_violin() +
```

```
  coord_flip()
```

```
jugadores_con_valor %>%
  ggplot() +
  stat_summary(aes(position,market_value_in_gbp),
               fun = mean,
               fun.min = min,
               fun.max = max) +
  coord_flip() +
  scale_y_log10()
```

```
b <- jugadores_con_valor %>%
  ggplot(mapping = aes(x = position, y = market_value_in_gbp)) +
  geom_boxplot() +
  coord_flip() +
  scale_y_log10() +
  xlab("log(Valor de mercado)") +
  ylab("Posición")
```

```
c <- jugadores_con_valor %>%
  ggplot(mapping = aes(x = position, y = market_value_in_gbp)) +
  geom_violin() +
  coord_flip() +
  scale_y_log10() +
  xlab("log(Valor de mercado)") +
  ylab("Posición")
```

```
ggarrange(a, b ,c,
          labels = c("", "" ),
          ncol = 1, nrow = 3)
```

A.3.3.2. Pie

Código sobre el análisis de la variable objetivo en función del pie bueno de los futbolistas.

```
jugadores_pie <- jugadores_con_valor %>%
  group_by(foot) %>%
  summarise(Cantidad = n(),
            valor_medio = mean(market_value_in_gbp, na.rm = TRUE)) %>%
  filter(foot != "No Data")
jugadores_pie  %>%
  kable(caption = "\\label{tabla315}Valor por 'pie bueno' ",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 15,
                repeat_header_text = "(continúa)")
```

```
jugadores_con_valor %>%
  filter(foot != "No Data") %>%
  ggplot() +
  stat_summary(aes(foot,market_value_in_gbp),
               fun = mean,
               fun.min = min,
               fun.max = max) +
  xlab("Pie") +
  ylab("Valor de mercado")
```

A.3.3.3. Continente

Código sobre el análisis de la variable objetivo en función del continente de nacimiento de los futbolistas.

```
jugadores_continent <- jugadores_con_valor %>%
  group_by(continent) %>%
  summarise(Numero = n(),
            valor_medio = mean(market_value_in_gbp, na.rm = TRUE))
jugadores_continent %>%
  kable(caption = "\\label{tabla316}Valor por continente ",
        booktabs = T, longtable = T) %>%
  kable_styling(latex_options = c("striped","scale_down","repeat_header"),
                position = "center",
                full_width = FALSE,
                font_size = 15,
                repeat_header_text = "(continúa)")
```

```
jugadores_con_valor %>%
  ggplot(mapping = aes(x = continent, y = market_value_in_gbp)) +
  geom_violin() +
  coord_flip() +
  scale_y_log10() +
  xlab("Valor de mercado") +
  ylab("Continente")
```

A.3.3.4. Ligas

Código sobre el análisis de la variable objetivo en función de la liga donde juegan de los futbolistas.

```
jugadores_liga <- jugadores_con_valor %>%
  group_by(name) %>%
  summarise(num_jugadores = n(),
            valor_medio = mean(market_value_in_gbp, na.rm = TRUE))
```

```
jugadores_liga %>%
  ggplot() +
  geom_point(aes(name,valor_medio),size = 0.5, alpha = 0.3) +
```

```
  coord_flip()
```

```
jugadores_con_valor %>%
  ggplot(mapping = aes(x = name, y = market_value_in_gbp)) +
  geom_boxplot() +
  coord_flip() +
  scale_y_log10() +
  xlab("Valor de mercado") +
  ylab("Liga")
```

A.4. Modelización

Se muestra el código relacionado con la modelización para la predicción de la variable objetivo *valor de mercado*.

```
library(tidymodels)
```

A.4.1. Initial split

División de la muestra, en muestra de entrenamiento y test.

```
set.seed(46072925)
```

```
jugadores_split_estratos <- jugadores_con_valor %>%
  initial_split(strata = market_value_in_gbp,
                breaks = 4)
jugadores_split_estratos
```

```
jugadores_train <- training(jugadores_split_estratos)
jugadores_test <- testing(jugadores_split_estratos)
```

```
cv_folds <-
    vfold_cv(jugadores_train,
             v = 10,
             strata = market_value_in_gbp,
             breaks = 4)
cv_folds
```

A.4.2. Función de estratificación del rmse

Se define la función raíz del error cuadrático medio para hacer el cálculo por estratos.

```
rmse_filtrados <- function (minimo, maximo, modelo) {

  pred_filtradas <- modelo$.predictions[[1]] %>%
    filter(market_value_in_gbp > minimo) %>%
    filter(market_value_in_gbp < maximo)

  return(sqrt(mean((pred_filtradas$market_value_in_gbp
```

```
                    - pred_filtradas$.pred) ^ 2)))
}
```

A.4.3. Receta

Receta para el modelado.

```
receta <- recipe(market_value_in_gbp ~. , data = jugadores_con_valor) %>%
  step_dummy(all_nominal()) %>%
  step_corr(all_numeric(),-all_outcomes()) %>%
  step_center(all_numeric(),-all_outcomes()) %>%
  step_scale(all_numeric(),-all_outcomes())
receta
```

A.4.4. Árbol de decisión

Código dónde se define el modelo árbol de decisión (decision tree).

```
modelo_tree <- decision_tree(tree_depth = tune(),
                             min_n = tune()) %>%
  set_engine("rpart") %>%
  set_mode("regression")
```

```
tree_wf <-
  workflow() %>%
  add_recipe(receta) %>%
  add_model(modelo_tree)
tree_wf
```

```
tree_results <- tree_wf %>%
  tune_grid(resamples = cv_folds)
```

```
tree_rmse_best <- tree_results %>%
  show_best(metric = "rmse")
tree_rmse_best
tree_rmse <- tree_rmse_best$mean[1]/1e+6
```

```
tree_rsq_best <- tree_results %>%
  show_best(metric = "rsq")
tree_rsq_best
tree_rsq <- tree_rsq_best$mean[1]
```

A.4.5. Bosques aleatorios

Código dónde se define el modelo bosques aleatorios (random forest).

```
rf_tuner <-
  rand_forest(mtry = tune(),
              min_n = tune()) %>%
```

```
    set_engine("ranger") %>%
    set_mode("regression")
```

```
bosque_wf <-
  workflow() %>%
  add_recipe(receta) %>%
  add_model(rf_tuner)
bosque_wf
```

```
rf_results <-
  bosque_wf %>%
    tune_grid(resamples = cv_folds)
```

```
rf_rmse_best <- rf_results %>%
  show_best(metric = "rmse")
rf_rmse_best
rf_rmse <- rf_rmse_best$mean[1]
```

```
rf_rsq_best <- rf_results %>%
  show_best(metric = "rsq")
rf_rsq_best
rf_rsq <- rf_rsq_best$mean[1]
```

A.4.6. Vecino más cercano

Código dónde se define el modelo vecino más cercano (knn).

```
k1_30 <- expand_grid(neighbors = 1:30)
k1_30
```

```
knn_tuner <-
  nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn") %>%
    set_mode("regression")
```

```
knn_twf <-
  workflow() %>%
    add_recipe(receta) %>%
    add_model(knn_tuner)
```

```
knn_results <-
  knn_twf %>%
    tune_grid(resamples = cv_folds,
              grid = k1_30)
```

```
knn_rmse_best <- knn_results %>%
  show_best(metric = "rmse")
knn_rmse_best
knn_rmse <- knn_rmse_best$mean[1]
```

```
knn_rsq_best <- knn_results %>%
  show_best(metric = "rsq")
knn_rsq_best
knn_rsq <- knn_rsq_best$mean[1]
```

```
knn_results %>% autoplot()
```

A.4.7. Tabla resumen

Tabla resumen sobre el entrenamiento de los modelos.

```
tabla_resumen <- data.frame(
  `Método` = c("Árbol de decisión(rmse)","Árbol de decisión(rsq)",
               "Random Forest(rmse)", "Random Forest(rsq) ",
                "knn(rmse)","knn(rsq)"),

  `Métrica` = c(tree_rmse,tree_rsq,rf_rmse,rf_rsq,knn_rmse,knn_rsq))


library(kableExtra)

tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

A.4.8. Mejor modelo Árbol de decisión (rmse)

Código dónde se define el mejor modelo árbol de decisión según la raíz del error cuadrático medio y sus resultados en la muestra test.

```
tree_best_rmse <-
  tree_results %>%
    select_best(metric = "rmse")
tree_best_rmse
```

```
tree_wfl_final_rmse <-
  tree_wf %>%
    finalize_workflow(tree_best_rmse)
```

```
tree_test_results_rmse <-
  tree_wfl_final_rmse %>%
    last_fit(split = jugadores_split_estratos)
```

```
tree_final_rmse <- tree_test_results_rmse %>%
  collect_metrics()
tree_final_rmse
```

```
tabla_resumen <- data.frame(
  `Método` = c("<100k","<500k","500k - 1.5M", "1.5M - 5M",
               "<1.5","<5M",">5M","5M - 20M",
```

```
                ">20M","5M - 30M", ">30M",">40M","20 - 60M"
                ,">60M",">70M"),

  rmse = c(rmse_filtrados(0,100000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(0,500000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(500000,1500000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(1500000,5000000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(0,1500000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(0,5000000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,144000001,tree_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,20000000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(20000000,144000001,tree_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,30000000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(30000000,144000001,tree_test_results_rmse)/1e+06,
           rmse_filtrados(40000000,144000001,tree_test_results_rmse)/1e+06,
           rmse_filtrados(20000000,60000000,tree_test_results_rmse)/1e+06,
           rmse_filtrados(60000000,144000001,tree_test_results_rmse)/1e+06,
           rmse_filtrados(70000000,144000001,tree_test_results_rmse)/1e+06)
)


tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

A.4.9. Mejor modelo Árbol de decisión (rsq)

Código dónde se define el mejor modelo árbol de decisión según el R^2 y sus resultados en la muestra test.

```
tree_best_rsq <-
  tree_results %>%
    select_best(metric = "rsq")
tree_best_rsq
```

```
tree_wfl_final_rsq <-
  tree_wf %>%
    finalize_workflow(tree_best_rsq)
```

```
tree_test_results_rsq <-
  tree_wfl_final_rsq %>%
    last_fit(split = jugadores_split_estratos)
```

```
tree_final_rsq <- tree_test_results_rsq %>%
  collect_metrics()
tree_final_rsq
```

```
tabla_resumen <- data.frame(
  `Método` = c("<100k","<500k","500k - 1.5M", "1.5M - 5M",
```

```
                "<1.5","<5M",">5M","5M - 20M",
                ">20M","5M - 30M", ">30M",">40M",
                "20 - 60M" ,">60M",">70M"),

  rmse = c(rmse_filtrados(0,100000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(0,500000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(500000,1500000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(1500000,5000000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(0,1500000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(0,5000000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(5000000,144000001,tree_test_results_rsq)/1e+06,
           rmse_filtrados(5000000,20000000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(20000000,144000001,tree_test_results_rsq)/1e+06,
           rmse_filtrados(5000000,30000000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(30000000,144000001,tree_test_results_rsq)/1e+06,
           rmse_filtrados(40000000,144000001,tree_test_results_rsq)/1e+06,
           rmse_filtrados(20000000,60000000,tree_test_results_rsq)/1e+06,
           rmse_filtrados(60000000,144000001,tree_test_results_rsq)/1e+06,
           rmse_filtrados(70000000,144000001,tree_test_results_rsq)/1e+06)
)

tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

A.4.10. Mejor modelo Random Forest (rmse y rsq)

Código dónde se define el mejor modelo bosques aleatorios según el R^2 y raíz del error cuadrático medio y sus resultados en la muestra test.

```
rf_best_rmse <-
  rf_results %>%
    select_best(metric = "rmse")
rf_best_rmse
```

```
rf_wfl_final_rmse <-
  bosque_wf %>%
    finalize_workflow(rf_best_rmse)
```

```
 rf_test_results_rmse <-
  rf_wfl_final_rmse %>%
    last_fit(split = jugadores_split_estratos)
```

```
rf_final_rmse <- rf_test_results_rmse %>%
  collect_metrics()
rf_final_rmse
```

```
tabla_resumen <- data.frame(
  `Método` = c("<100k","<500k","500k - 1.5M", "1.5M - 5M", "<1.5",
```

```
                "<5M",">5M","5M - 20M",
                ">20M","5M - 30M", ">30M",
                ">40M","20 - 60M" ,">60M",">70M"),

  rmse = c(rmse_filtrados(0,100000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(0,500000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(500000,1500000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(1500000,5000000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(0,1500000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(0,5000000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,144000001,rf_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,20000000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(20000000,144000001,rf_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,30000000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(30000000,144000001,rf_test_results_rmse)/1e+06,
           rmse_filtrados(40000000,144000001,rf_test_results_rmse)/1e+06,
           rmse_filtrados(20000000,60000000,rf_test_results_rmse)/1e+06,
           rmse_filtrados(60000000,144000001,rf_test_results_rmse)/1e+06,
           rmse_filtrados(70000000,144000001,rf_test_results_rmse)/1e+06)
)

tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

A.4.11. Mejor modelo kNN (rmse)

Código dónde se define el mejor modelo árbol de decisión según la raíz del error cuadrático medio y sus resultados en la muestra test.

```
knn_best_rmse <-
  knn_results %>%
    select_best(metric = "rmse")
knn_best_rmse
```

```
knn_wfl_final_rmse <-
  knn_twf %>%
    finalize_workflow(knn_best_rmse)
```

```
knn_test_results_rmse <-
  knn_wfl_final_rmse %>%
    last_fit(split = jugadores_split_estratos)
```

```
knn_final_rmse <- knn_test_results_rmse %>%
  collect_metrics()
knn_final_rmse
```

```
tabla_resumen <- data.frame(
  `Método` = c("<100k","<500k","500k - 1.5M", "1.5M - 5M",
```

```
                "<1.5","<5M",">5M","5M - 20M",
                ">20M","5M - 30M", ">30M",">40M",
                "20 - 60M" ,">60M",">70M"),

  rmse = c(rmse_filtrados(0,100000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(0,500000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(500000,1500000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(1500000,5000000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(0,1500000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(0,5000000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,144000001,knn_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,20000000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(20000000,144000001,knn_test_results_rmse)/1e+06,
           rmse_filtrados(5000000,30000000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(30000000,144000001,knn_test_results_rmse)/1e+06,
           rmse_filtrados(40000000,144000001,knn_test_results_rmse)/1e+06,
           rmse_filtrados(20000000,60000000,knn_test_results_rmse)/1e+06,
           rmse_filtrados(60000000,144000001,knn_test_results_rmse)/1e+06,
           rmse_filtrados(70000000,144000001,knn_test_results_rmse)/1e+06)
)

tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

A.4.12. Mejor modelo kNN (rsq)

Código dónde se define el mejor modelo kNN según el R^2 y sus resultados en la muestra test.

```
knn_best_rsq <-
  knn_results %>%
    select_best(metric = "rsq")
knn_best_rsq
```

```
knn_wfl_final_rsq <-
  knn_twf %>%
    finalize_workflow(knn_best_rsq)
```

```
knn_test_results_rsq <-
  knn_wfl_final_rsq %>%
    last_fit(split = jugadores_split_estratos)
```

```
knn_final_rsq <- knn_test_results_rsq %>%
  collect_metrics()
knn_final_rsq
```

```
tabla_resumen <- data.frame(
  `Método` = c("<100k","<500k","500k - 1.5M", "1.5M - 5M",
```

```
                "<1.5","<5M",">5M","5M - 20M",
                ">20M","5M - 30M", ">30M",">40M",
                "20 - 60M" ,">60M",">70M"),

  rmse = c(rmse_filtrados(0,100000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(0,500000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(500000,1500000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(1500000,5000000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(0,1500000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(0,5000000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(5000000,144000001,knn_test_results_rsq)/1e+06,
           rmse_filtrados(5000000,20000000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(20000000,144000001,knn_test_results_rsq)/1e+06,
           rmse_filtrados(5000000,30000000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(30000000,144000001,knn_test_results_rsq)/1e+06,
           rmse_filtrados(40000000,144000001,knn_test_results_rsq)/1e+06,
           rmse_filtrados(20000000,60000000,knn_test_results_rsq)/1e+06,
           rmse_filtrados(60000000,144000001,knn_test_results_rsq)/1e+06,
           rmse_filtrados(70000000,144000001,knn_test_results_rsq)/1e+06)
)

tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

A.4.13. Regresión lineal

Código dónde se define el mejor modelo de regresión lineal múltiple según el R^2 y la raíz del error cuadrático medio y sus resultados en la muestra test.

```
modelo_lm <-
   linear_reg() %>%
   set_engine(engine = "lm")
```

```
lm_wf <-
  workflow() %>%
  add_recipe(receta) %>%
  add_model(modelo_lm)
lm_wf
```

```
modelo9 <- lm_wf %>%
  fit_resamples(resamples = cv_folds)
```

```
errores9 <- modelo9 %>% collect_metrics()
errores9
```

```
rsq9 <- errores9$mean[2]
rmse9 <- errores9$mean[1]/1e+06
```

A.4.14. Redes neuronales

Código dónde se define el mejor modelo de redes neuronales según el R^2 y la raíz del error cuadrático medio y sus resultados en la muestra test.

```
library(keras)
```

```
receta_train <- recipe(market_value_in_gbp ~. ,
                       data = jugadores_con_valor) %>%
  step_dummy(all_nominal(), one_hot = TRUE) %>%
  step_corr(all_numeric(),-all_outcomes()) %>%
  step_center(all_numeric(),-all_outcomes()) %>%
  step_scale(all_numeric(),-all_outcomes()) %>%
  prep()


jugadores_train <- bake(receta_train, new_data = NULL)
train_labels <- as.vector(as.matrix(jugadores_train[,11]))
jugadores_train <- as.matrix(jugadores_train[,-11])

jugadores_test <- receta_train %>%
                        bake(testing(jugadores_split_estratos))
test_labels <- jugadores_test[,11]
test_labels <- as.vector(as.matrix(test_labels))
jugadores_test <- jugadores_test[,-11]
jugadores_test <- as.matrix(jugadores_test)
```

```
build_model <- function() {

  model <- keras_model_sequential() %>%
    layer_dense(units = 64, activation = "relu",
                input_shape = dim(jugadores_train)[2]) %>%
#    layer_dropout(0.6) %>%
    layer_dense(units = 32, activation = "relu") %>%
#    layer_dropout(0.4) %>%
    layer_dense(units = 16, activation = "relu") %>%
#    layer_dropout(0.2) %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_squared_error")
  )

  model
}

model <- build_model()
```

```
model %>% summary()
```

```
print_dot_callback <- callback_lambda(
  on_epoch_end = function(epoch, logs) {
    if (epoch %% 80 == 0) cat("\n")
    cat(".")
  }
)
```

```
epochs <- 500

history <- model %>% fit(
  jugadores_train,
  train_labels,
  epochs = epochs,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(print_dot_callback)
)
```

```
plot(history, metrics = "mean_squared_error", smooth = FALSE,
     ylab = "Error cuadrático medio") +
  coord_cartesian(ylim = c(0, 1.5e+14))
```

```
plot(history$metrics$mean_squared_error, smooth = FALSE,
     ylab = "Error cuadrático medio",
     xlab = "Índices")
```

```
c(loss, rmse) %<-% (model %>% evaluate(jugadores_test,
                                       test_labels, verbose = 0))
```

```
test_predictions <- model %>% predict(jugadores_test)
```

```
rmse_rrnn = (sqrt(rmse))/1e+06
r2_rrnn = ( sum((test_predictions[ , 1]-mean(test_labels))^2 )
            / sum((test_labels-mean(test_labels))^2 ))
```

A.4.15. Tabla resumen final

Tabla resumen sobre el resultado final de los modelos sobre la muestra test.

```
tabla_resumen <- data.frame(
  `Método` = c("Árbol de decisión(rmse)","Árbol de decisión(rsq)",
               "Random Forest(rmse y rsq) ",
                "knn(rmse)","knn(rsq)",
               "Regresión lineal","Red mono-neuronal"),

  rsq = c(tree_final_rmse$.estimate[2],
    tree_final_rsq$.estimate[2],
    rf_final_rmse$.estimate[2],
```

```
    knn_final_rmse$.estimate[2],
    knn_final_rsq$.estimate[2],
    rsq9,rsq_neurona),

  rmse = c(tree_final_rmse$.estimate[1]/1e+06,
           tree_final_rsq$.estimate[1]/1e+06,
           rf_final_rmse$.estimate[1]/1e+06,
           knn_final_rmse$.estimate[1]/1e+06,
           knn_final_rsq$.estimate[1]/1e+06,
           rmse9,rmse_neurona)
)

tabla_resumen %>%
  kable(booktabs=TRUE, longtable=TRUE) %>%
  kable_styling(latex_options = c("striped","repeat_header") )
```

Bibliografía

F. J. Martín Mateos D. Balbontín Noval, et al. *Introducción a las redes neuronales.*

Alison Hill. *Introduction to Machine Learning with the Tidyverse*, 2020.

Rob Kabacoff. *Data Visualization with R.* Opensource, first edition, 2015.

Kaggle. Búsqueda de datos. Disponible en https://www.kaggle.com.

Jerome H. Friedman Leo Breiman, et al. *Classification And Regression Trees.* Routledge, first edition, 1984.

Pedro L. Luque-Calvo. *Escribir un Trabajo Fin de Estudios con R Markdown*, 2017.

Juan Etxeberria Murgiondo. *Regresión múltiple.* La Muralla S.A, first edition, 2007.

Transfermarkt. Valores de mercado. Disponible en https://www.transfermarkt.es.

Claus O. Wilke. *Fundamentals of Data Visualization.* Editorial O'Reilly Media, Inc, first edition, 2019.

Karlijn Willems. keras: Deep learning in r. 1999.