

FACULTAD DE CIENCIAS
GRADO EN MATEMÁTICAS
TRABAJO FIN DE GRADO
CURSO ACADÉMICO [2024-2025]

TÍTULO:

**LAS MATEMÁTICAS EN EL DEPORTE: ANÁLISIS DE DATOS EN
EL FÚTBOL**

AUTOR:

BENLAGRAA GHELGAOUI, MOHAMED EL AMIN

Resumen

El análisis de datos se ha consolidado como una herramienta fundamental en el deporte y, en los últimos años, ha cobrado especial relevancia en el “deporte rey”, el fútbol. Inspirado por casos emblemáticos como “Moneyball” en el béisbol, que fue un caso pionero en demostrar el poder del análisis de datos en la toma de decisiones deportivas. En el fútbol, dada su naturaleza continua, se incorporó más tarde a esta revolución, abriendo un nuevo horizonte para optimizar el rendimiento deportivo y las decisiones económicas.

En este trabajo, se van a introducir y aplicar técnicas matemáticas de análisis de datos, inferencia y optimización al fútbol, con el objetivo de transformar datos deportivos en decisiones estratégicas. Para ello, se introducen conceptos básicos de análisis de datos como las variables, los conjuntos de datos y los modelos de predicción.

*A continuación, se realiza un **Análisis Exploratorio de Datos (EDA)** sobre un dataset de 1130 jugadores de LaLiga y la Premier League de la temporada 2023/24. Este análisis, junto con el estudio de correlaciones permitirá extraer las primeras conclusiones acerca del conjunto de datos y de cómo enfocar nuestro estudio.*

En el siguiente pasa, se presentan diversos métodos de Machine Learning, con un énfasis especial en redes neuronales de tipo Transformer (la arquitectura detrás de ChatGPT) y cómo a partir de ellos llegamos a construir modelos adaptados a datos tabulares, como TabPFN. Estos modelos se emplearán para definir y evaluar métricas personalizadas de rendimiento (KPIs), las cuales han sido optimizadas mediante la importancia de variables, para predecir el impacto de los jugadores en el terreno de juego y en su valor de mercado, identificando fichajes infravalorados con potencial económico.

Por último, se comparará el rendimiento de los distintos modelos según las diferentes métricas, y se discutirán sus limitaciones, como por ejemplo, la escalabilidad de TabPFN en conjuntos de datos muy grandes o los retos de predecir resultados en un deporte continuo como el fútbol. Como conclusión, se discutirán los resultados obtenidos y se propondrán trabajos futuros.

Palabras clave: Análisis de datos, Redes neuronales, Transformers, TabPFN, Fútbol.

Abstract

Data analysis has become a fundamental tool in sports and, in recent years, has gained particular relevance in the so-called “king of sports”, football. Inspired by emblematic cases like “Moneyball” in baseball, a pioneer example that showed the power of data analysis in sports decision-making, in football, due to its continuous nature, joined this revolution later, opening for a new horizon for improving both performance and economic decisions. In this project, mathematical techniques from data analysis, inference, and optimization are applied to football, aiming to turn sports data into strategic decisions.

To do so, some basic data analysis concepts are introduced, such as variables, datasets, and prediction models.

*Following this, we are going to do an **Exploratory Data Analysis (EDA)** is carried out on a dataset of 1130 players from LaLiga and the Premier League from the 2023/24 season. This analysis, combined with correlation analysis helps draw the first conclusions from the data.*

Next, several Machine Learning methods are presented, with a special focus on neural networks based on the Transformer architecture (used in models like ChatGPT), and how these ideas lead to models adapted to tabular data, such as TabPFN. These models are used to define and evaluate custom performance metrics (KPIs), which are optimized using feature importance to predict players’ impact on the pitch and their market value, helping identify undervalued players with economic potential.

Finally, the performance of different models is compared using several metrics, and some limitations are discussed, such as the scalability of TabPFN on large datasets or the challenge of predicting outcomes in a continuous sport like football. As a conclusion, the results are discussed and possible future work is proposed.

Key words: Data analysis, Neural networks, Transformers, TabPFN, Football.

Índice

| | |
|--|-----------|
| 1. Introducción | 5 |
| 2. Conjuntos de datos | 8 |
| 2.1. ¿Qué son los datos? | 8 |
| 2.1.1. Datos empleados. | 10 |
| 2.2. Obtención y procesamiento de los datos. | 11 |
| 3. Exploración y segmentación de datos | 15 |
| 3.1. Análisis exploratorio de de datos (EDA) | 15 |
| 3.2. Análisis de correlaciones | 20 |
| 4. Modelos predictivos de ML | 22 |
| 4.1. Métricas de evaluación | 22 |
| 4.2. Métodos lineales generalizados (GLM) | 25 |
| 4.3. Weak learners | 27 |
| 4.4. Modelos ensemble | 27 |
| 5. Modelos preentrenados | 34 |
| 5.1. Redes neuronales. | 34 |
| 5.1.1. Modelos Transformer | 38 |
| 5.1.2. TabPFN | 43 |
| 6. Discusión | 47 |
| 7. Conclusiones y futuros trabajos | 53 |
| Referencias | 54 |
| A. Detalles del desarrollo del trabajo | 56 |
| B. Apéndice | 57 |

1. Introducción

Motivación

El **análisis de datos** consiste en un conjunto de técnicas y herramientas diseñadas para recopilar, procesar y examinar información con el fin de extraer patrones, tendencias y datos interesantes de nuestros datos. Haciendo uso de métodos estadísticos, algoritmos de aprendizaje automático y tecnologías como redes neuronales o modelos transformers, se transforma la información cruda en decisiones.

La elección del tema del fútbol viene de que, en los últimos años, muchos clubes de fútbol se han interesado en el análisis de datos como una herramienta potente para tomar las decisiones correctas.

Además de ser un sector interesante para aplicar el análisis de datos, ya que no es un simple deporte, sino que tiene una gran importancia en muchos niveles, por ejemplo :

- **Impacto económico:**

Durante la temporada 2021/22 en España, el fútbol profesional generó más de 18.350 M€, lo que representa 1,4 % del PIB español. Y unos 80.000 M€ comparado a la industria cinematográfica with unos 90.000 M€ anuales [1].

- **Influencia política:**

El fútbol es una herramienta muy utilizada en la política como para “lavar” la imagen de un país, promover el turismo, mandar mensajes en contra del racismo, etc.

- **Poder social:**

Los jugadores famosos llegan a conseguir mucha notoriedad a tal punto que puede llegar a acabar con guerras como Didier Drogba en 2005 con la guerra civil en Costa de Marfil [2].

Objetivos del trabajo

Este trabajo tiene un enfoque teórico-práctico con el objetivo de aplicar técnicas matemáticas de análisis de datos al fútbol. Para ello, el objetivo principal de este trabajo será descubrir, explicar y aplicar distintas técnicas de **análisis de datos**, desde las más básicas como EDA, a las más complejas como las Redes Neuronales. En las cuales se va a profundizar en la creación y funcionamiento de los modelos *Transformers* y los preentrenados más modernos, como **TabPFN**.

El objetivo secundario es aplicar estas técnicas o modelos a lo largo del TFG. Para ayudar al **Villarreal CF** a detectar los problemas del rendimiento del equipo y proponer

soluciones, como identificar fichajes estratégicos. Con el fin de hacer el trabajo más realista, vamos a suponer que somos el equipo de análisis de datos del club y tenemos un presupuesto de **50M€**. Para ello, será necesario obtener y procesar un conjunto de datos, relacionado con el fútbol, que nos permita hacer este estudio.

Y se buscará responder a las siguientes preguntas:

Q: ¿Cómo podemos implementar el análisis de datos en los deportes?

Q: ¿Cómo se contruyen los modelos transformer desde el punto de vista matemático?

Q: ¿Por qué autoatención?

Q: ¿Qué ventajas ofrece TabPFN frente a otros modelos de regresión tradicionales en datos tabulares?

Q: ¿Podemos usar el análisis de datos para tomar decisiones económicas, como fichar jugadores con proyección? ¿En qué variables o estadísticas se basan los expertos para definir el valor de mercado de los jugadores?

Estructura de trabajo

Con el fin de poder abordar el análisis de datos en el fútbol desde una perspectiva matemática, y siguiendo unos pasos coherentes, el trabajo se va a organizar en las siguientes secciones:

- **Conjunto de datos:** Se introducirá el concepto de variable, conjunto de datos, y presentará nuestro conjunto de datos, objeto de estudio. Describiendo su obtención, limpieza y normalización. Además de introducir el concepto de indicadores de rendimiento y su empleo para definir nueva variables de gran utilidad.
- **Exploración y Segmentación de Datos:** Se realizará un análisis exploratorio para identificar patrones, correlaciones entre variables y segmentación mediante clustering para así sacar conclusiones sobre el rendimiento jugadores.
- **Modelos Predictivos de Machine Learning:** En esta sección se aplican modelos supervisados, como Regresión Lineal, Random Forest y XGBoost, para predecir métricas de rendimiento ofensivo y defensivo, evaluando su eficacia con métricas

como MSE y MAE. Esto nos servirá para encontrar jugadores con un buen rendimiento, y los vamos a añadir una *shortlist*, de la cual vamos a seleccionar los jugadores que vamos a fichar al final.

- **Modelos Preentrenados:** En esta sección se introducirán las redes neuronales como los modelos Transformer y a continuación TabPFN, con el objetivo de descubrir modelos más complejos y robustos comparado con los modelos de ML vistos hasta este punto. Y esto lo vamos a comprobar con ejemplos prácticos para comparar las capacidades predictivas de los modelos anteriores con TabPFN.
- **Discusión:** Se comparan los resultados de los modelos, y se utiliza el mejor modelo para predecir el valor de mercado de los jugadores preseleccionados en la *shortlist*. Se responderá a las preguntas planteadas en el apartado anterior, y se discutirán las principales limitaciones del trabajo.
- **Conclusiones y futuros trabajos:** Y para concluir se hará un resumen de los hallazgos, destacando cómo el análisis de datos optimiza el rendimiento deportivo. Y se proponen algunas mejoras, como el uso de datos en tiempo real.

2. Conjuntos de datos

Esta sección tiene como objetivo introducir el concepto de conjunto de datos, de variable, y la creación de un conjunto de datos fiable relacionado con el fútbol. Y se presentará este conjunto de datos, el cual usaremos a lo largo del trabajo, describiendo su obtención y tratamiento. Además, se introducirá el concepto de indicadores de rendimiento y su empleo para definir nuevas variables de gran utilidad.

2.1. ¿Qué son los datos?

En análisis de datos llamamos datos, conjunto de datos o en inglés *dataset*, a una colección organizada de información.

Tipos de datos

El formato de los datos puede venir en varias formas, desde **datos no estructurados**, como texto libre, imágenes, audios. **datos semi-estructurados**, son datos que no están organizados en formato de filas y columnas, por ejemplo, archivos JSON, XML.

Datos tabulares

Los datos estructurados por excelencia son a los que llamamos datos tabulares, a los conjuntos de datos organizados en una tabla o matriz, donde cada fila representa una observación o individuo y las columnas representan variables sobre esas observaciones o individuos. Es el tipo de datos más común en las empresas, ya que es el más fácil de entender e interpretar. Como, por ejemplo, las tablas de Excel.

De este modo podemos expresar los datos como la matriz de datos X , donde X_1, \dots, X_p son las variables del conjunto, y así tenemos:

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \quad (1)$$

Donde x_{ij} , es el valor que toma el individuo i en la variable j .

Series temporales

Un tipo de datos muy populares son las series temporales. Los datos de series temporales presentan una ventaja respecto a los datos atemporales, y esta es que podemos estudiar como evoluciona una variable respecto del tiempo.

Tipos de variables

Las variables X_1, \dots, X_p , pueden ser de varios tipos, tenemos:

- **Variables cualitativas:** Son las variables que representan una característica o propiedad de la observación.
- **Variables cuantitativas:** Son las que representan una cantidad numérica.
- **Variables binarias:** Son variables que representan la posesión de dicha característica o pertenencia a dicho grupo. Un ejemplo, son las **dummies**.

Además de esto, también diferenciamos las **variables independientes o explicativas** x_{i1}, \dots, x_{ip} y las **dependientes o explicadas** y_i . Y suelen seguir la siguiente relación:

$$y_i = f(x_{i1}, \dots, x_{ip}) + \varepsilon_i, \quad i = 1, \dots, n, \quad (2)$$

De forma matricial, se expresa como $y = X\beta + \varepsilon$, donde X es la matriz de variables explicativas, β el vector de coeficientes y ε el vector de errores. Un ejemplo muy utilizado es **regresión lineal**, donde la relación entre estas variables se modela comúnmente mediante una función lineal con un término de error, expresada como:

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i, \quad i = 1, \dots, n, \quad (3)$$

donde $\beta_0, \beta_1, \dots, \beta_p$ son los parámetros del modelo, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ es el término de error aleatorio, y n es el número de observaciones. Esta formulación asume una relación lineal entre las variables explicativas y la explicada, con el objetivo de minimizar la suma de errores cuadrados $\sum_{i=1}^n \varepsilon_i^2$.

2.1.1. Datos empleados.

El conjunto de datos usado está formado por 1130 individuos, que son **Jugadores** de equipos de la liga española (La Liga) y la liga inglesa (Premier League). Y 55 variables, de las cuales tenemos cuantitativas, cualitativas y binarias, que representan estadísticas y características de jugadores de la temporada 2023/24. En la **Figura 1** tenemos una captura de parte del conjunto de datos empleado, en Excel:

| Jugador | País | Posc | Equipo | Edad | Nacimiento | PJ | PG | PE | PP |
|---------------------|------|------|-------------|------|------------|----|----|----|----|
| Erling Haaland | NOR | DL | Manchester | 23 | 2000 | 31 | 28 | 7 | 3 |
| Lamine Yamal | ESP | DL | Barcelona | 16 | 2007 | 37 | 26 | 7 | 5 |
| Jude Bellingham | ENG | CC | Real Madrid | 20 | 2003 | 28 | 29 | 8 | 1 |
| Bukayo Saka | ENG | DL | Arsenal | 21 | 2001 | 35 | 28 | 5 | 5 |
| Phil Foden | ENG | DL | Manchester | 23 | 2000 | 35 | 28 | 7 | 3 |
| Federico Valverde | URU | CC | Real Madrid | 25 | 1998 | 37 | 29 | 8 | 1 |
| Cole Palmer | ENG | CC | Manchester | 21 | 2002 | 1 | 28 | 7 | 3 |
| Cole Palmer | ENG | DL | Chelsea | 21 | 2002 | 33 | 18 | 9 | 11 |
| Rodri | ESP | CC | Manchester | 27 | 1996 | 34 | 28 | 7 | 3 |
| Martin Ødegaard | NOR | CC | Arsenal | 24 | 1998 | 35 | 28 | 5 | 5 |
| Declan Rice | ENG | CC | Arsenal | 24 | 1999 | 38 | 28 | 5 | 5 |
| Pedri | ESP | CC | Barcelona | 20 | 2002 | 24 | 26 | 7 | 5 |
| Rodrygo | BRA | DL | Real Madrid | 22 | 2001 | 34 | 29 | 8 | 1 |
| Luis Díaz | COL | DL | Liverpool | 26 | 1997 | 37 | 24 | 10 | 4 |
| Raphinha | BRA | DL | Barcelona | 26 | 1996 | 28 | 26 | 7 | 5 |
| William Saliba | FRA | DF | Arsenal | 22 | 2001 | 38 | 28 | 5 | 5 |
| Gavi | ESP | CC | Barcelona | 18 | 2004 | 12 | 26 | 7 | 5 |
| Alexis Mac Allister | ARG | CC | Liverpool | 24 | 1998 | 33 | 24 | 10 | 4 |
| Aurélien Tchouaméni | FRA | CC | Real Madrid | 23 | 2000 | 27 | 29 | 8 | 1 |
| Moisés Caicedo | ECU | CC | Chelsea | 21 | 2001 | 35 | 18 | 9 | 11 |
| Eduardo Camavinga | FRA | CC | Real Madrid | 20 | 2002 | 31 | 29 | 8 | 1 |

Figura 1: Aquí tenemos una pequeña muestra del conjunto de datos. Nuestro dataset entero se puede representar como una matriz de datos $X_{1130 \times 55}$.

En la **Tabla 1** vemos algunas de las variables más interesantes de nuestro conjunto de datos. El resto de las variables las podemos encontrar explicadas en el apéndice.

| Variables | Descripción | Categoría |
|-----------|--|--------------|
| País | Nacionalidad del jugador el | Cualitativa |
| Equipo | Equipo donde juega el | Cualitativa |
| Posc | Posición de jugador en el terreno de juego | Cualitativa |
| Liga | La liga en la que juega el jugador | Binaria |
| Edad | Edad de cada jugador | Cuantitativa |
| Gls | Goles del jugador durante la temporada | Cuantitativa |
| Ass | Asistencias del jugador durante la temporada | Cuantitativa |
| ... | ... | ... |

Tabla 1: Algunas variables del conjunto de datos usado

2.2. Obtención y procesamiento de los datos.

Antes de empezar a hacer el estudio, debemos obtener el conjunto de datos deseado para cumplir con los objetivos del trabajo. Para obtener un conjunto de datos, tenemos principalmente las siguientes opciones.

1. Bases de datos gratuitas:

Una de las opciones más populares para obtener un conjunto de datos es la base de datos *Kaggle*. Y aparte de esta, hay una multitud de bases de datos gratuitas.

2. *Web scrapping*

Para obtener el conjunto de datos se han filtrado varias opciones de bases de datos con datos gratuitos, pero los datos ofrecidos no me contenían suficiente información para realizar nuestro estudio. Otra opción gratuita y por la que he optado es el *web scrapping*, que consiste en: 1º Encontrar la página/base de datos que tenga datos interesantes, 2º Usar programación, principalmente Python para obtener los datos en un bloque de texto y 3º Procesar los datos.

El conjunto de datos ha sido recolectado de la página web o base de datos *FBref* [3].

Las variables han sido escogidas de manera subjetiva, ya que hay muchas variables que no han sido tomadas en cuenta. En la página, cada jugador tiene alrededor de 200.

Procesamiento y depuración de los datos

Uno de los pasos que pueden ser más pesados y que consumen más tiempo es el tratamiento de los datos. Ya que la mayoría de los modelos de ML suelen tener problemas cuando un *dataset* posee muchos valores atípicos, valores nulos, etc.

■ Eliminación de observaciones:

En este paso se eliminan las observaciones que tengan algún tipo de error, y las variables que puedan ser un poco redundantes o inútiles. En nuestro *dataset* hemos eliminado los jugadores con menos de 4 partidos ya que son insignificantes para un estudio serio. El problema de esto es que hay jugadores que han tenido una lesión que les haya dejado fuera durante la temporada, pero en realidad son buenos jugadores como *Tyrone Mings* y que pueden ser interesantes para comparar con otros jugadores.

■ Detectar y complera los *missing values*:

Como en la mayoría de conjuntos de datos reales, en nuestro conjunto de datos

hemos encontrado muchos valores nulos.

Existen varias estrategias para corregir esto. Una de ella es quitar la observación donde hay algún valor nulo, pero esta no es una muy buena estrategia ya que así podemos perder observaciones con información importante además de que puede reducir considerablemente de conjunto de datos. Otra técnica podría ser sustituir los valores nulos por 0 o por la media o mediana de esa variable. Un mejor alternativa podría ser MICE [4], un paquete de R robusto y ampliamente utilizado para la imputación de datos faltantes que se puede usar para imputar datos faltantes con varios métodos, como regresión lineal, regresión logística, y CARTs.

■ Normalización de variables:

Es una técnica que consiste transformar la escala de los datos a una escala más fácil de entender y para prevenir errores como por ejemplo que las variables con escales mayores tenderán a tener mayor varianza y dominan en la matriz de covarianza. Un tipo de estandarización es la normalización que lleva los datos al intervalo entre 0 y 1. Esta técnica es necesaria par modelos como regresión lineal donde las variables con valores en valor absoluto más grandes que el resto, van a pesar más sobre el modelo. Se normaliza una variable x de la siguiente forma:

$$x_{normalizado} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (4)$$

Los modelos que no se ven afectados por la estandarización de variables son los modelos CART y sus derivados (Random Forest, etc).

Técnicas para enriquezer nuestros datos

Estas técnicas consisten en usar nuestro *dataset* original y, mediante combinaciones estadísticas entre las variables, sacar variables extras que pueden ser de gran utilidad a la hora de definir el objetivo del análisis.

Bootstrap

La técnica de *Bootstrap* es un método de remuestreo estadístico utilizado para estimar la variabilidad de un estadístico cuando únicamente se dispone de una muestra. Dada una muestra de tamaño n , $\mathbf{X} = (X_1, X_2, \dots, X_n)$, obtenida de una distribución poblacional desconocida F , y un estadístico $\theta = s(\mathbf{X})$ (por ejemplo, la media). Esta técnica consiste en [6]:

1. Generar B muestras *bootstrap* (e.g., $B = 1000$), cada una de tamaño n , seleccionando

observaciones con reemplazo de \mathbf{X} . Cada muestra $\mathbf{X}_b^* = (X_{b1}^*, X_{b2}^*, \dots, X_{bn}^*)$, para $b = 1, \dots, B$, puede contener elementos repetidos de la muestra original.

2. Calcular el estadístico $\theta_b^* = s(\mathbf{X}_b^*)$ para cada una de las muestras *bootstrap*.
3. Aproximar la distribución de θ mediante la distribución empírica de $\{\theta_1^*, \theta_2^*, \dots, \theta_B^*\}$.
Por ejemplo, el error estándar se puede estimar como:

$$SE^* = \sqrt{\frac{1}{B-1} \sum_{b=1}^B (\theta_b^* - \bar{\theta}^*)^2}, \quad \text{donde} \quad \bar{\theta}^* = \frac{1}{B} \sum_{b=1}^B \theta_b^*.$$

4. Construir intervalos de confianza (e.g., método percentil): un intervalo de confianza del $100(1-\alpha)\%$ se obtiene tomando los percentiles $\alpha/2$ y $1-\alpha/2$ de $\{\theta_1^*, \theta_2^*, \dots, \theta_B^*\}$.

La validez del *Bootstrap* se fundamenta en el Teorema Central del Límite (TCL), que garantiza que para muestras grandes, la distribución de θ se aproxima a la normal alrededor del valor poblacional, siempre que $s(\mathbf{X})$ sea una función suave. Las muestras *bootstrap* replican la variabilidad de la población y, cuando $B \rightarrow \infty$, la distribución empírica de θ_b^* converge a la distribución muestral real de θ [?]. En este TFG, *Bootstrap* se emplea para estimar intervalos de confianza de KPIs (e.g., métricas de rendimiento defensivo) en el conjunto de datos de fútbol, permitiendo compensar la limitación de disponer únicamente de una muestra observada.

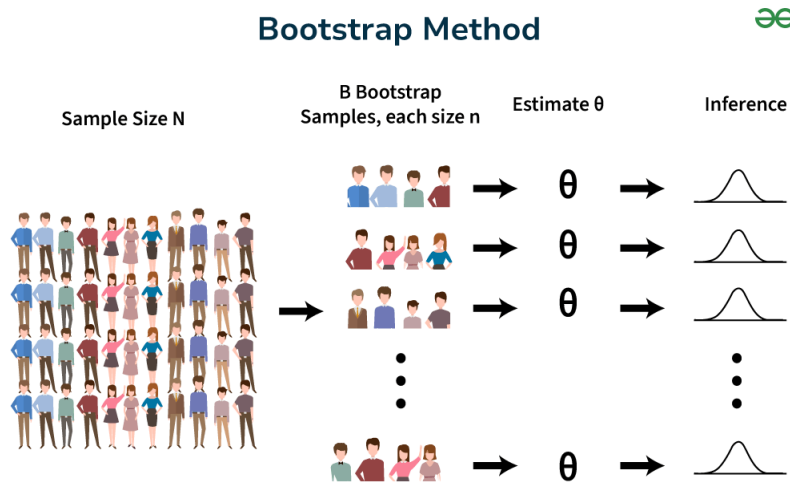


Figura 2: Imagen de Geeksforgeeks

KPIs

Llamamos KPI (*Key Performance Indicator*) o en castellano, indicador clave de rendimiento a la función que nos sirve para evaluar el rendimiento de un jugador.

Sea $X \subseteq \mathbb{R}^n$, entonces la función KPI vendrá dada por:

$$KPI : X \rightarrow \mathbb{R} \quad (5)$$

Siendo $KPI(x)$ el nivel de rendimiento del vector $x \in X$. Un par de ejemplos de KPIs, son las métricas de rendimiento ofensivo y defensivo que hemos creado para poder evaluar la calidad defensiva y ofensiva de una forma más objetiva: Más

```
df['Rend_Defensivo'] = 0.1*df['DisBlok'] + 0.15*df['Pases.Peligrosos'] + 0.1*df['Dentro-Fuera'] + 0.22*df['Intercep'] + 0.43*df['DesafiosExt']
```

Figura 3: Nueva variable para evaluar el rendimiento defensivo

```
df['Rend_Ofensivo'] = 0.43*df['Gls'] + 0.1*df['Ass'] + 0.29*df['Pases.Peligrosos'] + 0.08*df['G/T'] + 0.1*df['RegatesExt']
```

Figura 4: Nueva variable para evaluar el rendimiento ofensivo

adelante se verá de dónde han salido los pesos o coeficientes que acompañan a cada variable y el por qué de la elección de estas variables entre las 55 que tenemos en total.

Variables adicionales

Parecido a los KPIs pero solo como combinación de variables interesantes para reducir el número de variables para tener las más interesantes. En la **Figura 5** hemos transformado dos variables en una sola.

```
# Definir nueva variable que representa mejor los desafios con exito de los defensas
df['DesafiosExt'] = (df['DesafiosDef']) * (0.01*df['DesafExt%'])
```

Figura 5: Variable para definir la cantidad de desafios defensivos exitosos

Esta variable representa la diferencia **Dentro-Fuera** = $\frac{\text{onG} - \text{onGC}}{90}$, donde onG - onGC es la diferencia de goles que marca el equipo y los que recibe, cuando el jugador está en el campo, por cada 90 minutos (partido)

Para finalizar esta sección y tras todos estos cambios, nos hemos quedado con un conjunto de datos de **1005 individuos** y **48 variables** (variables más importantes, incluidos KPIs) y sin valores nulos. Y nuestro *dataset* será una matriz de datos $X_{1005 \times 48}$.

3. Exploración y segmentación de datos

Una vez que tenemos nuestro conjunto de datos limpio y organizado, podemos pasar a realizar una de las partes del análisis de datos que tiene como objetivo estudiar y aplicar diferentes métodos estadísticos más visuales y poder así obtener ciertas conclusiones acerca de la distribución de las distintas variables.

Para ello se va a hacer uso de herramientas muy conocidas, como R a través de RStudio y Power BI. Siguiendo los objetivos de este trabajo, en este apartado vamos a proceder a encontrar los jugadores, variables o características que pueden servir para emparejar y sacar conclusiones a través de gráficas u otra forma.

Aprendizaje no supervisado

El aprendizaje no supervisado es una técnica de *Machine Learning* que se distingue por trabajar sobre datos no “etiquetados”, es decir, se desconoce a qué grupo o clase pertenecen los individuos y, por tanto, la variable respuesta es desconocida. A lo largo de este trabajo se utilizarán diferentes técnicas propias del aprendizaje no supervisado para la ejecución del análisis exploratorio de los datos.

3.1. Análisis exploratorio de de datos (EDA)

La EDA es un conjunto de técnicas de resumen y visualización de un conjunto de datos con el propósito de descubrir patrones, detectar anomalías, comprobar supuestos y formular hipótesis, empleando métodos estadísticos y gráficas antes de cualquier modelización. Y sirve tanto con el objetivo de clasificación como de regresión.

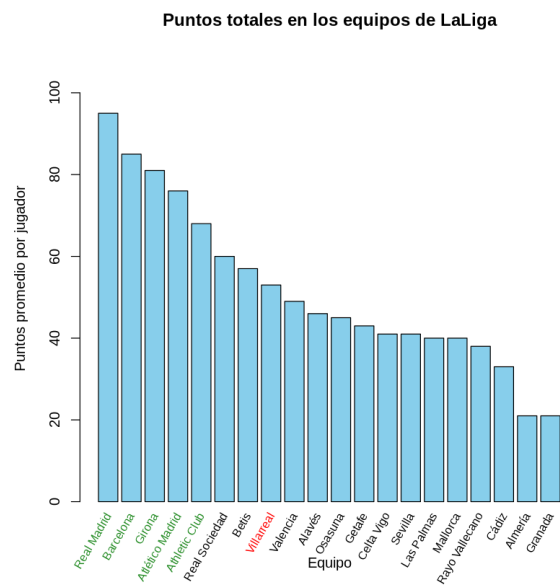


Figura 6: 8ª posición en LaLiga

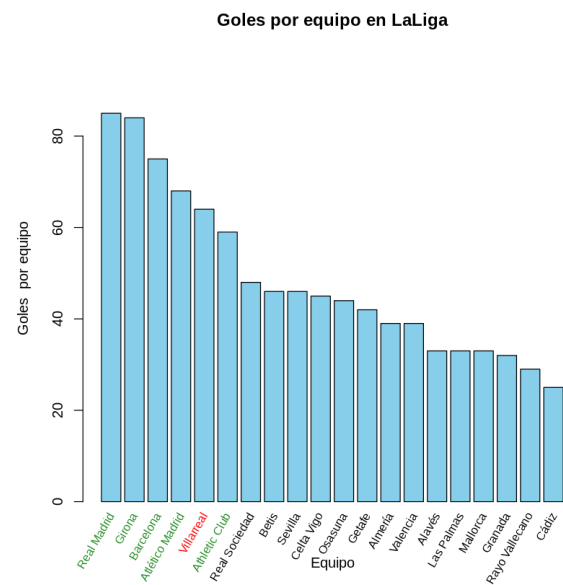


Figura 7: 5ª mejor delantera de LaLiga

En las Figuras 5 y 6 podemos observar que el Villarreal, a pesar de tener una buena delantera, se ha quedado en la 8ª posición, lo cual le impide participar en competiciones europeas. Veamos ahora en las siguientes figuras alguna explicación de a qué se puede deber esto.

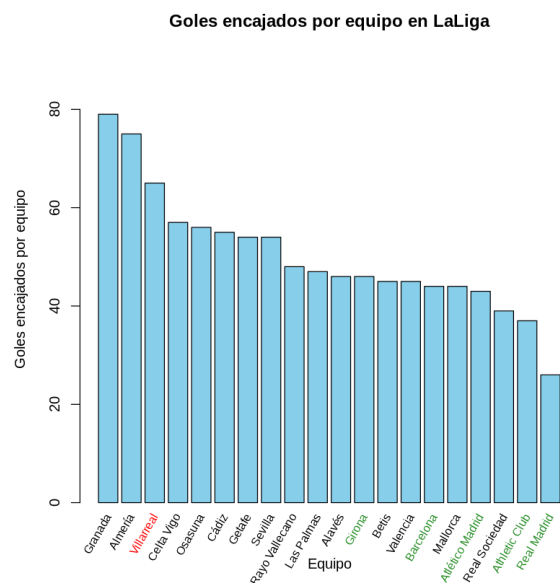


Figura 8: De las peores defensas de LaLiga

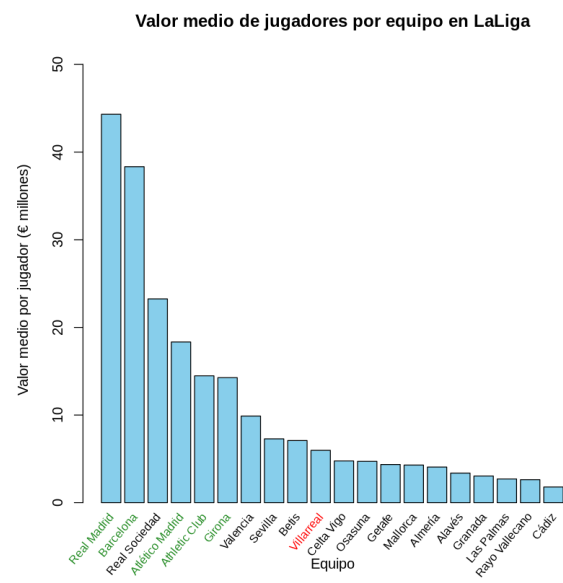


Figura 9: Valor de mercado normal

A partir de las figuras 9 y 10 podemos sacar conclusiones, como que el hecho de que el equipo esté en una posición tan mala, es debido a que tiene una de las peores defensas de la competición, junto a que el valor medio de los jugadores de la plantilla es mediocre, comparado con los mejores equipos de LaLiga.

Detección de valores atípicos:

La detección de los casos atípicos es importante ya que son observaciones que son significativamente diferentes al resto y esto puede perjudicar el buen funcionamiento de los modelos de *Machine learning*. Esto puede ser de gran utilidad, por ejemplo, nos puede servir para detectar qué jugadores destacan sobre el resto en algún aspecto.

En particular, definimos la variable aleatoria X , como el **valor de mercado** de un jugador, y consideramos una muestra aleatoria simple de tamaño n compuesta por todos los **defensores** del *dataset*, X_1, X_2, \dots, X_n .

De este modo, podemos construir un **intervalo de confianza con cota superior** para el valor de mercado medio de los defensores θ , con un nivel de confianza del 0.9, utilizando la siguiente fórmula:

$$P_{\theta}(T_2(X_1, \dots, X_j) \geq \theta) = 0,9 \quad (6)$$

Para detectar los *outliers* (valores atípicos) hay varias técnicas, las más básicas son las distribuciones estadísticas y los percentiles. En la **Figura 10**, los casos atípicos eran los que pertenecían a un $Precentil_{0,9}$ (valores superiores al 90 %), ya que la variable presenta una distribución asimétrica positiva.

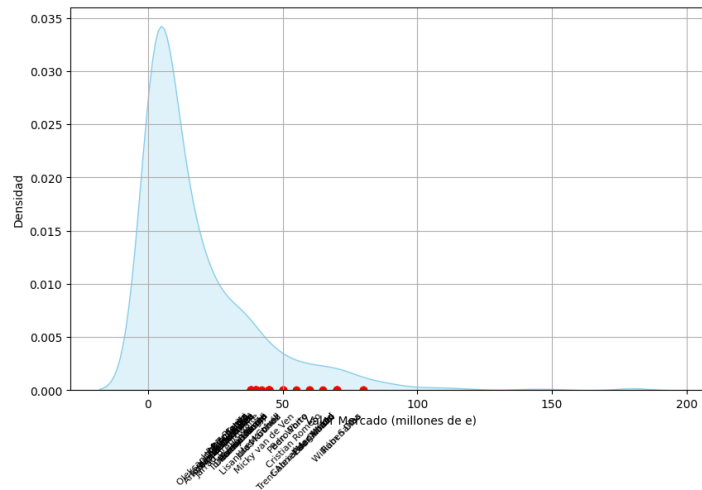


Figura 10: Esta figura muestra los defensores más caros de nuestro *dataset*

Para definir **defensas caros** y **defensas normales**, hemos usado la variable *Valor de Mercado* hemos diferenciado los defensores en el percentil 0,9 (caros) y los otros (normales). Y esto lo hemos usado para encontrar las variables para crear la KPI *Rendimiento defensivo*. En la **Figura 11** podemos observar las variables defensivas donde más destacan los defensas caros.

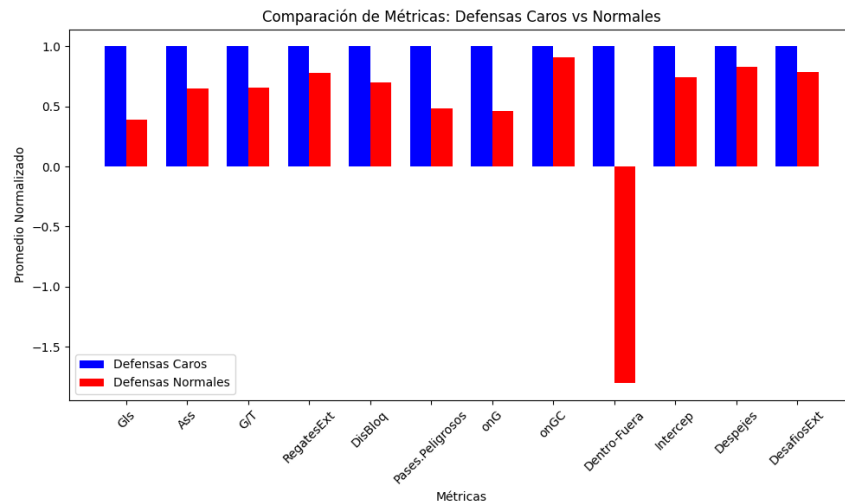


Figura 11: Variables más importantes a tener en cuenta para ser un buen defensa

Para poder encontrar los peores jugadores del equipo vamos a definir una nueva variable **Dentro-Fuera**.

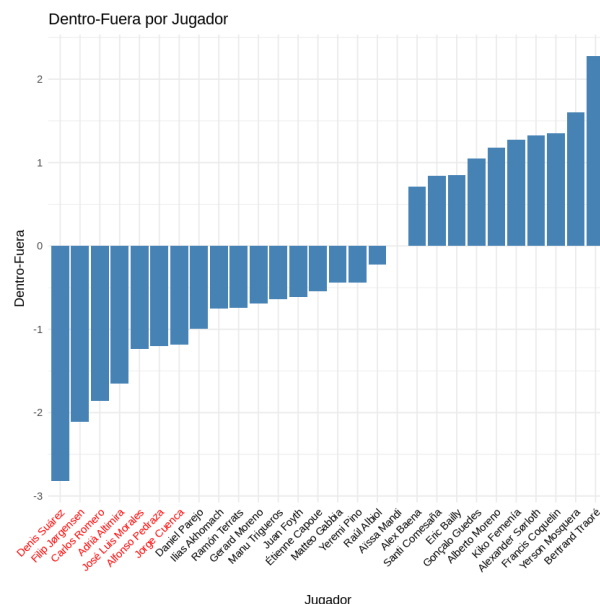


Figura 12: Jugadores con menos *Dentro-Fuera*

De esta forma se va a decidir tomar medidas y deshacernos de los jugadores: Denis Suárez, Filip Jörgensen, Carlos Romero, Adrià Altimira, José Luis Morales, Alfonso Pedraza y Jorge Cuenca.

Así que tras la finalización de contrarto de los jugadores sin un valor de mercado importante, y la venta de aquellos que tenían una valor de mercado interesante, nuestro presupuesto actualizado es de **63 M€**

3.2. Análisis de correlaciones

En este apartado, realizaremos un estudio acerca de la covarianza y la correlación de variables con el objetivo de estudiar el grado de dependencia entre ellas y los posibles aumentos o disminuciones que se producen en una variable como resultado del incremento o reducción de otra.

Así podremos sacar ciertas conclusiones de dependencias entre variables para detectar las variables que más nos interesan a la hora de fichar jugadores con una buena proyección de valor de mercado.

Matriz de covarianza

La covarianza es una medida estadística que indica el grado y la dirección de la relación lineal entre dos variables. Para un conjunto de variables aleatorias X_1, X_2, \dots, X_p , la matriz de covarianza \mathbf{V} es una matriz simétrica $p \times p$ cuyos elementos son las covarianzas entre cada par de variables. La covarianza entre las variables X_i, X_j de la matriz viene dada por:

$$\text{Cov}(X_i, X_j) = \frac{1}{n} \sum_{k=1}^n (x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j) \quad (7)$$

donde \bar{x}_i y \bar{x}_j son las medias de las variables X_i y X_j , respectivamente. La diagonal de la matriz contiene las varianzas de cada variable $\text{Var}(X_i) = \text{Cov}(X_i, X_i)$, mientras que los elementos fuera de la diagonal indican la covarianza entre pares de variables. Una covarianza positiva sugiere que las variables tienden a aumentar juntas, mientras que una covarianza negativa indica que una aumenta cuando la otra disminuye. Sin embargo, la magnitud de la covarianza depende de las escalas de las variables, lo que dificulta su interpretación.

Análisis de correlación de variables

La correlación extiende el concepto de covarianza al estandarizar los valores, permitiendo medir la intensidad de la relación lineal entre dos variables independientemente de sus escalas. La matriz de correlación se deriva de la matriz de covarianza \mathbf{V} , donde cada elemento $\text{Corr}(X_i, X_j)$ se calcula como:

$$r = \text{Corr}(X_i, X_j) = \frac{\text{Cov}(X_i, X_j)}{\sqrt{\text{Var}(X_i) \cdot \text{Var}(X_j)}} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \quad (8)$$

El coeficiente de correlación r varía entre -1 y 1 , siendo los valores cercanos a 1 indican

una fuerte correlación positiva, cercanos a -1 una fuerte correlación negativa, y cercanos a 0 una correlación insignificante. Para visualizar estas relaciones, se puede emplear un mapa de calor que resalte las variables con mayor correlación, facilitando la identificación de patrones en los datos.

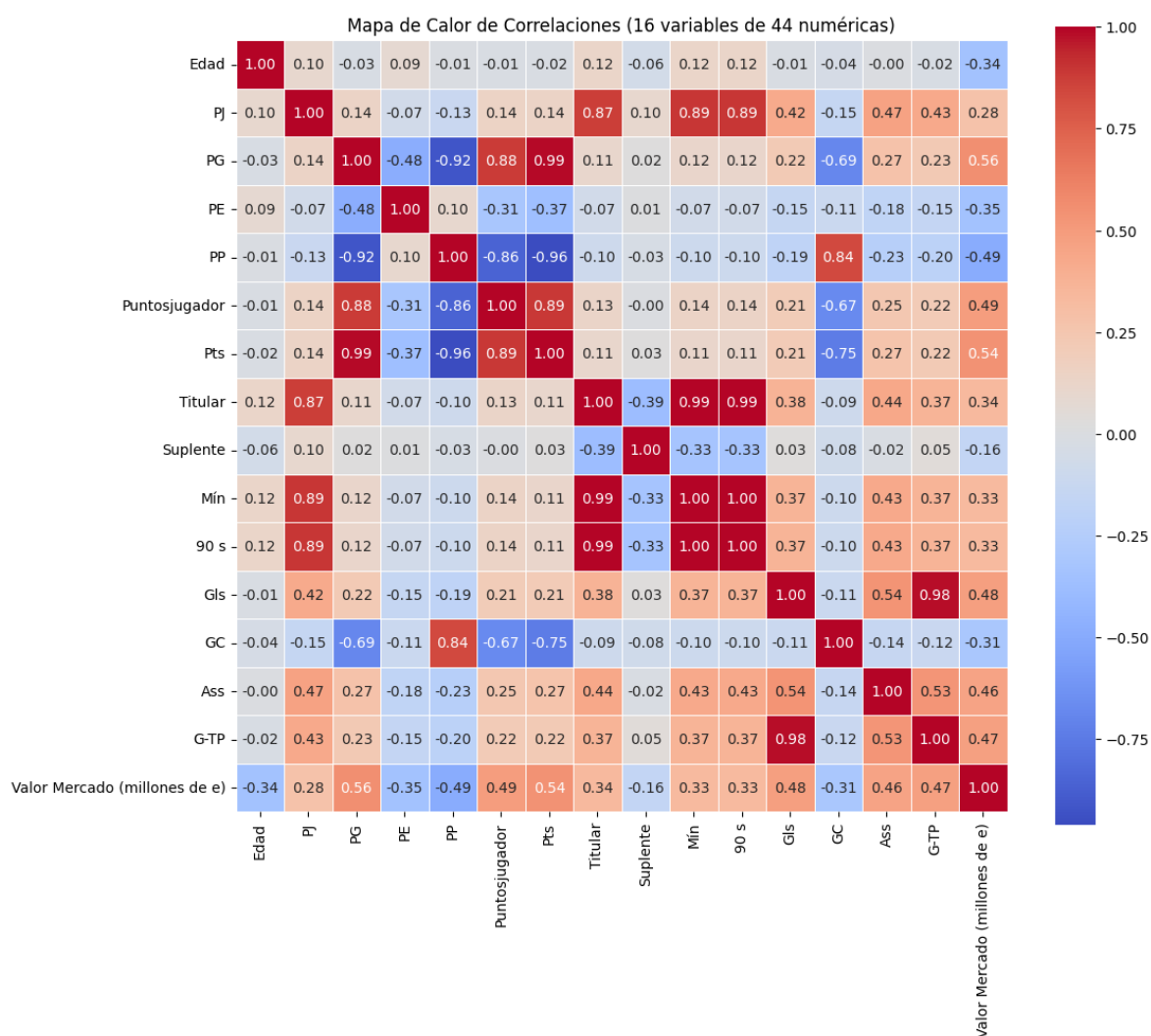


Figura 13: Mapa de calor para la correlaciones.

Tomando la variable *Valor Mercado (millones de e)*, podemos deducir qué variables, características o estadísticas influyen en el valor de un jugador. Podemos observar que variables como PG, Gl's y Ass influyen positivamente en el valor de mercado del jugador. En cambio, variables como PP y GC influyen negativamente. Otra de las variables que influye negativamente es Edad, y esto es debido a que cuanto más veterano sea un jugador, menos demanda tiene en el mercado de fichajes y por tanto, menor será su valor.

4. Modelos predictivos de ML

El **aprendizaje automático** (del inglés, **Machine Learning**) es una rama de la **inteligencia artificial** (IA) que tiene como objetivo desarrollar técnicas que permitan a las computadoras crear modelos capaces de realizar predicciones o diferenciar elementos según sus características.

En esta sección vamos a trabajar en una de las partes más importantes de nuestro estudio, donde vamos a utilizar los distintos modelos de ML supervisado. Con el objetivo de predecir métricas de rendimiento ofensivo y defensivo, y así poder seleccionar un conjunto de jugadores para la *shortlist*.

4.1. Métricas de evaluación

A fin de ver qué tan bien “funcionan” los modelos, vamos a usar métricas que nos indiquen la calidad y desviación de nuestras predicciones \hat{y} , y lo bien que han sido entrenados. Siguiendo el objetivo de nuestro trabajo, el cual consiste principalmente en usar modelos de regresión, las métricas que nos van a interesar serán las métricas que tienen que ver con la regresión. En el apéndice se verán las métricas que tienen que ver con los modelos de clasificación (precisión, sensibilidad, AUC, ROC, etc.).

- **Error Cuadrático Medio (MSE)**: Mide el promedio de los errores al cuadrado entre los valores observados y_i y los valores predichos \hat{y}_i . Penaliza más los errores grandes.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (9)$$

La naturaleza cuadrática de esta métrica hace que la presencia de los valores atípicos influya demasiado en el error.

- **Error Absoluto Medio (MAE)**: Calcula el promedio de los errores absolutos entre los valores observados y los predichos. Es más robusto frente a valores atípicos que el MSE.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (10)$$

- **Error Porcentual Absoluto Medio (MAPE)**: Calcula el promedio del error absoluto expresado como porcentaje relativo respecto al valor real observado. Es

útil para interpretar el error en términos porcentuales.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (11)$$

Puede ser problemático si existen valores cercanos a cero, debido a que el denominador del sumatorio hace que el error tienda al infinito.

- **Coeficiente de Determinación (R^2):** Indica qué proporción de la variabilidad de la variable dependiente es explicada por el modelo. Su valor varía entre 0 y 1, teoreicamente.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (12)$$

Si $R^2 = 1$, la predicción es perfecta, si $R^2 = 0$, el modelo predice igual que la media. En el caso que $R^2 < 0$, el modelo predice peor que la media.

Validación cruzada:

Otro tipo de errores muy frecuentes de los modelos es el caracterizado por el ajuste de los modelos a características específicas de los datos distintas al objeto de estudio, también conocido como *overfitting* o sobreajuste. O en caso contrario, cuando el modelo no logra capturar la relación entre las variables objetivo de estudio, conocido como *underfitting*, dando de esta forma predicciones no precisas.

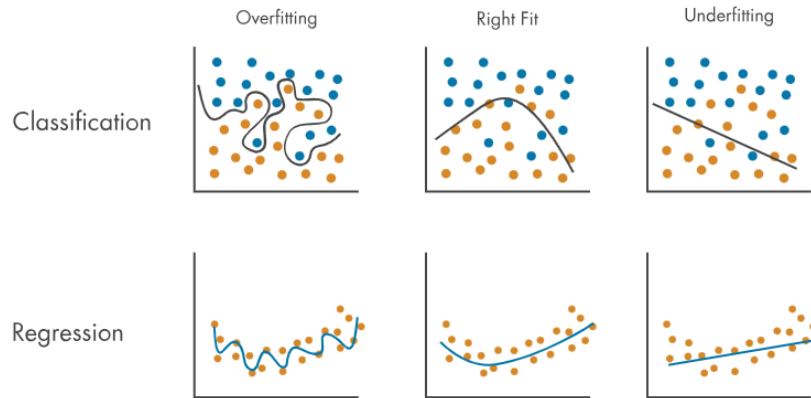


Figura 14: Esquema visual del ML.

Es por ello que la aplicación de estrategias como la validación cruzada adquiere importancia y se hacen las transformaciones necesarias para poder desarrollar estudios estadísticamente representativos.

Dicho esto, la validación cruzada consiste en un método para estudiar qué tan bien el modelo aprende de los datos de entrenamiento y logra generalizar bien con datos nuevos. Y consiste en aplicar los siguientes pasos:

1. **División entre conjunto de entrenamiento, validación y test:**

Para comprobar qué tan bien predice el modelo, podemos utilizar el conjunto de entrenamiento *train* para así ver si aprende bien el conjunto. El conjunto de datos de entrenamiento incluye datos de entrada y valores de respuesta. Para la división en train, validation, test hay que tener en cuenta el tamaño del *datasets*, los casos más habituales suelen ser divisiones de 60/20/20 o 80/10/10%.

2. **Entrenar el modelo con el conjunto *train*.**

3. **Comprobar qué tan bien funciona el modelo con *test*.**

Ver el apéndice para algunas técnicas de validación cruzada.

Modelos de regularización

Cuando tenemos un modelo con problemas de *overfitting* necesitamos regularizarlo, es decir, hacer el modelo menos complejo. Para ello necesitamos:

Regularización L1 (Lasso): Este método consiste en añadir una penalización proporcional a la suma de los valores absolutos de los coeficientes del modelo. Favorece modelos más simples, ya que puede hacer que algunos coeficientes se vuelvan exactamente cero. Aunque es muy popular en Regresión Lineal, también se puede aplicar en otros GLM y variantes adaptadas en aprendizaje automático

$$\text{LASSO} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (13)$$

Regularización L2 (Ridge): Añade una penalización proporcional a la suma de los cuadrados de los coeficientes. Tiende a reducir el tamaño de los coeficientes, pero no los lleva exactamente a cero.

$$\text{Ridge} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (14)$$

Problemas de underfitting.

Cuando tenemos *underfitting* tenemos que evitar usar regularizaciones, una forma de detectar *underfitting*, es la siguiente:

Bias: Este error ocurre cuando un modelo no se entrena lo suficiente mente bien o cuando el modelo es demasiado simple para capturar la complejidad o las variaciones en los datos.

Variance: Error por variabilidad excesiva en las predicciones del modelo, causado por la sensibilidad del modelo a las fluctuaciones en el conjunto de datos de entrenamiento

4.2. Métodos lineales generalizados (GLM)

El **aprendizaje automático supervisado** se le llama a la técnica que emplea un conjunto de datos conocidos (el denominado conjunto de datos de entrenamiento o **train**) para realizar predicciones con nuevos datos desconocidos. A partir de él, el algoritmo de aprendizaje supervisado busca crear un modelo que pueda realizar predicciones acerca de los valores de respuesta para un nuevo conjunto de datos.

Dentro de los algoritmos o modelos supervisados destacamos los de clasificación y los de regresión. En este trabajo, dada la naturaleza del objetivo de este trabajo, nos centraremos sobre todo en los modelos de regresión.

Los modelos lineales generalizados pertenecen a la familia de modelos supervisados. El objetivo de estos modelos es predecir una **variable dependiente** Y a partir de p variables conocidas como **variables explicativas** o **independientes** X_1, X_2, \dots, X_p .

$$g(E(Y | X = x)) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p \quad (15)$$

Donde g es conocida como **función link**, una función monótona y diferenciable que sirve para... Y los coeficientes $\beta = \beta_1, \beta_2, \dots, \beta_p$, representan el peso de la variable a la que van asociados.

| Distribución de Y | Modelo | Función Link |
|---------------------|---------------------|--------------|
| Gaussiana | Regresión Lineal | Identidad |
| Bernoulli | Regresión Poisson | Logit |
| Poisson | Regresión Logística | Logaritmo |

Tabla 2: Principales modelos GLM

Regresión lineal

El principal modelo predictivo y más conocido es el modelo de regresión lineal, el cual es un modelo de regresión, y tiene la siguiente fórmula:

$$Y = E(Y | X = x) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon \quad (16)$$

Donde:

- β_0 : Es el *intercept*, se corresponde con la media de y cuando todas las variables explicativas son cero.
- β_i : Estos coeficientes, β_i nos ayudan a interpretar el modelo. Si $\beta_i > 0$ hay una correlación positiva entre la variable X_i y la variable explicada Y , si $\beta_i < 0$ hay una correlación negativa entre la variable X_i y la variable explicada. Y si es cercana a cero la variable se dice que no es muy significativa.
- ε : Es el término de error, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

Para obtener la KPI **Rendimiento Ofensivo**, hemos hecho un estudio análogo al de los defensas (**Fig 10 y 11**). Y hemos obtenido que las variables donde más destacan los delanteros más caros son: “Gls”, “Ass”, “Pases.Peligrosos”, “G/T”, “RegatesExt”, y para ver la importancia/peso de cada una vamos a utilizar Regresión Lineal.

Los bhetas que salen son: $\beta_1 = 0,44$, $\beta_2 = -0,13$, $\beta_3 = 0,4$, $\beta_4 = -0,01$, $\beta_5 = 0,02$, dado que RL no capta bien las relaciones no lineales, nos salen cosas como que las asisitencias tienen un bhetas negativo algo que no tiene sentido, en este contexto. Todos estos resultados los podemos encontrar en el repositorio de este trabajo.

4.3. Weak learners

En esta sección vamos a introducir las bases de muchos modelos ensemble y estos son los *weak learners*. El principal modelo son los árboles de decisión simples.

Árbol de decisión (CART)

Un árbol de decisión o CART [7] (del inglés *classification and regression tree*) es un algoritmo de decisión que se basa en una gráfica. Cada nodo representa una condición, cada nodo tiene una rama con una afirmación y otra con su contradicción. Pretende buscar de entre todas las variables, cuál es la mejor variable en el mejor corte respecto de su rango, que divide los datos en dos subconjuntos mejor separados respecto de la variable *target*.

Estos modelos presentan las siguientes ventajas: No depende de la distribución de la *target*. Captura relaciones no lineales y efectos de interacción. Mayor interpretabilidad, gracias al dibujo del árbol. Respecto a las principales **desventajas**, tenemos que no extrapolan a la variable *target*. No sabe predecir bien con valores fuera de rango (con los que no ha sido entrenado).

4.4. Modelos ensemble

Los ensembles hacen referencia a la unión de varios modelos de predicción de Machine Learning con el objetivo de obtener mejores predicciones que las resultantes de un modelo individual. Estos modelos suelen tener problemas de interpretabilidad, debido a la combinación de modelos.

Random forest

Random forest es un algoritmo que usa la técnica de *Bagging*, para combinar el resultado de los distintos árboles de decisión y así dar una respuesta más robusta. Sigue la siguiente expresión:

$$RF(x) = \frac{1}{N} \sum_{j=1}^N T_j(x) \quad (17)$$

El ***Bagging*** es una técnica que usa el ***Bootstrap*** con el fin de reducir la varianza de algunos métodos de aprendizaje estadístico, entre ellos los árboles de decisión.

La regresión lineal fracasa al modelar relaciones no lineales como la edad y el rendimiento.

Sin embargo, los modelos basados en árboles de decisión, como *Random Forest*, permiten particionar el dominio de la variable predictora y capturar comportamientos complejos de forma precisa, sin requerir transformaciones explícitas de los datos.

Para cada árbol vamos a permitir un límite de variables que puede usar de las p variables que tenemos en total, vamos a permitir que cada árbol use \sqrt{p} variables. Esa es la cantidad ideal de variables para cada árbol de decisión del *Random Forest*.

Básicamente, el modelo sigue los siguientes pasos:

1. Hacer uso de *Bootstrap* para crear nuevos conjuntos de datos.
2. Se crea un árbol de decisión con cada conjunto de datos, obteniendo diferentes árboles, ya que cada conjunto contiene diferentes observaciones.
3. Al crear los árboles se eligen variables al azar en cada nodo del árbol, dejando crecer el árbol en profundidad (sin podar).
4. Se recogen los resultados obtenidos de todos los árboles del bosque (sin podar).

En **regresión**, obtenemos un valor predicho y_i de cada árbol del bosque y la predicción final será la media de todos los valores predichos por cada árbol.

Dado que la Regresión lineal dió un mal modelo, debido a su incapacidad de captar relaciones no lineales. Vamos a usar los modelos basados en CART, debido a la capacidad de estos de captar relaciones no lineales entre las variables.

Usamos el modelo de *Random Forest* y obtenemos los siguientes coeficientes, que representan la importancia de cada variable en el modelo: $w_1 = 0,43$, $w_2 = 0,1$, $w_3 = 0,29$, $w_4 = 0,08$, $w_5 = 0,1$.

Y así, obtenemos los coeficientes para construir la variable Rendimiento_Ofensivo, que vimos al principio del trabajo en la **Fig 4**. De manera análoga obtenemos la expresión del Rendimiento_Defensivo (ver repositorio).

Definición 4.1 Decimos que p_k es una **dirección de descenso** de una función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ en el punto x_k , si se cumple:

$$f'(x_k; p_k) = \nabla f(x_k)^T p_k < 0. \quad (18)$$

Dada una función $f : \mathbb{R}^n \rightarrow \mathbb{R}$, a partir de un valor inicial x_0 y una dirección de descenso p_k , obtenemos de manera iterativa la sucesión $\{x_k\}_k$ mediante la fórmula:

$$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0, 1, 2, \dots \quad (19)$$

donde α_k es un escalar positivo que determina el tamaño del paso en cada iteración.

Cuando el método converge, la sucesión $\{x_k\}_k$ converge a un mínimo local de la función f . En particular, en el método de **descenso del gradiente** la dirección de descenso se define como:

$$p_k = -B_k^{-1} \nabla f_k, \quad (20)$$

donde B_k son matrices simétricas y no singulares. En el caso específico del descenso del gradiente clásico, B_k es, la matriz identidad.

XGBoost

El XGBoost, en inglés (*eXtreme Gradient Boosting*) es un modelo que utiliza árboles de decisión como *Random forest* y el reforzamiento de gradientes. Este modelo emplea el proceso de **boosting**, una técnica que busca mejorar las predicciones de los modelos previos. En primer lugar, se entrena un modelo y el modelo siguiente intenta reducir el error del modelo anterior para lograr finalmente, alcanzar un umbral de tolerancia aceptable. Sea un conjunto de datos $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, donde $\mathbf{x}_i = (x_i^1, \dots, x_i^m)$ son vectores de características y $y_i \in \mathbb{R}$ es la variable target, el objetivo es minimizar una función de pérdida $L(y, F(\mathbf{x}))$. La estructura de los modelos XGBoost sigue estos pasos:

1. Se inicializa un modelo base \hat{y}_{base} , un árbol de decisión, que busca predecir la variable objetivo y .
2. Cada árbol sucesivo se ajusta a los residuos ($\hat{\epsilon}_i = y_i - \hat{y}_i$) del modelo anterior, minimizando la función de pérdida. En nuestro caso para regresión será:

$$L(y_i, \hat{y}_i) = \frac{1}{2}(y_i - \hat{y}_i)^2 \quad (21)$$

cuya derivada respecto a \hat{y}_i es

$$\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} = -(y_i - \hat{y}_i) = -\hat{\epsilon}_i. \quad (22)$$

y aplicando el descenso del gradiente con un tamaño de paso η , la predicción se actualiza de la siguiente forma:

$$\hat{y} = \hat{y}_{base} + \eta \sum_{i=1}^N \hat{\epsilon}_i = \hat{y}_{base} + \eta \sum_{i=1}^N \text{ErrorTree}_i. \quad (23)$$

3. Para controlar el *overfitting*, se incorporan parámetros de regularización:

- γ : Especifica la reducción mínima de la función de pérdida requerida para dividir un nodo, controlando la complejidad del árbol.
- λ : Regularización L2 sobre los pesos de los nodos hoja, basada en el error promedio al cuadrado.

La función de pérdida total incluye estos términos:

$$L(y_i, \hat{y}_i) = \text{Error}_{\hat{y}_{base}} + \eta \sum_{i=1}^N \text{ErrorTree}_i + \gamma \cdot N_{\text{hojas}} + \frac{1}{2} \lambda \sum_{j=1}^{J_{\text{hojas}}} \text{ErrorPromedioHoja}_j^2 \quad (24)$$

Valores altos de γ o λ reducen el número de hojas, promoviendo el *underfitting*.

4. El proceso se repite, añadiendo árboles hasta alcanzar un número máximo N o detenerse mediante *early stopping* si la métrica de validación no mejora tras un número de iteraciones.

Siguiendo nuestro objetivo, vamos a comparar el rendimiento de los modelos para ver cuál sería el modelo que predice mejor el **rendimiento defensivo**. Las variables explicativas empleadas son las que han sido citadas anteriormente. La métrica que más nos va a interesar analizar será el **error absoluto medio parcial (MAPE)**, por su facilidad de interpretación. La *Figura 15* compara de una forma visual y superficial los errores del *train* y *test* de los distintos modelos. Además, nos servirá para detectar la presencia de *overfitting* y *underfitting*.

LightGBM

El LightGBM (*Light Gradient Boosting Machine*) es un modelo basado en árboles de decisión que mejora la eficiencia de XGBoost, reduciendo la complejidad computacional y el uso de memoria, especialmente en conjuntos de datos grandes. Para reducir la complejidad, LightGBM implementa dos técnicas principales [10]:

- **Gradient-based One-Side Sampling (GOSS)**: Selecciona instancias con gradientes altos ($\text{top } a \times 100\%$) y muestrea aleatoriamente un subconjunto ($b \times 100\%$) de

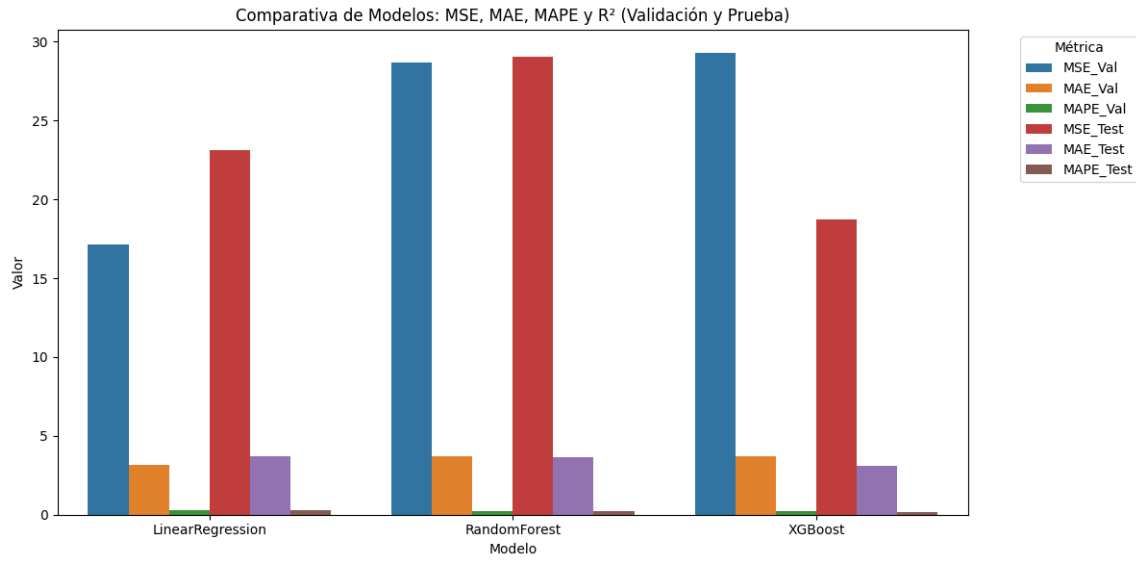


Figura 15: Comparación de modelos

instancias con gradientes bajos, amplificando su contribución por $\frac{1-a}{b}$ al calcular la ganancia de información:

$$\tilde{V}_j(d) = \frac{1}{n} \left(\frac{(\sum_{x_i \in A_l} g_i + \frac{1-a}{b} \sum_{x_i \in B_l} g_i)^2}{n_l^j(d)} + \frac{(\sum_{x_i \in A_r} g_i + \frac{1-a}{b} \sum_{x_i \in B_r} g_i)^2}{n_r^j(d)} \right) \quad (25)$$

donde A_l, A_r, B_l, B_r son subconjuntos de instancias según el punto de división d . Esto reduce el costo computacional manteniendo la precisión.

- **Exclusive Feature Bundling (EFB):** Agrupa características exclusivas en un solo *bundle* para reducir el número de características, asignando valores con desplazamientos para evitar conflictos, optimizando el cálculo en histogramas.

CatBoost

Como su nombre indica, CatBoost (**C**ategorical **B**oosting), este modelo presenta una ventaja respecto del modelo anterior, y esta es que permite utilizar las variables categóricas como variables explicativas.

Algo que la mayoría de modelos de ML no logra debido a que las variables categóricas suelen requerir una transformación numérica para ser procesadas por estos algoritmos. Una técnica común para esto es la codificación one-hot, que genera una nueva variable binaria por cada categoría, aumentando significativamente la dimensionalidad. CatBoost

aborda este problema, primero las convierte en características numéricas mediante **Target Statistics (TS)**, y luego mediante el uso de *Ordered Target Statistics* (OTS) [11] un método que convierte las variables categóricas en valores numéricos sin expandir la dimensionalidad. Este enfoque calcula estadísticas basadas en la variable objetivo (*target*) de manera ordenada, evitando el problema de *target leakage* (fuga de información del objetivo), que ocurre cuando las estadísticas se calculan usando el propio valor objetivo del ejemplo en cuestión, llevando a un sobreajuste.

Sea un conjunto de datos $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1}^n$, donde $\mathbf{x}_k = (x_k^1, \dots, x_k^n)$ son vectores de características y $y_k \in \mathbb{R}$ es la variable objetivo. El proceso de OTS funciona como sigue, para un conjunto de datos de entrenamiento con n ejemplos, CatBoost genera una permutación aleatoria σ de los datos. Para cada ejemplo \mathbf{x}_k , el valor numérico asignado a una variable categórica x^i se calcula considerando solo los ejemplos previos en la permutación, definidos por el subconjunto $\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$. La fórmula de OTS es:

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{I}_{\{x_j^i = x_k^i\}} \cdot y_j + a \cdot p}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{I}_{\{x_j^i = x_k^i\}} + a} \quad (26)$$

donde:

- $\mathbb{I}_{\{x_j^i = x_k^i\}}$ es un indicador que vale 1 si $x_j^i = x_k^i$ (es decir, si los ejemplos comparten la misma categoría), y 0 en caso contrario.
- y_j es el valor objetivo del ejemplo \mathbf{x}_j .
- $p = \mathbb{E}(y)$ es la media global del objetivo, usada como valor inicial o *prior*.
- $a > 0$ es un parámetro de suavizado que estabiliza las estimaciones cuando el número de ejemplos previos es pequeño y reduce el impacto de categorías poco frecuentes.

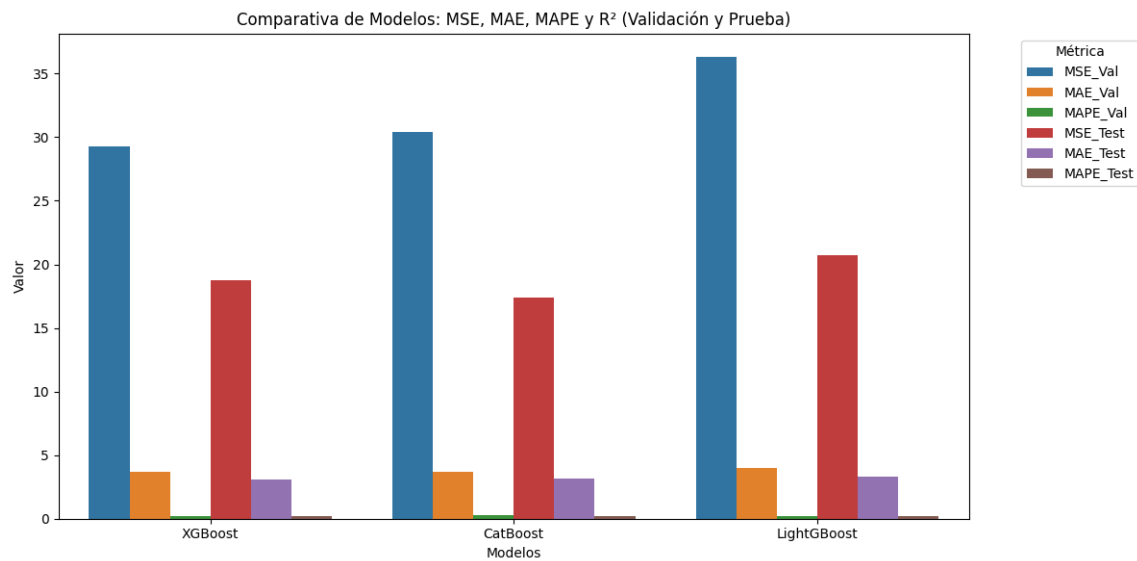


Figura 16: Podemos ver que esta capacidad de utilizar variables categóricas hace que tenga un menor error MSE del test. En las siguientes tablas veremos en detalle qué modelo predice mejor.

| TEST | MAPE | MSE |
|---------------|------|-----|
| Reg. lineal | 27 % | 23 |
| Random Forest | 20 % | 29 |
| XGBoost | 18 % | 18 |

Figura 17: Errores del test

| TEST | MAPE | MSE |
|--------------|------|-----|
| Rand. Forest | 20 % | 29 |
| XGBoost | 18 % | 18 |
| CatBoost | 21 % | 17 |

Figura 18: Errores del test

A pesar de que el CatBoost nos proporciona el mejor error MSE, el modelo XGBoost sigue siendo el modelo que nos proporciona un mejor error absoluto medio parcial (MAPE). Y por tanto, podemos decidir que este modelo es el mejor, hasta ahora.

5. Modelos preentrenados

5.1. Redes neuronales.

Una red neuronal es un algoritmo matemático que tiene como objetivo recrear el funcionamiento del cerebro humano, haciendo uso de neuronas artificiales en lugar de la neuronas biológicas. Las neuronas artificiales planteadas funcionan de la siguiente manera: la neurona artificial recibe una o más entradas binarias y activa su salida en función del número de entradas activas que recibe. Al conjunto de redes neuronales interconectadas le llamamos red de neuronas artificiales (ANN).

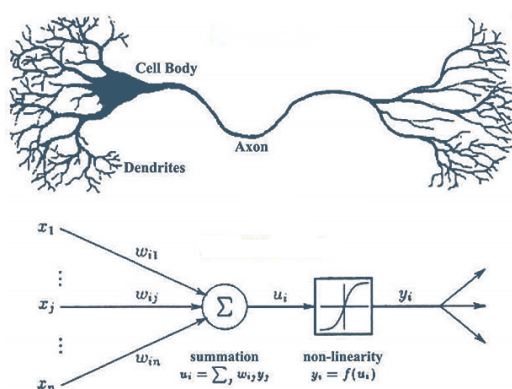


Figura 19: Neurona biológica vs Neurona artificial.

Deep learning (Aprendizaje profundo)

El deep learning es un método de aprendizaje automático que emplea redes neuronales profundas, es decir, muchas capas intermedias, para extraer patrones complejos a partir de grandes cantidades de datos. A diferencia de las redes neuronales más simples, este enfoque permite capturar relaciones altamente no lineales, lo que lo hace especialmente potente en tareas como la visión artificial o el procesamiento del lenguaje. Uno de los hitos más relevantes dentro del deep learning es la aparición del **modelo Transformer**, con un mecanismo basado completamente en la atención [8][13].

Arquitectura de las redes neuronales

Como hemos visto anteriormente, llamamos al conjunto de neuronas artificiales, **red neuronal**. La arquitectura de las redes neuronales multicapa se conoce como *feedforward* (redes neuronales prealimentadas) de forma que la información fluye en un sentido, desde la capa de entrada hacia la capa de salida. Cada neurona de una capa está conectada a todas las neuronas de la capa siguiente, garantizando una transmisión completa de

información entre capas consecutivas. De este modo, las redes neuronales siguen la siguiente estructura:

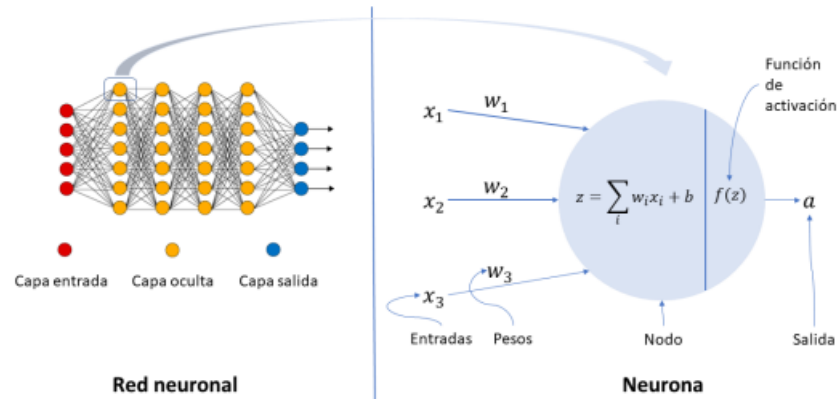


Figura 20: Ilustración de una red neuronal y sus componentes.

Los componentes de una red neuronal son las neuronas, que siguen la siguiente estructura [12]:

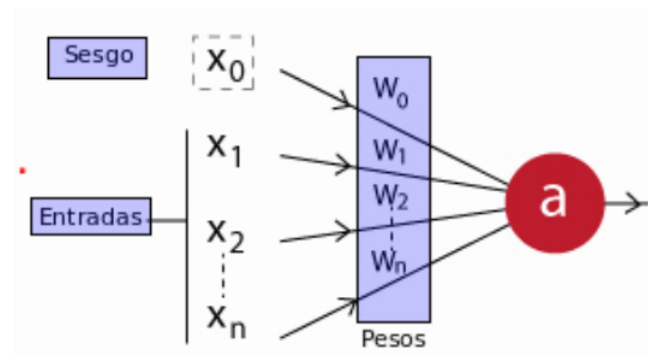


Figura 21: Ilustración de las componentes de una neurona

Donde:

- x_1, \dots, x_n : Los datos de entrada en la neurona, los cuales también puede ser que sean producto de la salida de otra neurona de la red.
- x_0 : La unidad de sesgo; un valor constante que se le suma a la entrada de la función de activación de la neurona. Generalmente tiene el valor 1. Este valor va a permitir cambiar la función de activación hacia la derecha o izquierda, otorgándole más flexibilidad para aprender a la neurona.

- w_1, \dots, w_n : Los pesos relativos de cada entrada. Y actúan de forma análoga a los pesos en Regresión Lineal. Tener en cuenta que la unidad de sesgo tiene un peso.
- a : La salida de la neurona.

Así podemos expresar una neurona con la fórmula $f(\sum_{i=0}^n w_i \cdot x_i) = a$, donde f es una función de activación de la neurona.

Función de activación

La función de activación sirve para transformar la salida lineal de una neurona en una salida no lineal. Matemáticamente, si denotamos la salida lineal de la neurona como z , entonces la salida activada es:

$$a = \phi(z) \quad (27)$$

donde ϕ es la función de activación elegida.

Su objetivo es introducir no linealidad al modelo, permitiendo que la red neuronal aprenda funciones complejas. Y su utilidad viene de que, sin funciones de activación no lineales, una red neuronal con varias capas sería equivalente a una sola capa lineal, una única neurona artificial, ya que la composición de funciones lineales es una función lineal.

Ejemplos de funciones de activación:

- **ReLU (Rectified Linear Unit):**

$$\phi(z) = \max(0, z) \quad (28)$$

- **Sigmoide:**

$$\phi(z) = \frac{1}{1 + e^{-z}} \quad (29)$$

- **Tangente hiperbólica (tanh):**

$$\phi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (30)$$

Cada una de estas funciones tiene propiedades específicas, por ejemplo, ReLU es computacionalmente eficiente y ayuda a evitar el problema de gradientes desvanecidos en capas profundas [8] [12].

▪ **Softmax:**

$$\text{softmax}(z) = \left(\frac{e^{z_1}}{\sum_{k=1}^n e^{z_k}}, \dots, \frac{e^{z_n}}{\sum_{k=1}^n e^{z_k}} \right), \quad z \in \mathbb{R}^n. \quad (31)$$

Esta función tiene una especial importancia en los modelos transformer que veremos más adelante [9].

Propagación hacia atrás

La propagación hacia atrás o backpropagation es un algoritmo que funciona mediante la determinación de la pérdida en la salida y luego propagándola de nuevo hacia atrás en la red, por medio del mecanismo de descenso de gradiente. De esta forma, los pesos se van actualizando para minimizar el error resultante de cada neurona. Este algoritmo es lo que les permite a las redes neuronales aprender [12][13]. El objetivo es encontrar los pesos que minimicen la función de pérdida, definida como el error cuadrático medio:

$$\min_w L(w) = \min_w \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i(w))^2 \quad (32)$$

donde y_i es el valor real y $\hat{y}_i(w)$ es la predicción de la red neuronal para el ejemplo i , dependiente de los pesos w . A partir de esta función de pérdida, los pesos se actualizan mediante el descenso de gradiente según:

$$w = w - \eta \cdot \nabla_w L \quad (33)$$

w : Vector de pesos de la red neuronal que se actualiza en cada iteración.

η : Tasa de aprendizaje (*learning rate*), un escalar positivo que controla el tamaño de los pasos en la actualización de pesos.

$\nabla_w L$: Gradiente de la función de pérdida L con respecto a los pesos w , es decir, un vector que indica la dirección de máximo incremento de L en el espacio de pesos.

Funcionamiento de una red neuronal

El uso de esta estructura de capas en la red neuronal, como se ve en la figura 20, se puede justificar debido a que la red aprende algo simple en la capa inicial de la jerarquía y luego envía esta información a la siguiente capa. La siguiente capa toma esta información simple, lo combina en algo que es un poco más complejo, y lo pasa a la tercera capa. Este proceso continúa de forma tal que cada capa de la jerarquía construye algo más complejo

de la entrada que recibió de la capa anterior. De esta forma, la red irá aprendiendo por medio de la exposición a los datos de ejemplo.

Tipos de redes neuronales

Desde que surgieron las redes neuronales no se ha parado de mejorar, optimizar y adaptar su estructura a las diferentes necesidades. A continuación, tenemos las principales redes neuronales clásicas.

- **Perceptron:** Son las principales redes neuronales prealimentadas, las cuales fueron las primeras que se desarrollaron y son el modelo más sencillo.
- **CNN:** las CNN (Convolutional Neural Network) surgieron para el procesamiento de datos visuales.
- **RNN (Recurrent Neural Networks):** Para el procesamiento de texto las arquitecturas de redes neuronales que se utilizaron fueron las RNN.
- **LSTM (Long Short-Term Memory):** Las LSTM han permitido resolver una de las principales limitaciones de las redes neuronales clásicas, como el concepto de contexto, lo que nos permitió que, en lugar de analizar las palabras una por una, las oraciones podían analizarse en un contexto muy específico..

5.1.1. Modelos Transformer

Las redes neuronales son modelos muy eficientes para analizar datos complejos en distintos formatos. Sin embargo, estas presentan ciertas limitaciones, como:

- Memoria demasiado corta para procesar textos demasiado largos.
- Las RNN procesan los datos secuencialmente y, por tanto, son difíciles de paralelizar. Sin embargo, los mejores modelos de lenguaje actuales se caracterizan por una cantidad astronómica de datos. Entrenar un modelo LSTM con los datos consumidos por GPT-3 habría llevado décadas.

Aquí entran los modelos **Transformers**, los cuales presentan la siguiente ventaja principal: que son fácilmente paralelizables, lo que hace que el entrenamiento de *datasets* grandes sea más rápido. Por ejemplo, GPT-3 se entrenó en una base de datos de más de 45 TB de texto, casi todo Internet.

La mayoría de los modelos de transducción de secuencias (*Seq to Seq*) neuronales más competitivos usan la estructura de codificador-decodificador, como los RNN. En los transformers, el codificador mapea una secuencia de entrada de representaciones de símbolos (x_1, \dots, x_n) en una secuencia de representaciones continuas $z = (z_1, \dots, z_n)$. Dada z , el decodificador genera una secuencia de salida (y_1, \dots, y_m) de símbolos, generando cada elemento, de uno a uno.

Arquitectura de los Transformer

El modelo Transformer se basa en que tiene una arquitectura basada en la autoatención, conectando capas conectadas aplicadas, tanto en el codificador como en el decodificador, como se puede ver en la **Figura 22**.

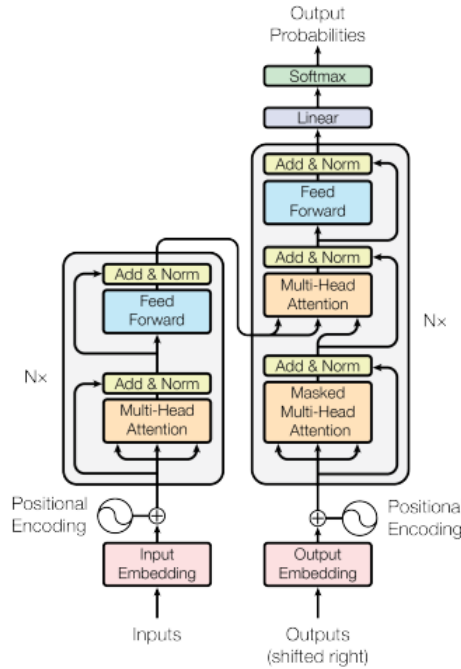


Figura 22: Arquitectura del modelo Transformer [13].

■ Mecanismo de atención

El principio de atención empleado por los transformadores es un mecanismo que permite al modelo centrar su atención en las palabras (**tokens**) más relevantes de una secuencia de texto. Dados vectores de entrada $x_1, \dots, x_n \in \mathbb{R}^d$, se calculan proyecciones lineales:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \quad (34)$$

Donde:

- Q es la matriz de las queries.
- K la matriz de pesos.
- V la matriz de los valores de entrada.
- W^Q, W^K, W^V son matrices de pesos aprendidas.

Calculamos los productos punto de la consulta con todas las claves, dividimos cada uno por $\sqrt{d_k}$, y aplicamos una función softmax para obtener los pesos sobre los valores. En la práctica, calculamos la función de atención sobre un conjunto de consultas de forma matricial y calculamos la matriz de salidas como:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right) V \quad (35)$$

Esto le permite realizar una predicción basada en la secuencia completa, en lugar de considerar cada palabra por separado. El mecanismo de atención utiliza pesos para cada palabra de la secuencia, que indican su importancia relativa para realizar una predicción. Estos pesos, w_1, \dots, w_n se utilizan para ponderar las representaciones de las palabras de la secuencia [13].

Attention Visualizations

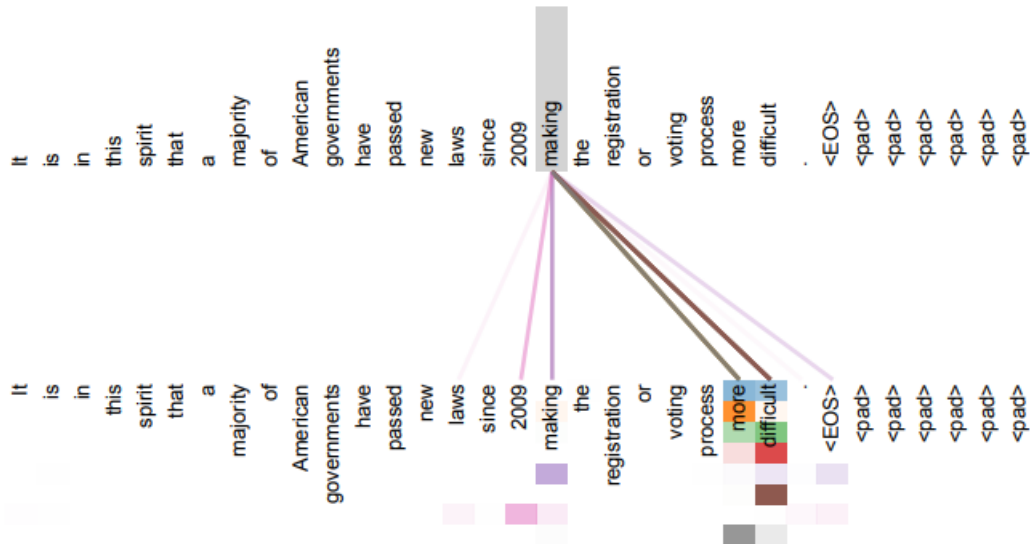


Figura 23: Ejemplo visual del mecanismo atención [13].

■ Multi-Head Attention

La atención multi-cabeza (*multi-head attention*) permite que el modelo atienda simultáneamente a información proveniente de diferentes subespacios de representación y en distintas posiciones. Con una única cabeza de atención, este proceso se ve limitado debido al efecto de promediar la información.

Se define como:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (36)$$

donde cada cabeza de atención se calcula como:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \quad (37)$$

Aquí, las proyecciones son matrices de parámetros:

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, \quad W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}.$$

■ Codificación posicional

La codificación de posición es un componente clave en los Transformers, ya que permite al modelo comprender el orden de las palabras y mejorar sus predicciones. Y esto lo hace mediante una representación numérica (token) de la posición de cada palabra en la secuencia. La codificación posicional $\text{PE} \in \mathbb{R}^{s \times d}$ se define para cada posición $\text{pos} = 0, 1, \dots, s-1$ y dimensión $i = 0, 1, \dots, d-1$ como:

$$\text{PE}_{(\text{pos}, 2i)} = \sin\left(\frac{\text{pos}}{10000^{2i/d}}\right), \quad \text{PE}_{(\text{pos}, 2i+1)} = \cos\left(\frac{\text{pos}}{10000^{2i/d}}\right) \quad (38)$$

La entrada al Transformer se convierte en:

$$X' = X + \text{PE} \quad (39)$$

donde X son los embeddings originales de la secuencia y PE aporta información sobre la posición relativa y absoluta de cada elemento.

■ La autoatención

Los inventores de los transformadores trajeron una nueva innovación, y es la autoatención.

Con esto, podemos darle un valor diferente para cada una de las palabras de la secuencia, donde la atención propuso una ponderación a nivel de toda la secuencia. En un modelo de autoatención, cada palabra de la oración puede sopesar la importancia de otras para comprender el contexto general de la oración, y así poder introducir el concepto que hemos citado anteriormente, el aprendizaje en contexto. En la autoatención, las matrices Q, K, V se derivan de la misma secuencia de entrada X' :

$$Q = X'W_Q = (X + \text{PE})W_Q \quad (40)$$

$$K = X'W_K = (X + \text{PE})W_K \quad (41)$$

$$V = X'W_V = (X + \text{PE})W_V \quad (42)$$

Los puntajes de atención crudos se calculan como:

$$\frac{QK^T}{\sqrt{d_k}} = \frac{((X + \text{PE})W_Q)((X + \text{PE})W_K)^T}{\sqrt{d_k}} \quad (43)$$

Esto implica que los puntajes de atención dependen tanto del contenido de la secuencia (X) como de las posiciones (PE), ya que X' incluye ambos términos. El softmax transforma estos puntajes en pesos normalizados:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (44)$$

¿Por qué autoatención?

La autoatención (*self-attention*) se ha convertido en un mecanismo clave debido a su eficiencia y capacidad para modelar dependencias de largo alcance en secuencias. A diferencia de las RNN, y de las CNN que requieren operaciones secuenciales y tienen una complejidad mayor. Así, la autoatención permite procesar todas las posiciones en paralelo con un número constante de operaciones secuenciales y una menor complejidad.

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|-----------------------------|--------------------------|-----------------------|---------------------|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(\log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

Figura 24: Comparación de la complejidad de las distintas estructuras [13].

Los modelos GPT (Generative Pre-trained Transformer) utilizan únicamente la parte de decodificador de la arquitectura Transformer para generar texto de forma autoregresiva.. Un ejemplo muy conocido de estos modelos es *chatGPT*.

5.1.2. TabPFN

En noviembre de 2022, un nuevo modelo vio la luz de la mano de un grupo de investigadores alemanes, este es tabPFN. Uno de los modelos con la arquitectura de transformadores es tabPFN que significa *tabular Prior data Fitted Network* en castellano datos tabulares ordenados... Este modelo puede predecir mejor que modelos como *Random Forest*, *XGBoost* y la regresión lineal sin que sea este su objetivo principal. Ya que a diferencia de estos modelos, tabPFN sí que toma en cuenta el nombre de las variables para obtener el significado de cada variable/columna, y esto lo logra hacer gracias al aprendizaje contextual.

El hecho de que el modelo entienda todas las variables es muy interesante para encontrar relaciones entre variables y así crear nuevas variables que tengan más sentido estudiar o que sean más significativas.

Por ejemplo: Queremos estudiar el nivel de salud de una población y tenemos el peso, la altura, etc., entonces podemos obtener el IMC.

Aunque aún no se haya llegado a este punto, lo que hace el modelo hasta la fecha es utilizar la misma información que XGBoost. Además, este modelo no necesita el *Deep learning* para que funcione bien.

Dado que nosotros estamos trabajando con datos tabulares y además de un tópico muy popular como el fútbol, a priori parece que TabPFN sería un modelo muy bueno para nuestro estudio.

Aprendizaje contextual (ICL)

El aprendizaje contextual o en inglés *in context learning*, es una técnica que se usa en las redes neuronales que consiste en entrenar previamente sobre millones de tareas sintéticas con distintas distribuciones de entrada, etiquetas y tamaños de muestras. Así, el modelo ha internalizado un comportamiento bayesiano aproximado que se puede aplicar a nuevos conjuntos de datos sin reentrenamiento.

Clasificación bayesiana

Sea $\mathcal{D} = (X_{\text{train}}, y_{\text{train}}, X_{\text{test}})$, con $X_{\text{train}} \in \mathbb{R}^{m \times d}$, $y_{\text{train}} \in \{1, \dots, C\}^m$, y $X_{\text{test}} \in \mathbb{R}^{n \times d}$, y produce $\hat{y}_{\text{test}} \in \mathbb{R}^{n \times C}$ en un solo paso hacia adelante. En el enfoque clásico de clasificación bayesiana, el objetivo es estimar la probabilidad posterior de una clase y dada una observación x y un conjunto de datos D [15]:

$$P(y | x, D) = \int P(y | x, \theta) P(\theta | D) d\theta \quad (45)$$

donde θ representa los parámetros del modelo (ICL).

Predicción directa del posterior con TabPFN

TabPFN (Tabular Prior-Formed Network) es un modelo basado en Transformers que aprende directamente una función de predicción de la distribución posterior de clases sin necesidad de entrenamiento en el conjunto de datos específico:

$$f : (D_{\text{train}}, x_{\text{test}}) \longrightarrow P(y_{\text{test}}) \quad (46)$$

Es decir, dado un pequeño conjunto de entrenamiento D_{train} y una nueva observación x_{test} , el modelo predice directamente la distribución $P(y_{\text{test}})$.

Mecanismo de atención del Transformer

TabPFN se basa en la arquitectura Transformer, cuyo núcleo es el mecanismo de autoatención. Este mecanismo permite que el modelo relacione cada punto con los demás, ponderando la relevancia de cada ejemplo en la predicción final. Debido a que TabPFN ha sido entrenado utilizando ICL, de este modo, se puede aplicar a nuevos conjuntos de datos sin reentrenamiento. TabPFN usa la autoatención entre muestras de entrenamiento y atención cruzada de las muestras de prueba a las de entrenamiento [14][15].

$$\text{CrossAttention}(Q_{\text{test}}, K_{\text{train}}, V_{\text{train}}) = \text{softmax} \left(\frac{Q_{\text{test}} K_{\text{train}}^T}{\sqrt{d_k}} \right) V_{\text{train}} \quad (47)$$

Donde:

- Las consultas, denotadas $Q_{\text{test}} = X_{\text{test}} W_Q \in \mathbb{R}^{n \times d_k}$, se generan a partir de las muestras de prueba X_{test} .
- Las claves $K_{\text{train}} = X_{\text{train}} W_K \in \mathbb{R}^{m \times d_k}$ y los valores $V_{\text{train}} = X_{\text{train}} W_V \in \mathbb{R}^{m \times d_v}$ se obtienen de las muestras de entrenamiento X_{train} .
- El producto matricial $Q_{\text{test}} K_{\text{train}}^T \in \mathbb{R}^{n \times m}$ calcula las similitudes entre cada muestra de prueba y cada muestra de entrenamiento.
- La normalización por $\sqrt{d_k}$ estabiliza la varianza.

A continuación vamos a seguir comparando los modelos. Para ello vamos a usar el modelo que acabamos de ver cómo se construye y comparar con los modelos más clásicos, como los vistos en la **sección 4**.

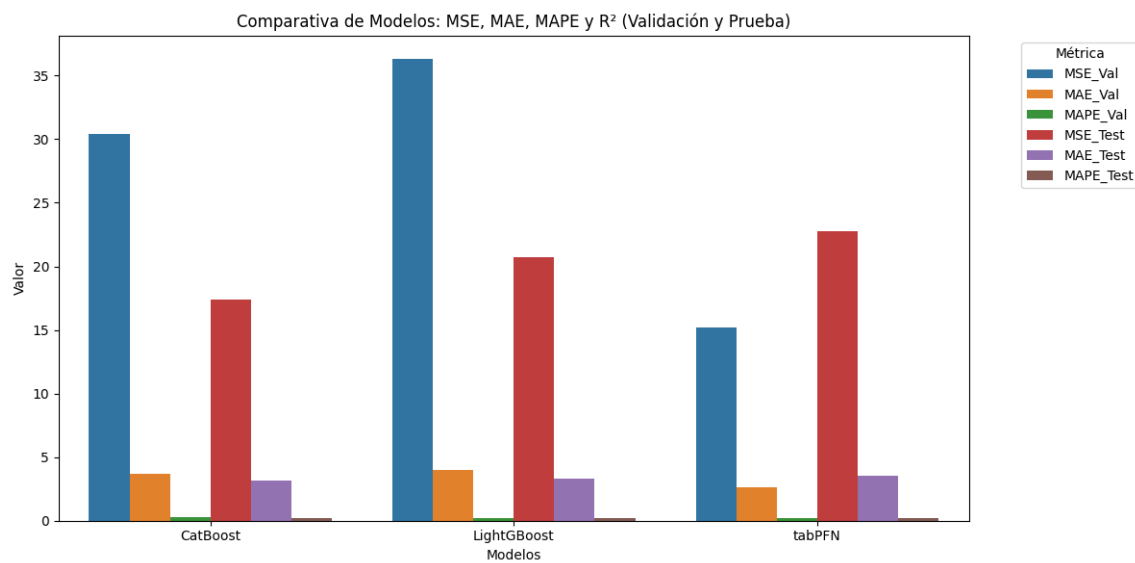


Figura 25: Comparación de modelos

Podemos ver a simple vista que los modelos CATBoost y LightGBM, tienden a entrenar peor que el modelo TabPFN. Además, podemos observar que la predicción test es mejor con TabPFN. En la **Figura 26** podemos ver de manera más objetiva, utilizando números, qué modelo predice mejor.

| TEST | MAPE | MSE |
|----------|------|-----|
| CatBoost | 21 % | 17 |
| LightGBM | 20 % | 17 |
| tabPFN | 18 % | 23 |

Figura 26: TabPFN es el modelo que mejor predice, como nos podíamos imaginar

Ventajas de tabPFN

Desde un punto de vista matemático y computacional, TabPFN permite:

- Capturar relaciones no lineales e interacciones complejas entre variables a través de ICL. Y dado que está es un modelo preentrenado con una base de datos muy grande, nos da resultados muy buenos.
- Trabajar directamente con variables con valores nulos sin necesidad de preprocesamiento.

Desventajas de tabPFN

Algunas limitaciones desde un punto de vista matemático y computacional, de TabPFN:

- TabPFN está diseñado para resolver tareas tabulares completas de golpe y recibe todo el *dataset* como secuencia de entrada. Y dado que TabPFN no deja de ser un modelo transformer, tendrá una complejidad de $O(n^2 \cdot d)$ (autoatención) + $O(n \cdot d)$ (del *dataset*).
- Por tanto esto hace que tabPFN considere grandes conjuntos a los conjuntos de más de 10,000 observaciones y no pueda trabajar con conjuntos de datos tan grandes, por su alta complejidad.

6. Discusión

En esta sección vamos a revisar los resultados obtenidos y valorar las diferentes opciones para proponer la mejor solución al equipo, utilizando los modelos y datos que hemos ido presentando a lo largo del trabajo.

Recopilando los resultados obtenidos de las secciones 4 y 5, podemos seleccionar una *shortlist* con los mejores defensas, dentro de nuestro presupuesto actualizado de 63 M€.

| Jugador | Rendimiento Defensivo | Valor de mercado (en millones) |
|------------------|-----------------------|--------------------------------|
| Antonee Robinson | 59.42 | 30 |
| Daley Blind | 55.46 | 3 |
| Emerson Palmieri | 54.57 | 12 |
| Vladimír Coufal | 51 | 8 |
| Sergi Cardona | 47 | 5 |
| Mario Hermoso | 46.11 | 10 |
| Óscar Mingueza | 45.9 | 5 |

Tabla 3: *Shortlist de los candidatos para ser fichados. De esta lista se han excluido jugadores con un salario fuera de presupuesto, como O. Zinchenko y Virgil van Dijk.*

Análogamente al estudio de los jugadores en el área defensiva, vamos a analizar el **Rendimiento Ofensivo**. Y llegamos a las siguientes tablas donde vemos una comparación de cada modelo empleado. El resto de pasos y resultados, se puede encontrar en el repositorio de este trabajo.

| TEST | MAPE | MSE |
|---------------|------|-----|
| Reg. lineal | 45 % | 133 |
| Random Forest | 59 % | 240 |
| XGBoost | 52 % | 230 |

Figura 27: *Errores del test*

| TEST | MAPE | MSE |
|----------|------|-----|
| CatBoost | 41 % | 129 |
| LightGBM | 53 % | 209 |
| tabPFN | 39 % | 120 |

Figura 28: *Errores del test*

Y como podíamos esperar, el modelo que con un menor error absoluto parcial medio (MAPE) es TabPFN. Podemos observar que el error cuadrático medio es significativamente más elevado que para el rendimiento defensivo, y esto se puede deber a una mayor presencia de valores atípicos.

Utilizando entonces el modelo TabPFN y prediciendo el rendimiento ofensivo de los jugadores y clasificándolos de mayor a menor, podemos seleccionar a los mejores jugadores dentro de nuestro presupuesto.

| Jugador | Rendimiento ofensivo | Valor de mercado (en millones) |
|---------------|----------------------|--------------------------------|
| Son Heung-min | 71 | 45 |
| Savinho | 55.9 | 35 |
| Artem Dovbyk | 53 | 35 |
| Iago Aspas | 50.9 | 2 |
| Willian | 47 | 2 |
| Alex Iwobi | 43 | 25 |
| Lucas Ocampos | 40 | 12 |

Tabla 4: Shortlist de los candidatos para ser fichados. De esta lista se han excluido jugadores con un salario fuera de presupuesto. Y a parte tenemos candidatos fuera de la shortlist que pueden llegar a entrar.

Ahora nos queda predecir la rentabilidad de cada fichaje. Y para ello nuestra variable target será **Valor de Mercado**, y como se ha hecho hasta ahora vamos a empezar con la fase de elección de las variables explicativas, fase de elección de modelo, y finalmente la predicción.

| TEST | MAPE | MSE |
|----------|-------|-----|
| CatBoost | 129 % | 105 |
| LightGBM | 95 % | 79 |
| tabPFN | 60 % | 55 |

Como podíamos intuir, TabPFN sigue siendo el mejor modelo. El segundo mejor modelo es LightGBM, el cual vamos a utilizar para predecir también a fin de comparar

de una forma más objetiva los resultados obtenidos de TabPFN. Predecimos el valor de mercado de los jugadores de nuestra $shortlist = Tabal3 \cup Tabla4$. Y nos quedamos con:

| Jugador | Val. Mercado actual (M€) | Val. Mercado predicho (M€) |
|-----------------|--------------------------|----------------------------|
| Savinho | 35 | 52 |
| Sergi Cardona | 5 | 7 |
| Mario Hermoso | 10 | 29 |
| Willian | 2 | 6.9 |
| Daley Blind | 3 | 7 |
| Vladimír Coufal | 8 | 10 |

Tabla 5: Fichajes propuestos por el equipo de análisis de datos

Resultados

- Savinho fue fichado por el Manchester City y su valor de mercado actual es 55 M€, una diferencia de 3 M€ respecto de nuestra predicción.
- Un de los peores jugadores de nuestro equipo y uno de los peores porteros de nuestro *dataset*, F.Jørgensen, fue fichado por 24M€. Y su rendimiento sigue sin mejorar, Así que fue un acierto su venta.
- El Villarreal terminó comprando a Sergi Cardona, y su rendimiento ha sido muy bueno. Y su valor de mercado es 8 M€, habíamos predicho 7 M€. Y su rendimiento ha sido muy bueno.



| TEMPORADA | EQUIPO | COMPETICIÓN | Ø | 🏠 | ⚽ | A |
|-----------|--|--|-----|----|---|---|
| 2024/2025 |  Villarreal |  LaLiga EA Sports | 7.0 | 35 | 1 | 6 |

Figura 29: Rendimiento de Sergi Cardona en la temporada 2024/25

- Algunos de los jugadores que hemos predicho que aumentaría su valor de mercado ha bajado, esto se debe sobre todo al factor edad o posiblemente como en el caso de *Daley Blind* a una temporada fuera de lo normal para el jugador. Estas son algunas de las limitaciones que discutiremos más adelante sobre utilizar el análisis de datos en deportes, y sobre todo en el fútbol.

- El Villarreal terminó fichando el mismo número de jugadores que nosotros y jugadores del mismo perfil al propuesto por el equipo de análisis. Y lo más importante, el Villarreal volvió a clasificar por las competiciones europeas.

| # | ▲ | EQUIPO | PJ | G | E | P | G | DG | PTS | FORMA |
|----|---|--|----|----|----|----|--------|----|-----|-----------|
| 1. | |  Barcelona | 38 | 28 | 4 | 6 | 102:39 | 63 | 88 | G P G G G |
| 2. | |  Real Madrid | 38 | 26 | 6 | 6 | 78:38 | 40 | 84 | G G G P G |
| 3. | |  Atlético de Madrid | 38 | 22 | 10 | 6 | 68:30 | 38 | 76 | G G P G E |
| 4. | |  Athletic Club | 38 | 19 | 13 | 6 | 54:29 | 25 | 70 | P G G G E |
| 5. | |  Villarreal | 38 | 20 | 10 | 8 | 71:51 | 20 | 70 | G G G G G |
| 6. | |  Real Betis | 38 | 16 | 12 | 10 | 57:50 | 7 | 60 | E P E E G |

Figura 30: Clasificación de LaLiga de la temporada 2024/25

Y así, se ha logrado cumplir con los objetivos del trabajo.

Limitaciones del trabajo

- En gran parte de este trabajo se ha usado un ordenador no muy potente, y esto puede hacer que algunos modelos no funcionen de manera óptima o que el tiempo de ejecución sea excesivo.
- Hemos observado que hasta usando las técnicas más avanzadas de predicción, como redes neuronales, tabPFN, no podemos predecir con precisión los resultados deseados.
- En nuestro conjunto de datos no tenemos componente temporal. Lo cual me impide ver otra información interesante, como el rendimiento de ciertos jugadores o del equipo en distintas épocas de la temporada.

Limitaciones de la aplicación al fútbol:

- En comparación con otros deportes, el fútbol es mucho más continuo, sin muchos descansos y donde la influencia de una jugada realizada hace 20 jugadas puede ser muy significativa, y de manera acumulada puede hacer que la capacidad predictiva sea muy baja. En comparación, deportes como el baseball con el caso del “*Moneyball*” es más fácil de analizar ya que no tiene tantas posibles jugadas.
- Algunas de las limitaciones de la aplicación de técnicas de análisis de datos para predecir el rendimiento de un jugador puede no ser correcta por que hay factores que no se pueden medir mediante números, por ejemplo la motivación del jugador durante esa temporada, el amor del jugador por un club.

Responder a las preguntas planteadas al inicio

A lo largo del TFG, hemos intentado responder a las preguntas planteadas al inicio del trabajo, y hemos llegado a las siguientes conclusiones:

Q: ¿Cómo podemos implementar el análisis de datos en los deportes?

R: El análisis de datos se implementa en el deporte mediante la recolección, procesamiento y modelización de datos para optimizar el rendimiento deportivo y la toma de decisiones económicas. En deportes con métricas discretas, como el béisbol, su implementación ha sido más directa (caso “*Moneyball*”), mientras que en deportes continuos como el fútbol, requiere el uso de modelos más avanzados (transformers, TabPFN) para capturar la complejidad de las interacciones entre variables y aunque no lo hemos tratado en este trabajo, pero se deja para futuros trabajos, su evolución temporal.

Q: ¿Cómo se construyen los modelos transformer desde el punto de vista matemático?

R: En la sección 5, Se explica cómo los modelos *Transformer* revolucionan el análisis de secuencias. Estos modelos emplean *tokenización* para dividir los datos en partes manejables y un mecanismo de *atención* para enfocarse en lo más relevante. Esto se logra calculando pesos mediante productos matriciales y la función *softmax*, que distribuye la importancia entre los datos. Además, del uso de la *atención multi-cabeza* que permite analizar varias relaciones simultáneamente, así el modelo refina sus predicciones, capturando patrones complejos. Lo más destacado es que, al procesar todo en paralelo, resulta mucho más rápido que otros modelos más clásicos.

Q: ¿Por qué autoatención?

R: La respuesta a esta pregunta esta explícitamente en la sección 5.1.1.

Q: ¿Qué ventajas ofrece TabPFN frente a otros modelos de regresión tradicionales en datos tabulares?

R: TabPFN presenta ventajas significativas, como su capacidad de inferir directamente distribuciones posteriores gracias al aprendizaje contextual (ICL), su robustez ante datos nulos sin necesidad de preprocesamiento, y su habilidad para capturar relaciones no lineales y complejas entre variables, superando en rendimiento a modelos tradicionales como los ensembles que hemos visto en la sección 4.

Q: ¿Podemos usar el análisis de datos para tomar decisiones económicas, como fichar jugadores con proyección? ¿En qué variables o estadísticas se basan los expertos para definir el valor de mercado de los jugadores?

R: Sí, el análisis de datos permite evaluar de forma objetiva el rendimiento y potencial económico de los jugadores. Entre los aspectos en los que se suelen basar los expertos para definir el valor de mercado de los jugadores, tenemos variables como la edad, los goles, las asistencias, los minutos jugados, la contribución defensiva, la posición y las características técnicas. Esta información, combinada con modelos predictivos, ayuda a identificar jugadores infravalorados con alto potencial de revalorización. Sin embargo, el análisis de datos no es una ciencia o creencia que deba seguirse ciegamente, ya que, como hemos visto a lo largo de este trabajo y en las asignaturas de análisis de datos, los modelos suelen equivocarse y los analistas pueden malinterpretar los resultados.

7. Conclusiones y futuros trabajos

En este TFG se han utilizado herramientas matemáticas aprendidas a lo largo de la carrera, y han sido aplicadas para resolver problemas reales, donde se ha podido llegar a dar respuesta utilizando exclusivamente las matemáticas en sectores como el fútbol. Para ello, se han introducido, explicado y finalmente utilizado modelos de análisis de datos, desde los más básicos hasta los más complejos, como los modelos de aprendizaje profundo TabPFN, los cuales han supuesto un reto para comprender en profundidad y para encontrar material con una explicación matemática clara.

Como limitaciones de este trabajo, tal y como se mencionó en el apartado anterior, destaca la necesidad de un ordenador con una GPU potente para poder trabajar con modelos de *deep learning*, como las redes neuronales utilizadas, así como la dificultad de predecir deportes con tantos sucesos aleatorios como el fútbol. A este trabajo le ha faltado aplicar modelos de clasificación, y esto se debe a que habría alargado considerablemente su desarrollo.

Futuros trabajos

Como hemos visto, el margen de progreso de la IA en general y los modelos basados en la estructura de redes neuronales en específico, es bastante amplio. Para ello se propone como trabajos futuros:

- Optimizar la estructura de TabPFN, para evitar las desventajas que hemos citado en la discusión.
- Usar tabPFN combinado con otros modelos para poder clasificar con exactitud a qué idioma pertenece un texto. La dificultad de esto reside en entrenar un modelo para entender los sentimientos, los errores de ortografía, el lenguaje coloquial de la calle.
- Se deja para un futuro trabajo aumentar el conjunto de datos, incorporando jugadores de otras ligas interesantes y así obtener modelos más robustos con predicciones más acertadas.
- Otra forma de abordar el análisis de datos en el fútbol es teniendo en cuenta los patrones temporales y espaciales. Para ello, podríamos considerar un conjunto de datos con la característica datos temporales para aplicar la teoría de series temporales. Se recomienda para ello usar Modelos Moirai para series temporales.

Referencias

- [1] Statista, *El fútbol en España - Datos estadísticos* . <https://es.statista.com/temas/2864/el-futbol-en-espana/#topicOverview> (Consultado el 25 de febrero de 2025).
- [2] TyC Sports, *La emotiva historia del día que Didier Drogba frenó la guerra de Costa de Marfil*. <https://www.tycsports.com/interes-general/didier-drogba-cumpleanos-guerra-civil-costa-de-marfil-id418729.html> (Consultado el 25 de febrero de 2025).
- [3] Fbref.com. *La Liga*.
<https://fbref.com/en/comps/12/2023-2024/2023-2024-La-Liga-Stats>
(Consultado el 20 de febrero de 2025)
- [4] Stef van Buuren (2018). *Flexible Imputation of Missing Data* (2nd edición), Chapman Hall/CRC.
<https://stefvanbuuren.name/fimd/about-the-author.html>
- [5] Equipo inaCatalog (2023). *Mejora el rendimiento con los KPIs personalizables en una empresa* .
<https://www.inacatalog.com/blog/kpis-personalizables-empresa>
- [6] Efron, B., and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall/CRC.
- [7] Leo Breiman, Jerome Friedman, R.A. Olshen, Charles J. Stone. (1984). *Classification and Regression Trees*. 1ª edición. Chapman and Hall/CRC.
- [8] Ilyes Talbi (2023). *Introduction aux réseaux de neurones Transformers*.
<https://larevueia.fr/introduction-aux-reseaux-de-neurones-transformers/>
(Consultado el 2 de mayo de 2025)
- [9] Ian G., Yoshua B. and Aaron Courville (2017). *Deep Learning*. MIT Press.
- [10] Chen, W., Finley, T., Liu, T.Y., Ke, G., Ma, W., Meng, Q., Wang, T. and Ye, Q. (2017). *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf (Consultado el 2 de julio de 2025)

- [11] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V. and Gulin, A. (2018). *CatBoost: Unbiased Boosting with Categorical Features*. Advances in Neural Information Processing Systems (NeurIPS), pp. 6638–6648.
- [12] Raúl E. López Briega (2017). *Deep Learning*, Introducción al Deep Learning.
<https://iaarbook.github.io/deeplearning/> (Consultado el 1 de junio de 2025)
- [13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). *Attention Is All You Need*.
<https://arxiv.org/abs/1706.03762>
- [14] Hollmann, N., Müller, S., Eggensperger, K., Hutter, F. (2022). *TabPFN: A transformer that solves small tabular classification problems in a second*. Universidad de Fribourg, Alemania.
- [15] Hollmann, N., Müller, S., Eggensperger, K., Sebastian P., Josif G., Hutter, F. (2025). *TRANSFORMERS CAN DO BAYESIAN INFERENCE*.
<https://openreview.net/forum?id=KSugKcbNf9> (Consultado el 29 de Abril de 2025)
- [16] O'Reilly Media, Inc (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2^a Edición.
- [17] Daniel Memmert, Dominik Raabe. (2023). *Data Analytics in Football (2nd Edition)*. Routledge.
- [18] Adam Fleischhacker. (2025). *A Business Analyst's Introduction to Business Analytics (2nd Edition)*. Amazon.

A. Detalles del desarrollo del trabajo

Este trabajo ha sido desarrollado usando el lenguaje de programación Python y en una máquina virtual ofrecida por *Google Colab*. Cuya capacidad del hardware viene dada por:

Núcleos físicos de CPU: 1

Núcleos totales de CPU: 2

Frecuencia de CPU: 2000.226 MHz

Memoria total: 12.67 GB

Memoria disponible: 11.71 GB

La mayoría de los paquetes usados han sido **pandas**, **seaborn**, **sklearn** (**scikit-learn**) y **tensorflow**. Todo el material utilizado para elaborar este trabajo se puede encontrar en el repositorio de *GitHub*, <https://github.com/BenlagraaMohamed/TFG>.

| Tarea | Tiempo (horas) |
|--|----------------|
| Recopilación de materiales | 40 |
| Estudio de bibliografía | 45 |
| Elaboración de resultados gráficos/numéricos | 5 |
| Redacción de la memoria | 65 |
| Total | 150 |

Tabla 6: Tiempo aproximado de dedicación al trabajo

| Asignatura | Páginas | Descripción |
|------------------------|---------|---|
| Análisis de Datos I | 8-9 | Definición de datos y variables se saca del tema 1. |
| Inferencia Estadística | 13, 17 | Definición TCL y de cota superior de un IC. |
| Análisis de Datos I | 20-21 | Matrices de covarianza y de correlaciones. |
| Optimización II | 28-29 | Definición del descenso del gradiente. |
| Análisis de Datos II | 27-32 | Definiciones generales de los modelos ensemble y más. |

Tabla 7: Asignaturas relacionadas con el trabajo

B. Apéndice

Variables del conjunto de datos

Explicación del resto de las variables del conjunto de datos que no salen en la **Tabla 1**.

- **Valor Mercado (millones de e):** Valor de mercado del jugador a la hora de la creación del *dataset*, a principios de febrero de 2025. De la página líder en valor de mercado de los jugadores: Transfermarkt, <https://www.transfermarkt.es/>
- **PJ:** Partidos jugados.
- **PG:** Partidos ganados.
- **PE:** Partidos empatados.
- **PP:** Partidos perdidos.
- **GC:** goles recibidos por el equipo.
- **Puntosjugador:** Puntos por jugador por cada partido.
- **Pts:** Puntos del equipo del jugador.
- **Titular:** N^o de partidos donde ha sido titular.
- **Suplente:** N^o de partidos donde ha sido suplente.
- **Mín:** Minutos jugados.
- **G-TP:** Goles sin contar penaltis.
- **G-TP:** N^o de penalties tirados.
- **Penal.Anotado:** Goles de penalti.
- **TA:** N^o de tarjetas amarillas recibidas.
- **TR:** N^o de tarjetas rojas recibidas.
- **Avances:** Ocasiones peligrosas con avance individual.
- **Pases.Peligrosos:** Pases importantes con probabilidad de gol alto.
- **Pases.Recibidos:** Pases peligrosos recibidos por el jugador.

- **Regates:** N^o de regates efectuados.
- **RegatesExt:** N^o de regates efectuados con éxito.
- **DesafiosDef:** Desafios defensivos.
- **DesafExt %:** Porcentaje de desafios defensivos exitosos.
- ...

Material adicional (Para modelos de clasificación):

Métricas para clasificación.

Matriz de confusión

La matriz de confusión nos da información sobre qué tan bien nuestro modelo predice. Para ello, se utiliza un mecanismo que se llama **validación cruzada**. El esquema de una matriz de confusión es el siguiente:

| | | Clase Real | |
|----------------|----------|------------|----------|
| | | Positivo | Negativo |
| Clase Predicha | Positivo | TP | FP |
| | Negativo | FN | TN |

Tabla 8: Matriz de confusión.

Donde:

TP: Verdaderos Positivos

FP: Falsos Positivos

FN: Falsos Negativos

TN: Verdaderos Negativos

Métricas de evaluación para clasificación

- **La precisión** representa el porcentaje de los positivos que han sido bien clasificados:

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

- Otro criterio es **la sensibilidad**, la cual representa el porcentaje de verdaderos positivos detectados, dada por:

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

- La **accuracy** es el ratio de aciertos, es decir de observaciones que han sido clasificadas correctamente y se calcula con la siguiente fórmula:

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

- El **F1-score** es útil cuando se necesita buscar un balance entre la precisión y la sensibilidad. Generalmente más útil que la precisión cuando la distribución de las clases es distinta. Y viene dado por:

$$F1 = 2 * \frac{\text{Precisin} * \text{Recall}}{\text{Precisin} + \text{Recall}}$$

- **Curva ROC y AUC**

La **curva ROC** (*Receiver Operating Characteristic*), sirve para medir las tasa de aumento de los FP respecto de los TP. En la figura 6 tenemos un ejemplo de una curva ROC:

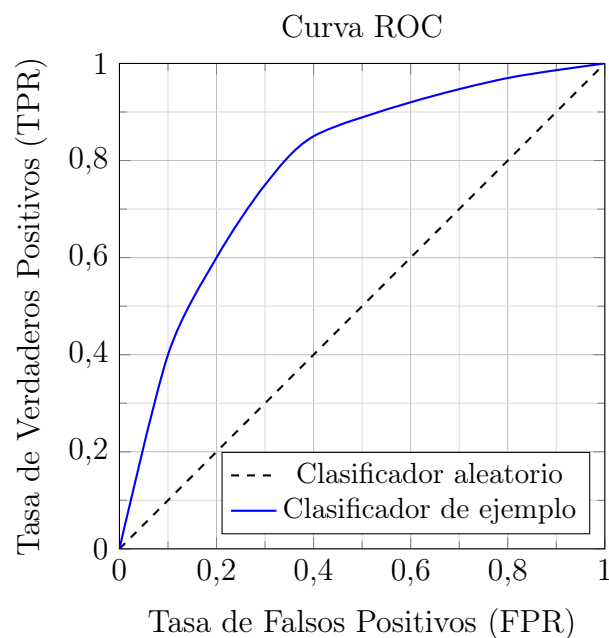


Figura 31: La línea discontinua representa un clasificador aleatorio ($AUC = 0.5$).

AUC

Llamamos AUC a el área que hay debajo de la curva ROC. El AUC es el criterio para decidir si un modelo es mejor que otro, comparando quién tiene un AUC mayor. Un AUC perfecto es $AUC = 1$.

Y para decidir qué modelo es mejor, podemos usar la curva ROC, usando el AUC. Y en algunos casos es preferible usar la precisión en la curva ROC antes que la tasa de falsos positivos. Teniendo ya la curva ROC, podemos decidir qué threshold es preferible y comparar entre modelos también.

Modelos GLM para clasificación

Regresión logística

A diferencia del modelo anterior, la regresión logística mapea la imagen de $(-\infty, \infty)$ a $(0, 1)$ usando el logaritmo neperiano, además de ser un modelo de clasificación. La fórmula del modelo tiene la siguiente fórmula:

$$Y = E(Y \mid X = x) = e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p} \quad (48)$$

Este método construye un modelo que intenta separar mediante fronteras lineales/*thresholds* los elementos de las diferentes clases. De modo que el método clasifica a la variable x como 0 o 1. Si $P(x) < 0,5 \Rightarrow x \in class0$ o $P(x) > 0,5 \Rightarrow x \in class1$.

La Regresión Logística tiene como objetivo la clasificación, y debido a que no vamos a usar modelos de clasificación, este modelo no lo vamos a utilizar más que para introducir la función Softmax que siguen una estructura parecida.

Algunas de estas técnicas de *cross validation* son:

LOCO (Leave-One-Out Cross-Validation). Esta técnica utiliza todo el conjunto de datos menos una observación para entrenar el modelo, y la observación excluida se usa para probarlo. Este proceso se repite para cada observación del conjunto de datos, de modo que cada punto de datos actúa como conjunto de prueba exactamente una vez. En una muestra de tamaño n donde queremos aplicar esta técnica de validación cruzada, debemos seguir los siguientes pasos:

1. Utilizar como conjunto *train* a $n-1$ observaciones.
2. Usar a la observación restante como conjunto *test*.

3. Evalua el modelo usando MSE, R^2 , accuracy, precisión, etc.

- **Ventajas de este método:** Uso máximo de los datos. Proporciona una estimación más robusta del rendimiento del modelo.
- **Cuándo usar este método:** Obtener una estimación más fiable del rendimiento del modelo.

K-fold validación cruzada.

Esta técnica tiene la particularidad de en lugar de entrenar el modelo con todos los datos y evaluar con una sola partición como la validación cruzada, el modelo:

1. Divide el conjunto en K partes (folds).
 2. El modelo se entrena con K-1 conjuntos aleatorios de esas K partes.
 3. Evalua el modelo con la parte restante, usando MSE, R^2 , accuracy, precisión, etc.
- **Ventajas de este método:** Obtener una estimación más fiable del rendimiento del modelo. Menor coste computacional que LOCO. Los valores atípicos no causan tanto problema como con LOCO.
 - **Cuándo usar este método:** Ideal para conjuntos de datos grandes. Solo para *datasets* out of time (no temporales).