El valor de Shapley y su aplicación a modelos de regresión.

Jose Ramón Buergo Lagunas
Trabajo de fin de grado de Matemáticas
Universidad de Zaragoza

Director del trabajo: José Tomás Alcalá Nalvaiz
12 de junio de 2024

Summary

This paper aims to dive into the connection between game theory and the field of artificial intelligence. We will focus on the application of the Shapley value in the field of machine learning, exploring both its theoretical foundations and practical applications. Throughout the first three chapters, we establish conceptual bases, detail specific methods and approaches, and finally present case studies that illustrate the relevance of the Shapley value in artificial intelligence. Additionally, an analysis of data from the Spanish Football League is included as a practical example to reinforce and demonstrate the utility of the theoretical concepts developed.

In the first chapter, the fundamental concepts of game theory and artificial intelligence are presented, highlighting the importance of explainability in machine learning models. Game theory, initiated in 1944 by Von Neumann and Morgenstern and based on earlier research by Zermelo and Borel, analyzes strategic decisions in situations with established rules and rationality. Key aspects of game theory are addressed, including definitions and preliminary solutions in N-player games, with an emphasis on the Shapley value for equitably distributing gains or costs. Regarding artificial intelligence (AI) and machine learning (ML), these fields emerged in response to the growth of data generation. AI aims to automate complex tasks and improve efficiency in various applications, while ML focuses on developing algorithms that enable computers to learn from data sets, facilitating prediction or classification. Probability theory and statistics play a crucial role in ML, providing models such as linear and logistic regression along with tools to evaluate the quality of results. Different approaches to machine learning are explored, emphasizing the importance of explainability in AI to understand and trust the results obtained.

The second chapter explores the application of the Shapley value for interpreting machine learning models, with a special focus on regression models. It describes how the Shapley value can provide both a global view of the model and an individual evaluation of variables. In terms of regression models, their fundamentals are explored, highlighting key concepts such as predictor variables, response variables, and regression coefficients. The types of regression models, both linear and nonlinear, are explained in detail, with specific examples such as multiple linear regression and logistic regression. Additionally, the chapter addresses methods for calculating the Shapley value, emphasizing techniques like SHAP (Shapley Additive Explanations), which provides far more efficient interpretation of model predictions. Computationally advanced approaches such as Kernel SHAP, which uses kernel estimation techniques to approximate the Shapley value; Tree SHAP, specifically designed for tree-based models; and FastSHAP, a model trained to compute Shapley values, are also discussed. Furthermore, other model-agnostic explanations, like LIME (local interpretable model-agnostic explanations), for the interpretation of machine learning models are explored.

The third chapter presents a case study in which the Shapley algorithm is applied to the analysis of the Spanish Football League. The methodology for data collection and analysis is detailed, and an exhaustive analysis of the obtained results is conducted. This case study demonstrates how the Shapley value can provide a deeper understanding of team performance.

Two appendices are included. The first presents pending theorem demonstrations. The model analysis, dataset acquisition, SHAP implementation, and pertinent code are covered in the second one.

In conclusion, this work demonstrates the importance of the Shapley value in the interpretation of machine learning models and its application across various domains. Through theoretical examples and practical applications, it emphasizes how these tools can enhance transparency and effectiveness in decision-making based on artificial intelligence.

Índice general

Capítulo 1

Introducción de Conceptos

1.1. Introducción a la Teoría de Juegos

La palabra *juego* esta asociada a una actividad lúdica en la que participan un grupo de personas que deben cumplir con ciertas normas. Con el objetivo de ganar, estas tratarán de desarrollar una estrategia que favorezca sus opciones.

Con el obtetivo de establecer una base para fundamentar las estrategias, surge la rama de las matemáticas conocida como la teoría de juegos. Su inicio se podría considerar en 1944 con la publicacion de *Game Theory and Economic Behaviour* de Von Neumann y Morgenstern, apoyandose en trabajos anteriores de Zermelo y Borel.

La teoría de juegos abarca una amplia variedad de temas fundamentales en el estudio de la toma de decisiones estratégicas, basándose en las reglas dadas, la información que poseen los jugadores y su racionalidad. En libros como [10] y [19], podemos profundizar en algunos de sus conceptos principales, como los juegos simples de dos jugadores, los juegos de suma cero y no suma cero, el equilibrio de Nash, la cooperación y la competencia entre jugadores, la formación de coaliciones, los juegos secuenciales, el manejo de la incertidumbre y la cantidad de información. Gracias a estos temas, se puede comprender cómo cada individuo y los grupos interactúan en las situaciones estratégicas.

Esto se traduce en aplicaciones en campos como la economía, para modelar la competencia entre empresas o la toma de decisiones en mercados complejos; la biología, para el estudio de la evolución de especies y el comportamiento de un ecosistema; y en política, para analizar las estrategias de negociación y el voto. Además, también encuentra aplicaciones en la toma de decisiones en informática y otros muchos campos en los cuales las interacciones estratégicas estén presentes.

1.1.1. Definiciones y Soluciones Preliminares en Juegos de N Jugadores

A continuacion, vamos a profundizar un poco más en algunos temas que nos ayudarán en el desarrollo y comprensión de las cuestiones que trataremos.

Definición 1. Un juego de **N-jugadores** consta de una serie de eventos que puede tener un número de resultados finito. Los acontecimientos del juego dependen de la libre decisión del conjunto de jugadores $J = \{1, 2, \ldots, n\}$. En algunos casos el resultado puede depender del azar. Para cada evento, se sabe que el jugador i determina su resultado conforme a sus estrategias disponibles y cuál es su estado de información respecto al resto de jugadores. Finalmente, tras el resultado obtenido apartir de los eventos se asigna un pago a cada uno de los jugadores.

Definición 2. Un juego en forma coalicional [19] o en forma de función característica con utilidades transferibles es un modelo matemático en el que los jugadores pueden cooperar entre sí para maximizar sus posibles ganancias individuales. Este consiste en::

- Un conjunto finito de jugadores $J = \{1, 2, \ldots, n\}$.
- Una función característica, que asocia a cada subconjunto $S \in \mathscr{P}(J)$ (o coalición) un número real $v(S)$ (valor de coalición), siendo $v(\emptyset) = 0$.

Por tanto, $G = (J, v)$ es un juego en forma coalicional o en forma de función característica con utilidades transferibles si J y v están especificados.

Definición 3. Si G es un juego coalicional para los jugadores 1, 2, ..., n entonces un vector de pagos $(x_1, \ldots, x_n)$ con x_i correspondiente al jugador i es una **imputación** [10] si cumple las siguientes propiedades:

1. Racionalidad individual: $x_i \geq v(i)$ para todo $1 \leq i \leq n$.
2. Racionalidad grupal: $x_1 + x_2 + \ldots + x_n = v(J)$

Denominaremos $I(J, v)$ al conjunto de imputaciones.

El vector de la pagos expresa la recompensa que recibe cada jugador, es decir, la componente x_i representa el pago que recibe el jugador i.

Definición 4. El **core** [19] de un juego $G = (J, v)$ es el siguiente conjunto de vectores de pagos:

$$C(J, v) = \{x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n : x(J) = v(J),\ x(S) \geq v(S),\ \forall S \in \mathscr{P}(J)\}$$

De la definición de core es importante el hecho de que es un subconjunto del conjunto de imputaciones y que ningún jugador puede conseguir más solo que lo que conseguiría en una coalición.

Definición 5. Para cada $x \in I(J, v)$, se define el vector de excesos [19] como el siguiente vector $\theta(x)$, con 2^n componentes:

$$\theta(x) = (v(S) - x(S))_{S \in \mathscr{P}(J)} = (\theta_1(x), \theta_2(x), \ldots, \theta_{2^n}(x))$$

en donde $\theta_k(x) \geq \theta_{k+1}(x), \forall k = 1, 2, \ldots, 2^n - 1$.

Definición 6. El **nucleolus** [19] de un juego (J, v) es un conjunto $N(J, v)$ definido de la siguiente forma:

$$N(J, v) = \{x \in I(J, v) : \theta(x) \leq_{Or} \theta(y), \forall y \in I(J, v)\}$$

dado un orden correspondiente $\leq_{Or}$.

El *nucleolus* es una de las opciones que existen a la hora de tratar de dar una solución a los problemas coalicionales de los que hemos hablado anteriormente, en este caso de trata de minimizar el grado de insatisfacción de las diferentes coaliciones posibles.

Bajo la condición de que la suma de los valores de todas las coaliciones individuales sea menor o igual que el valor de la coalición total, se obtiene que el *nucleolus* existe y es único. Además, este cumple las propiedades de pertenecer al core, de que si dos jugadores son simétricos entonces reciben el mismo pago, y de que si un jugador es pasivo y no afecta positivamente a ninguna coalición, entonces su pago es igual a su valor de coalición individual [19].

1.1.2. El Valor de Shapley

En esta sección, nos centraremos en otra manera que tiene la teoría de juegos de afrontar la búsqueda de soluciones para los problemas coalicionales: el valor de Shapley. Este fue propuesto por Lloyd Shapley en 1953 [20] y su objetivo es proporcionar un método que represente justamente la participación de cada jugador en las posibles coaliciones para llegar a una única asignación de pagos. A diferencia del método del *nucleolus*, que se focaliza en la estabilidad de la coalición, el valor de Shapley se preocupa por la equidad y justicia en la distribución, asegurándose de que cada jugador reciba una recompensa proporcional a la contribución marginal que aporta en todas las posibles coaliciones.

El valor de Shapley se caracteriza por una serie de propiedades deseables, como la eficiencia, la simetría, la nulidad y la aditividad, que lo convierten en una solución robusta y ampliamente utilizada en diversos campos, desde la economía hasta la inteligencia artificial.

La función de asignación de pagos $\phi(v)$ debe cumplir los siguientes axiomas de Shapley, como se describen en [19]:

1. **Eficiencia.** La función de asignación $\phi(v)$ debe distribuir el pago total del juego. Es decir, debe ser:
$$\sum_{i=1}^{n} \phi_i(v) = v(J)$$

2. **Simetría.** Para cualquier par de jugadores que realicen aportaciones equivalentes para cada coalición, es decir, tales que cumplan que:
$$v(S\cup\{i\}) = v(S\cup\{j\}), \quad \forall S \in \mathscr{P}(J), \text{ con } i,j \notin S$$
debe ser $\phi_i(v) = \phi_j(v)$.

3. **Tratamiento del jugador pasivo.** Si un jugador no aporta ningún beneficio adicional al resto de jugadores no debe recibir ningún pago adicional. Es decir, para cada jugador $i \in J$, para el cual se verifica que :
$$v(S) = v(S-\{i\}) + v(\{i\}), \text{ para toda coalición } S \text{ con } i \in S$$
debe ser $\phi_i(v) = v(\{i\})$.

4. **Aditividad.** La función de asignación ϕ debe ser invariante a cualquier descomposición arbitraria del juego. Formalmente, dados dos juegos cualesquiera (J,v_1) y (J,v_2) deber ser
$$\phi(v_1+v_2) = \phi(v_1) + \phi(v_2)$$

La única imputación que verifica que se cumplen estos cuatro axiomas se conoce como el valor de Shapley.

Definición 7. Valor de Shapley ($\phi(v)$). [19] Dado un juego coalicional de n jugadores, daremos una regla para determinar la imputación $\phi(v)$=$(\phi_1(v),\phi_2(v),\ldots,\phi_n(v))$ que verifica los axiomas de Shapley.

$$\phi_i(v) = \sum_{S\in\mathscr{P}(J)} q(s)[v(S) - v(S-\{i\})]$$

en donde $q(s) = \frac{(s-1)!(n-s)!}{n!}$, siendo $s = |S|$, el número de jugadores que hay en la coalición S.

El valor de Shapley considera todas las maneras de ordenar los n jugadores y, para cada una de estas, examina el valor que aporta añadir el jugador i en las diferentes posiciones. Si S es el conjunto de jugadores antes de añadir el jugador i, su valor de adhesión sería $v(S\cup\{i\}) - v(S)$. El pago $\phi_i(v)$ que se calcula con la fórmula anterior representa este valor medio de adhesión a todas las posibles coaliciones.

Proposición 1.1. El valor de Shapley definido anteriormente, pertenece al conjunto de imputaciones.

Demostración: Empecemos verificando que se cumple la racionalidad individual ($x_i \geq v(i)$, $1 \leq i \leq n$). Sea i un jugador cualquiera y $S \in \mathscr{P}(J)$ una coalición que no contiene a nuestro jugador i. Si tratamos de calcular cuál es el valor que añade este jugador a la coalición obtenemos $v(S\cup\{i\}) - v(S) \geq v(i)$. Como x_i es la media de estos valores para este tipo de coaliciones, se obtiene el resultado que buscábamos.

Continuemos ahora con la racionalidad grupal ($x_1 + \ldots + x_n = v(j)$). Si empezamos desde una coalición vacía y vamos añadiendo jugadores de uno en uno, calculando el valor que aporta cada uno hasta llegar al número total de jugadores obtenemos lo siguiente:

$$\begin{array}{r} v(1) \\ v(1,2) - v(1) \\ v(1,2,3) - v(1,2) \\ \vdots \\ + \, v(1,\ldots,n) - v(1,\ldots,n-1) \\ \hline v(1,\ldots,n) \end{array}$$

Indepedientemente del orden en el que añadamos los jugadores, llegamos a que el valor total es $v(J)$.Por lo tanto, $x_1 + \ldots + x_n$, que es la suma de los valores promedio agregados por los jugadores, será igual a $v(J)$, como se deseaba.

Ejemplo. Durante unas elecciones a las cuales se presentaban cuatro candidatos $\{1,2,3,4\}$, se han obtenido los siguientes porcentajes de votos $\{35,35,20,10\}$ correspondientes a cada uno, respectivamente. A continucion, calcularemos los valores de Shapley para ver como la importacia de los resultados que ha obtenido cada candidato como función de pago definiremos:

$$f(S) = \begin{cases} 1, & \text{si } \sum_{i \in S} p_i > 50 \\ 0, & \text{si } \sum_{i \in S} p_i \leq 50 \end{cases}$$

Esta función asigna el valor 1 a las coaliciones que representan una mayoria absoluta y 0 a las que no.

$$\begin{aligned} \phi_1(f) = &\frac{1}{4} f(\{1\}) + \frac{1}{12}[f(\{1,2\}) - f(\{2\})] + \frac{1}{12}[f(\{1,3\}) - f(\{3\})] + \frac{1}{12}[f(\{1,4\}) - f(\{4\})] + \\ &\frac{1}{12}[f(\{1,2,3\}) - f(\{2,3\})] + \frac{1}{12}[f(\{1,2,4\}) - f(\{2,4\})] + \frac{1}{12}[f(\{1,3,4\}) - f(\{3,4\})] + \\ &\frac{1}{4}[f(\{1,2,3,4\}) - f(\{2,3,4\})] = 0 + \frac{1}{12} + \frac{1}{12} + 0 + 0 + \frac{1}{12} + \frac{1}{12} + 0 = 1/3 \end{aligned}$$

De manera análoga, obtenemos que $\phi(f) = (\phi_1(f), \phi_2(f)\phi_3(f), \phi_4(f)) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$.

En este caso, los resultados del valor de Shapley le asignarían la misma importancia a los tres primeros candidatos, aunque el tercero tenga un menor porcentaje que los dos primeros. A su vez, se observa que el cuarto candidato no tiene importancia, a pesar de que su porcentaje no es nulo. Por otro lado, en este ejemplo podemos observar cómo se cumplen la propiedad de eficiencia $x_1 + x_2 + x_3 + x_4 = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} + 0 = 1 = f(1,2,3,4)$. Además, vemos que también se cumple la propiedad de simetría con los jugadores $\{1\}$ y $\{2\}$, y la propiedad de pasividad con el jugador $\{4\}$. En este caso, los resultados del valor de Shapley le asignarían la misma importancia.

1.2. Introducción a los Modelos de Inteligencia Artificial

Durante estas últimas décadas, la generación de datos en el mundo en que vivimos ha crecido de manera desmesurada. Esta gran cantidad de información se ha convertido en un recurso valioso que, si se trata de forma inteligente, puede lograr grandes avances en múltiples campos. Con el propósito de aprovechar estas ventajas de la mejor manera posible, surge la inteligencia artificial (IA) o el aprendizaje automático, mejor conocido como *machine learning (ML)*.

La *inteligencia artificial* es una rama de la informática que busca crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, la toma de decisiones y la resolución de problemas. El propósito principal de la IA es automatizar tareas complejas y mejorar la eficiencia en diversas aplicaciones.

El *machine learning* se define como el desarrollo de métodos y algoritmos que usan los ordenadores para "*aprender*" a partir de un conjunto de datos dados, con el fin de realizar una tarea, como podría ser una predicción o clasificación de un resultado.

Una de las ramas matemáticas que más ha contribuido al crecimiento de estos métodos de *machine learning* es la teoría de la probabilidad y la estadística, de la cual toma modelos prestados como los de regresión lineal en el caso de realizar predicciones o los de regresión logística para hacer clasificaciones. A su vez, esta disciplina también proporciona herramientas para verificar la calidad de los resultados.

1.2.1. Tipos de Aprendizaje Automático y Evaluación de Modelos

Dependiendo de la forma en que queramos tratar los diferentes tipos de datos y el objetivo que busquemos, podemos distinguir tres tipos de machine learning:

Aprendizaje supervisado

En este caso, nos interesa el uso de conjuntos de datos que incluyen tanto los datos de estudio como información sobre los resultados. Dado que la máquina ya posee los resultados que nos interesa obtener, los compara con las respuestas generadas a partir de los datos de estudio, tratando así de realizar un aprendizaje. Este grupo incluye diversos algoritmos como *Naive Bayes*, máquinas de vectores soporte, árboles de decisión o redes neuronales artificiales. Además, dentro de este grupo podemos distinguir dos grupos: los métodos de clasificación, cuyo objetivo es determinar a qué categoría pertenece una observación en función de los datos observados, y los métodos de regresión, que buscan devolver un valor numérico dado por una función continua en base a los datos de entrada.

Aprendizaje no supervisado

En este caso particular, no se cuenta con un conjunto de datos que incluya los resultados deseados, lo que impide la obtención de una solución específica. En cambio, el objetivo principal radica en buscar patrones y relaciones dentro de los datos para identificar diferentes grupos que permitan su clasificación. Es importante destacar que en estos algoritmos puede ser beneficioso que los resultados sean evaluados por una persona con conocimientos en el área relacionada con los datos. Dentro de esta categoría se encuentran métodos como el *clustering* (agrupamiento) con algoritmos como *k-means* y *Hierarchical*, así como otros métodos como el Análisis de Componentes Principales (PCA) y algoritmos como el *Apriori*.

Aprendizaje por refuerzo

Estos métodos de aprendizaje se fundamentan en la maximización de una recompensa, es decir, en evaluar si una acción es positiva o negativa. A diferencia de otros procedimientos, no requieren un conjunto de datos inicial, la máquina se adapta al entorno y mejora sus acciones a partir de la retroalimentación recibida. Estos métodos son capaces de enfrentarse a problemas complejos y logran obtener grandes resultados a largo plazo. Dentro de esta categoría, encontramos técnicas como *Q-Learning* en redes neuronales profundas y los *Policy Gradient Methods*.

Para profundizar en los conceptos de inteligencia artificial mencionados, se recomienda consultar el libro [12].

A la hora de evaluar la calidad del modelo que hemos obtenido deberiamos considerar tanto en los resultados logrados como en la capacidad de verificación y interpretación del método usado.

En general, cuando vamos a entrenar un modelo la hacemos de la siguiente manera, partimos el conjunto de datos en tres partes: entrenamiento, validación y test. El mayor volumen de datos se concentra en el grupo de entrenamiento, que es utilizado para construir el modelo. Esta construcción se evalúa con los datos de validación y, finalmente, se evalúa el modelo con los datos de prueba. Con esto conseguimos minimizar el riesgo de que nuestro modelo pueda ser demasiado sensible, es decir, que la modificación mínima de un dato de entrada provoque grandes cambios en el modelo o que pueda estar sobreentrenado, lo que significaría que le falta generalidad y le cueste trabajar con datos nuevos.

Otro aspecto significatio que se ha de tener en cuenta en nuestro modelo es la **interpretabilidad**. Esto concepto se puede definir como la capacidad que tiene el ser humano de enterder los resultados y causas de los resultados obtenidos mediante el *machine learning*.

Cuando mayor sea la interpretabilidad que posee un modelo más sencillo resultará comprender su comportamiento de manera que pueda llegar a ser explicado. Para aumentar el nivel de confianza que una persona le pueda otorgar al modelo, dependemos de este concepto para asegurar la presencia de los siguiente rasgos:

- **Equidad e imparcialidad** a la hora de hacer predicciones.
- **Fiabilidad**, para que pequeños cambios no afecten considerablemente la predicción.
- **Causalidad**, para que se pueda asociar una causa al resultado.

Algunos métodos, como los árboles de decisión, los modelos de regresión o los modelos lineales generalizados, son considerados modelos interpretables debido a su origen estadístico, que les aporta características que permiten evaluar la precisión que tiene un modelo y sus resultados. Por el contrario, tenemos los denominados modelos-agnósticos para los cuales presentan un mayor poder predictivo a costa de la interpretabilidad. En estos casos, es necesario que vayan acompañadas de herramientas como LIME (Local Interpretable Model-agnostic Explanations) o algoritmos relacionados con Shapley, que ayudan a poder determinar la contribución de cada variable.

La importancia de las variables dentro de un modelo de aprendizaje automático influyen en la precisión y la generalización que se puede hacer sobre un modelo. Con estos nos referimos al valor que aporta cada variable para llegar al resultado y comprender mejor su funcionamiento. Considerar la importancia de las variables es crucial para interpretar y confiar en los resultados del modelo, lo que nos permite comprender por qué toma ciertas decisiones.

Para evaluar cómo de importante resulta una variable, lo podemos enfocar de varias formas:

- La ganancia de información que busca medir cuanto aporta cada variable.
- La reducción del error que evalua la pérdida de precisión que provoca eliminar cada variable.
- El coeficiente de determinación que es la proporción de la varianza total de la variable explicada por la regresión.

Con los conocimientos introducidos sobre aprendizaje automático podemos hablar de uno de los problemas que ha surgido dentro de este campo en **los modelos de caja negra**. Estos se caracterizan por tener una estructura interna compleja y difícil de interpretar. Es cierto que este tipo de modelos puede lograr conseguir unos resultados muy favorables pero por otro lado su falta de interpretabilidad puede ser una gran desventaja.

En relación con el tema anterior, unido al auge que ha tenido el aprendizaje automático en el mundo durante estos últimos años, que ha conseguido estar presente en múltiples campos, ha evidenciado la necesidad de regular su aplicación. Aunque en ámbitos como el sistema de recomendación de películas de una plataforma no sea relavante una explicación, en otros casos, especialmente en el terreno económico o de salud, esto si atesora gran importancia. Por esta razón, esta falta de transparencia puede desvirtuar la confianza en un modelo y plantear preocupaciones éticas y legales sobre su uso.

Actualmente, podemos ver que en Europa se esta trabajando en su regulación [8].

1.2.2. Intelegencia Artigicial Explicada (XAI)

Debido a lo introducido anteriormente, se comienza el desarrollo de la siguiente rama, la **inteligencia artificial explicada (XAI)**. Esta se centra en la producción de detalles y razones que nos ayuden a que nuestro modelo sea más sencillo de entender. Esto puede implicar que para lograr una mayor interpretabilidad del modelo en busca de una explicación coherente sea necesario sacrificar algo de rendimiento. Con este factor explicatorio, podemos lograr ventajas tanto a nivel de resultados como a nivel de modelos, identificando las debilidades que muestran y debemos mejorar.

Estos son los cuatro puntos principales en los que nos debemos focalizar. Para mayor conocimiento del tema podemos consultar [2].

- **Explicabilidad de los datos**: Esto engloba las técnicas aplicadas sobre nuestro conjunto de datos inicial para conseguir una descripción detallada de los mismos.
- **Explicabilidad del modelo**: Este término se refiere a la capacidad de los modelos para ofrecer una comprensión propia o la utilización de herramientas que faciliten esta comprensión.
- **Explicabilidad de los resultados**, podría considerarse uno de los aspectos más importantes dentro de este campo y se encarga del tratamiento de métodos y algoritmos que buscan explicar el modelo una vez obtenido los resultados.
- **Evaluación de las explicaciones**, se enfoca en medir cúan adecuada es la explicación en relación con su objetivo y a quien se dirige.

Capítulo 2

Aplicaciones del Valor de Shapley en Machine Learning

Dentro de este contexto que hemos ido desarrollando durante el primer capítulo, explicábamos diferentes aspectos de la teoría de juegos y la inteligencia artificial, con el fin de sentar una buena base de conocimiento. Tratando conceptos como la importancia de la explicabilidad en los modelos de *machine learning* y el uso de herramientas como el valor de Shapley que nos ayuden a mejorar la interpretabilidad de nuestros modelos, véase [14] y [15].

Partiendo de esta situación, durante este segundo capítulo profundizaremos en el análisis y la aplicación del valor de Shapley para modelos de *machine learning*, centrándonos en su utilidad y funcionamiento en modelos interpretables, como los modelos de regresión. A su vez, también hablaremos de métodos de aproximación del cálculo que mejoran su rendimiento computacional y algún otro procedimiento para potenciar la interpretabilidad de los modelos.

2.1. Modelos de Regresión y su Interpretación con el Valor de Shapley

En esta sección, nos centraremos en las bases fundamentales de los modelos de regresión y exploraremos cómo podemos aplicar el valor de Shapley en el contexto del aprendizaje automático. Abordaremos los conceptos clave en modelos de regresión para comprender mejor sus fundamentos [18], facilitando así la comprensión de los enfoques de Shapley. Presentaremos dos perspectivas del valor de Shapley en *machine learning*: una centrada en una comprensión global del modelo [16] y otra que nos permite adoptar un enfoque individual [1].

2.1.1. Fundamentos de Modelos de Regresión y su Interpretación

Los modelos de regresión tienen sus raíces en los primeros desarrollos estadísticos del siglo XIX. Posteriormente, a lo largo del siglo XX, los modelos de regresión fueron evolucionando para ajustarse a situaciones y datos cada vez más complejos, desarrollándose como el método de los mínimos cuadrados. Gracias al avance tecnológico del siglo XXI, estos han seguido mejorando, teniendo un papel relevante en el campo del aprendizaje automático.Luego, podemos considerar los modelos de regresión como una herramienta fundamental en la ciencia de datos.

Antes de empezar a explicar en qué consisten los modelos de regresión, vamos a introducir algunos conceptos que nos ayuden a comprenderlos mejor:

- **Variables predictora:** estas también se conocen como variables independientes, dentro de los modelos de regresión se usan para predecir o explicar la variable respuesta.
  Notación: $\{x_i\}_{i=1}^n = \{(x_{i1}, x_{i2}, \ldots, x_{ik})\}_{i=1}^n$, siendo n el número de datos y k el número de variables predictoras.

- **Variable respuesta:**es conocida de igual modo como variable dependiente, esta se trata de explicar a partir de las variables predictoras del modelo.
  Notación: $\{y_i\}_{i=1}^n$ siendo n el número de datos.

- **Coeficientes de regresión:**son los parámetros del modelo encargados de representar la relación que hay entre la variable o variables predictoras. Cada coeficiente representa el cambio que sufriría la variable respuesta si se modifica en una unidad la variable predictora correspondiente, manteniendo las demás variables constantes.
  Notación: $\beta = (\beta_1, \ldots, \beta_k)$, siendo k el número de variables predictoras, donde β_0 es el término del intercepto que representa el valor esperado cuando el resto de variables independientes es cero y $\beta_j; j = 1, \ldots, k$ las variables de la regresión parcial.

- **Término del error:** también llamado error estándar, es una estimación de la cantidad que puede variar entre el valor de una estadística y el valor de la muestra.
  Notación: $\varepsilon_i \sim N(0, \sigma); i = 1, \ldots, n$, este error tiene se asemeja a una distribución normal de media 0 y desviación típica σ, siendo n el número de datos.

Respecto a los modelos de regresión, podemos distinguir dos tipos: los lineales y los no lineales. En los lineales, la relación entre las variables predictoras y las variables respuesta es lineal, y buscan representar esta relación con una ecuación donde los coeficientes de regresión determinan la pendiente y la intersección de la línea. Mientras que en los no lineales, donde la relación que puede existir entre las variables predictoras y la variable respuesta se aleja notoriamente de la linearidad. Esta relación podría ser polinómica, exponencial o logarítmica.

A continuación, vamos a tratar dos tipos de modelos de regresión: el modelo lineal múltiple y el modelo logístico.

Regresión lineal múltiple

Este tipo de modelos consta de varias variables predictoras, siguiendo la notación que hemos establecido antes, podemos representarlo con la siguiente ecuación.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon = \beta_0 + \beta^\top x + \varepsilon \tag{2.1}$$

Para conseguir calcular el valor de los coeficientes de regresión de una manera óptima tenemos varios métodos. Generalmente, el más habitual es el de mínimos cuadrados, que busca minimizar la diferencia entre los resultados reales y los resultados estimados, expresado a través de la siguiente fórmula.

$$\hat{\beta} = \arg\min_{\beta_0,\ldots,\beta_k} \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \right) \right)^2 \tag{2.2}$$

Siendo estos $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k)$ los valores de los coeficientes de regresión ajustados.

Para que un modelo regresión sea puede considerar "*correcto*", es necesario que cumplan los siguientes supuestos de la forma más satisfactoria posible:

- **Linealidad.** En teoría, la relación que debe existir entre la variable respuesta y las variables predictoras debe ser lineal o acercarse lo máximo posible a esta. Si esto no se consigue respetar, es más probable que se den errores en nuestro modelo.

- **Normalidad.** Estamos suponindo que los residuos del modelo siguen una distribución normal, esto es importante a la hora de validar los intervalos de confianza.

- **Homocedasticidad.** Con esto suponemos que la varianza de los residuos es constante para todos los valores de las variables predictoras, en caso contrario la eficiencia de los estimadores se puede ver afectada.

- **Independencia.** Consideramos que cada observación de nuestro conjunto de datos es independiente entre sí. Si no se cumpliera este supuesto, las predicciones generadas por el modelo no serían confiables y se complicaría la generalización del modelo.
- **Ausencia de multicolinealidad.** No queremos que las características que explican nuestro modelo estén relacionadas entre sí. Esto podría provocar que los valores de los coeficientes de regresión ajustados no sean buenos, pues si hay correlación entre variables se complica el hecho de cómo repartir la atribución de efectos.

Para medir la calidad y precisión que tienen los modelos se usan herramientas estadísticas como: el error cuadrático medio y el coeficiente de determinacion (R^2).

El error cuadrático medio es una medida de magnitud del error entre los datos observados (Y) y los datos obtenidos por el modelo ($\hat{Y}$). Su valor se obtiene como la raíz cuadrada del promedio de estos y cuanto menor sea este, más probable es que el error de predicción promedio del modelo sea menor.

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

El coeficiente de determinación, denotado comúnmente como R^2, es una cuantía de la proporción de la varianza en la variable respuesta que es explicada por las variables predictoras en el modelo de regresión. Cuanto mayor sea el R^2, mejor explicará el modelo los datos, aunque esto no es una garantía para predicciones fuera de la muestra.

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

Regresión logística

Los modelos de este tipo están pensados para la clasificación. En vez de ajustar una recta o un hiperplano, los modelos de regresión logística buscan forzar que los resultados de salida esten entre cero y uno. La función logística se define como:

$$\text{logistic}(\eta) = \frac{1}{1+\exp(-\eta)} \tag{2.3}$$

Partiendo de la ecuación anterior para modelos lineales múltiples (2.1) y la ecuación logistica, podemos obtener la siguiente ecuación logística con probabilidades entre 0 y 1.

$$P(y_i = 1) = \frac{1}{1+\exp(-(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik}))} \tag{2.4}$$

Como se aprecia, la interpretación de los pesos de la ecuación no se puede interpretar directamente de una manera lineal, al igual que antes. Luego, para mejor interpretación, es necesario realizar una reformulación buscando dejar despejada la expresión lineal.

$$\log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \log\left(\frac{P(y=1)}{P(y=0)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k \tag{2.5}$$

Esta expresión inicial del logaritmo se conoce como *log-odds*.Esta formulación nos va a admitir poder realizar una interpretación de los cambios de valor en una característica. Si despejamos el logartimo tenemos que $odds = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)$, lo que nos va a permitir poder entender de una manera más intuitiva los cambios de la variable x_j dejando el resto de variables fijas. De modo que las probabilidades estimadas por el cambio se miden como:

$$\frac{odds_{x_j+1}}{odds} = \frac{exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_j(x_j+1) + \ldots + \beta_k x_k)}{exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_j x_j + \ldots + \beta_k x_k)} = \exp(\beta_j(x_j+1) - \beta_j x_j) = \exp(\beta_j)$$

Por tanto, este cambio descrito viene determinado simplemenete por $\exp(\beta_j)$.

2.1.2. Shapley con Análisis de la Varianza

Durante este trabajo hemos ido destacando el uso del valor de Shapley como una herramienta clave para distribuir equitativamente las contribuciones de cada participante en una coalición. Este concepto se traduce directamente al contexto de modelos de regresión, donde buscamos entender cómo cada variable contribuye a la predicción de un resultado.

En este primer enfoque, con una perspectiva general del modelo, que busca proporcionar una visión global del impacto de las diferentes características en la predicción, facilitando una interpretación más profunda y detallada del comportamiento del modelo. Introducimos una visión de Análisis de Varianza (ANOVA), una metodología estadística que permite descomponer la varianza total de un modelo en componentes atribuibles a diferentes conjuntos de variables aleatorias.

Para entender mejor esta descomposición, consideraremos la variable aleatoria $g(x) \in \mathbb{R}$ asumiendo que $E(g(x)^2) < \infty$, de modo que $E(g(x)) = \mu$ y $E((g(x)-\mu)^2) = \sigma^2$. Exploramos cómo el ANOVA distribuye esta varianza σ^2 entre los subconjuntos de variables aleatorias, siendo K el conjunto de todas las variables, de modo que habrá $2^{|K|}-1$ subconjuntos no vacíos. Estaremos trabajando con subconjuntos $\{t_1, t_2, \ldots, t_{|S|}\} = S \in \mathscr{P}(K)$, denotaremos su complementario como $-S$ y escribiremos el conjunto de variables aleatorias como $x_S = (x_{t_1}, x_{t_2}, \ldots, x_{t_{|S|}})$.

La idea principal es la descomposición de la varianza σ^2 como la suma de la varianza de diferentes conjuntos de variable aleatorias. Usaremos la siguiente descomposición, $g(x) = \sum_{S \subseteq K} g_S(x)$, donde estos $g_S(x)$ dependen solo de x_t para los índices $t \in S$. Notar que hay muchas posibles descomposiciones, definiéndose este ANOVA de manera recursiva como:

$$g_S(x) = \mathbb{E}[g(x) \mid x_S] - \sum_{s \subset S} g_s(x)$$

Donde cada s en un subconjunto propio de S, y definimos $\sigma_u^2 = var(g_u(x))$, expresándose la varianza como $\sigma^2 = var(g(x)) = \sum_{S \subseteq [K]} \sigma_u^2$.

Asimismo, a partir de esta descomposición de la varianza, podemos obtener los índices Sobol. Distinguimos dos versiones principales de los índices Sobol, que aunque no coinciden con el valor de Shapley, nos pueden ayudar con su cálculo, pues sirven como cota superior e inferior de este . Por lo tanto, podemos usar los índices Sobol para delimitar el valor de Shapley. A su vez, el valor de Shapley es un punto medio muy razonable entre los dos índices Sobol.

Definición 8. Los índices de Sobol, dado un subconjunto S, se definen como:

$$\underline{\tau}_S^2 = \sum_{s \subseteq S} \sigma_s^2 \quad \text{y} \quad \bar{\tau}_S^2 = \sum_{s : s \cap S \neq \emptyset} \sigma_s^2$$

estos satisfacen que $\underline{\tau}_S^2 \leq \bar{\tau}_S^2$ y que $\bar{\tau}_S^2 = \sigma^2 - \underline{\tau}_{-S}^2$.

Para más detalles sobre esta forma de análisis de la varianza, consutar [17] y [16].

Ahora que hemos establecido cierto contexto sobre la descomposición de la varianza y la terminología que vamos a utilizar, podemos adentrarnos en nuestro enfoque general sobre los modelos de regresión.

Dado un subconjunto $S \subseteq K$ de variables, podemos expresar el poder explicativo de este grupo como:

$$val(S) = \underline{\tau}_S^2 = \operatorname{var}(\mathbb{E}[g(x) \mid x_S]) \tag{2.6}$$

La siguiente definición que vamos a dar para el cálculo de los valores de Shapley cumple los axiomas mencionados en la sección 1.1.2.

$$\phi_j = \frac{1}{k} \sum_{S \subseteq -\{j\}} \binom{k-1}{|S|}^{-1} (\underline{\tau}_{S+\{j\}}^2 - \underline{\tau}_S^2) \tag{2.7}$$

Como vemos en la ecuación que acabamos de dar, el valor de Shapley está definido siempre y cuando $\operatorname{var}(\mathbb{E}[g(x) \mid x_S])$ esté definida. Además, se tiene que $x_{S+\{j\}}$ siempre tiene al menos el mismo poder explicativo que x_S.

Teorema 2.1. *Sea la descomposición ANOVA de una función con k entradas independientes y componentes de varianza σ_S^2 para $S \subseteq K$. Si el valor de un subconjunto S de variables es $val(S) = \underline{\tau}_S^2$, entonces el valor de Shapley de la variable j es:*

$$\phi_j = \sum_{S \subseteq K,\ j \in S} \frac{\sigma_S^2}{|S|}$$

Demostración: Ver en Anexos (I.1).

En el caso de funciones lineales, habitualmente se suele aplicar el valor de Shapley para repartir el coeficiente de determinción R^2 de una regresión entre las variables. Sea $f(x) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$ donde los x_j son independientes con varianzas σ_j^2. Es entonces sencillo ver que $\phi_j = \sigma_j^2 \beta_j^2$. Si calculamos la esperanza condicionda, tenemos que $A = \mathbb{E}[f(x) \mid x_S] = \beta_0 + \sum_{j \in S} \beta_j x_j$, luego evaluamos $\mathrm{Var}(A) = \sum_{j \in S} \beta_j^2 \mathrm{Var}(x_j) = \sum_{j \in S} \beta_j^2 \sigma_j^2$, esto se debe a la independencia de las variables, y finalmente, por la cancelación de componentes y como el proceso de cálculo del valor de Shapley se asemeja a una especie de promedio, tenemos el resultado. Además, se puede realizar una reparametrización de x_j a cx_j para $c \neq 0$, entonces β_j se convierte en β_j / c. La dependencia entre los x_j, puede provocar efectos negativos durante el cálculo de Shapley en configuraciones lineales.

Abordando ya el cálculo de Shapley en estos modelos, tenemos que, identificando el valor de un subconjunto S de variables como R_S^2, podemos ver la aportación de la variable predictora x_j en el valor de R^2.

$$\phi_j = \frac{1}{k} \sum_{S \subseteq -\{j\}} \binom{k-1}{|S|}^{-1} (R_{S+\{j\}}^2 - R_S^2) \tag{2.8}$$

Esta medida se conoce como *LMG* debido a sus autores [11] y es una forma conveniente de partir nuestro R^2, aunque si hay un elevado número de variables predictoras supone un alto coste computacional.

Teorema 2.2. *Si $f(x) = \beta_0 + \beta^\top x$ para $x \sim N(\mu, \Sigma)$ donde $\Sigma \in \mathbb{R}^{k \times k}$ es una matriz simétrica semidefinida positiva de rango completo, entonces el efecto de Shapley para la variable j es:*

$$\phi_j = \frac{1}{k} \sum_{S \subseteq -\{j\}} \binom{k-1}{|S|}^{-1} \frac{Cov(x_j, x_{-S}^\top \beta_{-S} \mid x_S)^2}{Var(x_j \mid x_S)}$$

Demostración: Ver en Anexos (I.2).

Esta definición del valor de Shapley trabaja con varianzas condicionales, por lo que la distribución gaussiana hace que estas sean muy convenientes debido a las expresiones matriciales que podemos usar para su cálculo.

Este enfoque desde un punto de vista general del modelo tiene el potencial de mejorar el rendimiento del modelo. Incorporando la posibilidad de mejorar los entrenamientos reajustando los pesos, puesto que disponemos de mayor información sobre la importancia de las características. A su vez, acercamos estos al nivel de comprensión humana, permitiéndonos la oportunidad de disponer de datos sólidos con los que se puedan formular explicaciones. Sin embargo, es cierto que también presenta debilidades, como el alto costo computacional al trabajar con muchas características y la disminución de la precisión cuando hay una alta correlación entre las características.

2.1.3. SHAP: Evaluación Individual y Descomposición Aditiva

En este segundo enfoque presentamos Shapley desde una perspectiva de predicción individual de las predicciones, este se denomina SHAP. Volviéndonos a apoyar en la teoría de juegos de coalición, SHAP calcula los valores de Shapley, considerando las características de los datos como jugadores en una coalición y asignando equitativamente la "*recompensa*" de la predicción entre ellas. Este método nos va a permitir tener una interpretación detallada tanto de características individuales como de grupos

de características, facilitando la comprensión de cómo cada elemento influye en la predicción. Además, SHAP introduce una representación innovadora del valor de Shapley como un método de atribución de características aditivas, lo que nos proporciona explicaciones claras y constrastables al comparar la predicción con la predicción promedio.

Como punto de partida habitual en el aprendizaje automático, dispondremos de un conjunto de entrenamiento $\{y_i, x_i\}_{i=1,\dots,n}$ que se usará para entrenar un modelo predictivo $g(x)$ que aproxime lo mejor posible los valores de y. Si queremos explicar la predicción del modelo $g(x^*)$ para un vector de características específico $x = x^*$, tenemos que la predicción se descompone de la siguiente manera:

$$g(x^*) = \phi_0 + \sum_{j=1}^{k} \phi_j^*$$

Donde $\phi_0 = \mathbb{E}[g(x)]$ se puede entender cómo una predicción que no se debe a ninguna de las características, sino simplemente a la predicción promedio global y ϕ_j^* es el j-ésimo valor de Shapley para la predicción $x = x^*$. El propósito de los valores de Shapley es explicar la diferencia entre la predicción $g(x^*)$ y la predicción promedio global.

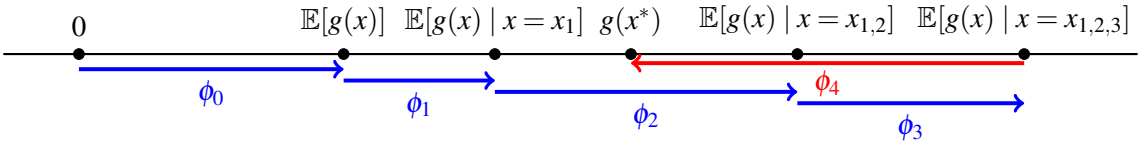


Figura 2.1: Ejemplo de los valores de SHAP para un modelo con cuatro características.

Un modelo de esta forma es un método de atribución de características aditivas que verifica las propiedades enumeradas en la sección 1.1.2.

- **Eficiencia.** La suma de los valores de Shapley para las diferentes características es igual a la diferencia entre la predicción y la predicción promedio global $\sum_{j=1}^{k} \phi_j = g(x^*) - \mathbb{E}[g(x)]$. Esta propiedad asegura que la parte del valor de la predicción que no se explica por la predicción media global (ϕ_0), se atribuye y es explicada por las características, de modo que podemos comparar los ϕ_j's de diferentes predicciones $g(x^*)$.
- **Simetría.** Si dos características diferentes contribuyen de manera igual a la predicción, cuando se combinan con cualquier otro subconjunto de características, sus valores de Shapley son iguales sí. No cumplir con este criterio sería inconsistente y daría explicaciones poco confiables.
- **Jugador nulo.** Es natural asignarle un valor de Shapley nulo a una característica que no cambia la predicción, independientemente de con qué otras características se combine.
- **Linealidad.** Cuando una función de predicción consiste en una suma de funciones de predicción, el valor de Shapley para una característica es idéntico a la suma de los valores de Shapley de esa característica de cada una de las funciones de predicción individuales.

Para poder calcular los valores de Shapley en el contexto de la explicación de predicciones, necesitamos definir nuestra función pago $v(S)$ para un cierto subconjunto S; esta debe tratar de asemejarse al valor de $g(x^*)$ cuando solo conocemos el valor de las características del subconjunto S. Para cuantificar esto, usaremos la salida esperada del modelo predictivo, condicionada a los valores de las características x_S.

$$v(S) = \mathbb{E}[g(x)|x_S = x_S^*]$$

Teorema 2.3. *En el caso de que el modelo predictivo que vayamos a usar sea de regresión lineal* $f(x) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$, *con todas las características* x_j, $j = 1, \dots, k$ *independientes los valores de Shapley tendran la siguiente expresión:*

$$\phi_0 = \beta_0 + \sum_{j=1}^{k} \beta_j \mathbb{E}[x_j] \ , \quad y \quad \phi_j = \beta_j (x_j^* - \mathbb{E}[x_j]) \ , \quad j = 1, \dots, k \tag{2.9}$$

Demostración: Partimos de la expresión de $v(S)$:

$$v(S) = \mathbb{E}[f(x)|x_S = x_S^*] = \int f(x_{-S}, x_S^*)\ \rho(x_{-S}|x_S = x_S^*)\, dx_{-S} \tag{1}$$

$$= \int \left(\sum_{i \in -S} \beta_i x_i + \sum_{i \in S} \beta_i x_i^* \right) \rho(x_{-S}|x_S = x_S^*)\, dx_{-S} \tag{2}$$

$$= \sum_{i \in -S} \beta_i \int x_i\ \rho(x_i)\, dx_i \ + \ \sum_{i \in S} \beta_i\ x_i^* \int \rho(x_{-S}|x_S = x_S^*)\, dx_{-S} \tag{3}$$

$$= \sum_{i \in -S} \beta_i\ \mathbb{E}[x_i] \ + \ \sum_{i \in S} \beta_i\ x_i^*$$

Para este desarrollo, hemos utilizado la expresión integral de la esperanza condicional. El paso de (1) a (2) se debe a la suposición de linealidad, lo que nos lleva a separar las variables. La transición siguiente, de (2) a (3), se sigue de la suposición de características independientes. Ahora, con una expresión más simple de $v(S)$, es fácil ver que:

$$v(S \cup \{j\}) - v(S) = \beta_j\ \left(x_j^* - \mathbb{E}[x_j]\right) \tag{4}$$

A partir de (4), vemos que en este caso la diferencia $v(S \cup \{j\}) - v(S)$ es independiente de S. Por lo tanto, la fórmula de Shapley puede escribirse como:

$$\phi_j = \beta_j\ \left(x_j^* - \mathbb{E}[x_j]\right) \sum_{S \subseteq K \setminus \{j\}} \frac{|S|!(k - |S| - 1)!}{k!}$$

Dado que la suma de los pesos de Shapley es igual a 1, tenemos el resultado buscado:

$$\phi_j = \beta_j (x_j^* - \mathbb{E}[x_j])\ , \quad \text{para}\ \ j = 1, \ldots, k$$

Notar que para facilitar la notación, aquí y a continuación, omitiremos el superíndice $*$ para los valores ϕ_j. De modo que cada predicción $f(x^*)$ resultará en diferentes conjuntos de valores ϕ_j.

A partir de esta idea de SHAP para tratar las predicciones de un modelo de forma individual, también podemos realizar un análisis de la importancia global de las características en el modelo. Definiendo la importancia de la función SHAP como:

$$I_j = \sum_{i=1}^{n} \left| \phi_{i,j} \right|$$

Siendo I_j la importancia de la característica j en el modelo.

Este enfoque que hemos desarrollado nos brinda una amplia gama de información, permitiendo explicaciones detalladas y comparaciones entre distintas predicciones. Además, nos posibilita la creación de múltiples estilos de gráficos para interpretar los datos obtenidos, así como el análisis de interacciones entre variables y una visión global de los resultados. No obstante, la falta de una fórmula explícita para casos de características dependientes y la complejidad computacional asociada al crecimiento exponencial del número de subconjuntos posibles limitan su aplicabilidad.

2.2. Soluciones Computacionales Basadas en el Valor de Shapley

Debido a una clara debilidad que hemos ido mencionando previamente, el valor de Shapley posee un alto coste computacional, especialmente en conjuntos de datos grandes o modelos complejos. Como solución a esta limitación, han ido surgiendo varios enfoques para aproximar el valor de Shapley de manera más eficiente.

2.2.1. Kernel SHAP

Kernel SHAP es una implementación popular del valor de Shapley que se basa en una interpretación alternativa del mismo. Para cada instancia de los valores x, estima las contribuciones de cada valor de característica a la predicción.

Antes de señalar los pasos concretos a dar para el cálculo de Kernel SHAP, vamos a contextualizar ciertos aspectos que nos ayuden a comprender mejor su funcionamiento. Este método lo podemos dividir en dos partes: encontrar una aproximación inteligente y manejable desde el punto de vista computacional para calcular los valores de Shapley y una manera simple de estimar las funciones de pago $v(S)$.

Para tratar de suavizar y reducir el número de cuentas necesarias, lo que se busca es evitar la realización de todos los cálculos de todas las coaliciones. Se puede obtener una aproximación consistente a partir de un subconjunto. En este podemos forzar la presencia de las coaliciones con solo una característica y con todas menos una característica que son bastante relevantes en la influencia del cálculo, pues nos aportan información de los efectos que tiene una característica de forma aislada. Además, queremos seleccionar coaliciones de manera que sigan la lógica de probabilidad y respeten su relevancia aparente en relación con el valor de Shapley. Para ello, usamos una distribución de probabilidad que siga el núcleo de Shapley, con la que también evitamos la presencia de la coalición vacía o de la coalición total. La fórmula del núcleo de Shapley es la siguiente:

$$\pi(S) = \frac{(k-1)}{\binom{k}{|S|}|S|(k-|S|)} \tag{2.10}$$

Por lo general, partimos siempre de una coalición S cualquiera en la que tenemos información de las carácteristas presentes y las ausentes. Esta información nos gustaría que representara un valor en función de la presencia de las características. De modo que se les asigne a las que están presentes su valor y a los que no lo están se les dé uno aleatorio de nuestros datos. La siguiente función será la encargada de este trabajo:

$$v(S) = E[g(x) \mid x_S = x_S^*]$$

Para calcular los valores exactos de $v(S)$, necesitaríamos la distribución condicional $\rho(x_{-S} \mid x_S = x_S^*)$, la cual rara vez se conoce. Debido a esto, durante el procedimiento, el método Kernel SHAP asumirá la independencia de las características para poder reemplazar $\rho(x_{-S} \mid x_S = x_S^*)$ por $\rho(x_{-S})$. Para la estimación de esta distribución se proponen cuatro enfoques en los que se profundiza más en [1]: asumir una distribución gaussiana para $\rho(x)$; asumir una distribución gaussiana copular para $\rho(x)$; aproximar $\rho(x_{-S} \mid x_S = x_S^*)$ mediante una distribución empírica (condicional); y una combinación del enfoque empírico con el enfoque gaussiano o el enfoque gaussiano copular.

Esta asunción de independencia provoca que se ignore la posible dependencia que hay entre las características ausentes y presentes. Por lo que, Kernel SHAP sufre el mismo problema que todos los métodos de interpretación basados en permutación.

A la hora de entrenar el modelo usaremos la siguiente función de pérdida que guarda cierta relación con la clásica fórmula de suma de errores al cuadrado que generalmente optimizamos para los modelos lineales. Los coeficientes estimados del modelo, los ϕ_j's son los valores de Shapley.

$$\sum_{S \subseteq K} \pi(S) \left(v(S) - \phi_0 + \sum_{j \in S} \phi_j \right)^2 \tag{2.11}$$

En resumen, los pasos a dar para el cálculo del Kernel SHAP son:

1. Dadas las diferentes coaliciones $S \in \mathscr{P}(K)$, vemos el valor que se le otorga nuestra supuesta función de pago $v(S)$.

2. Además, para estas coaliciones S se calcula el peso del núcleo SHAP (2.10).

3. Cuando ya hemos calculado el número de subconjuntos que hallamos considerados oportunos, podemos formular una especie de ecuación de regresión lineal $(\phi_0 + \sum_{j \in S} \phi_j)$ con la presencia de los valores de Shapley en función de cómo son nuestros subconjuntos.

4. Finalmente, realizamos la optimización de la función de pérdida (2.11), y obtenemos los valores de Shapley ϕ_j's, que hemos mencionado en la regresión anterior.

2.2.2. Tree SHAP

TreeSHAP [15] es una variante de SHAP diseñada específicamente para modelos de aprendizaje automático basados en árboles, como árboles de decisión, random forest y gradient boosted trees. Aprovechando la eficiencia computacional que le proporciona la estructura del árbol, destaca por su rapidez y exactitud en el cálculo de los valores de Shapley, esenciales para interpretar la contribución de cada característica en la predicción del modelo. Además, ofrece dos versiones: el estimador intervencional y el dependiente del camino del árbol, que se adaptan a diferentes contextos de análisis.

El estimador intervencional

La estimación se realiza recorriendo recursivamente el árbol. Formaremos coaliciones de valores de características que incluyen tanto características presentes como ausentes. En lugar de comenzar con todas las coaliciones, podemos invertir el proceso y explorar los caminos del árbol para determinar qué coaliciones producirían predicciones diferentes, ya que muchos cambios en las características pueden no tener impacto en la predicción. La parte complicada es la ponderación y combinación de manera precisa de las predicciones basadas en las características alteradas.

El estimador dependiente del camino del árbol

Este método se apoya en la esperanza condicional. La idea de cómo el estimador dependiente del camino del árbol calcula la predicción esperada para un solo árbol, una instancia x y un subconjunto de características S es la siguiente. Si condicionamos todas las características, lo que significa que S contiene todas las características, entonces la predicción del nodo en el que cae la instancia x sería la predicción esperada. Por el contrario, si no condicionamos ninguna característica, lo que significa que S está vacío, utilizamos el promedio ponderado de las predicciones de todos los nodos terminales. Si S incluye algunas, ignoramos las predicciones de los nodos inalcanzables. Un nodo se considera inalcanzable si el camino de decisión hacia él contradice los valores en x_S. A partir de los nodos terminales restantes, promediamos las predicciones ponderadas por el tamaño de los nodos, esto produce la predicción esperada para x dado S. El desafío es que tenemos que aplicar este procedimiento para cada posible subconjunto S. El concepto fundamental es empujar todos los posibles subconjuntos S por el árbol simultáneamente. El algoritmo debe realizar un seguimiento del peso general de los subconjuntos en cada nodo y un registro del número de subconjuntos.

Para conjuntos de árboles, podemos promediar los valores SHAP, con la contribución de la predicción de cada árbol influyendo en el peso final del conjunto. La aditividad de los valores SHAP permite que los valores de Shapley de un conjunto de árboles sean el promedio ponderado de los valores de Shapley de los árboles individuales.

En comparación con KernelSHAP, TreeSHAP ofrece una alternativa más rápida, eficiente y precisa. Además, TreeSHAP no necesita acceso a los datos para calcular los valores de Shapley, y el uso de la predicción condicional esperada lo hace más robusto frente a la correlación entre características. Sin embargo, la misma correlación abre la posibilidad de estimaciones no nulas para características sin efecto en la función de predicción, lo que es importante tener en cuenta al utilizar TreeSHAP.

2.2.3. FastSHAP

FastSHAP [9] es un enfoque innovador que busca la generación de explicaciones mediante el valor de Shapley en tiempo real. Proponemos aprender un modelo explicativo separado que proporcione estimaciones precisas del valor de Shapley en una sola pasada hacia adelante. Para un enfoque basado en el

aprendizaje, parecería necesario un gran conjunto de entrenamiento de valores de Shapley de referencia, lo cual sería computacionalmente inviable. En cambio, con este novedoso enfoque se entrena un modelo explicativo minimizando una función objetivo inspirada en la caracterización de mínimos cuadrados ponderados del valor de Shapley, lo que permite una optimización eficiente basada en gradientes.

El procedimiento a seguir es:

a) Se entrena una función paramétrica $\phi_{\text{fast}}(x, y; \theta)$ que genere los valores de Shapley por separado para cada muestra.

b) En lugar de necesitar conjuntos de datos de valores de Shapley de referencia, FastSHAP entrena la función $\phi_{\text{fast}}(x, y; \theta)$ con la idea de penalizar las predicciones del modelo, para que se ajusten bien a la caracterización de mínimos cuadrados ponderados del valor de Shapley que tenemos a continuación:

$$L(\theta) = \mathbb{E}_{p(x)}\mathbb{E}_{\text{Unif}(y)}\mathbb{E}_S\left[\left(v_{x,y}(S) - v_{x,y}(0) - S^\top \phi_{\text{fast}}(x, y; \theta)\right)^2\right]$$

Aquí, S simbiliza un subconjunto de características o variables, $\text{Unif}(y)$ representa una distribución uniforme sobre las clases, y $\mathbb{E}_{p(x)}$ denota la esperanza respecto a la distribución de probabilidad $p(x)$.

c) Para garantizar que las predicciones del modelo respeten la propiedad de eficiencia del valor de Shapley, existen dos métodos posibles. El primero consiste en ajustar las predicciones del modelo para que la suma sea correcta mediante la aplicación de una normalización. El segundo método implica la incorporación de un término de penalización directamente en la función de pérdida.

Los experimentos realizados en [9] muestran que FastSHAP produce estimaciones precisas de los valores de Shapley con una velocidad significativamente mayor en comparación con otros enfoques. Se estima que para alcanzar la precisión de FastSHAP, KernelSHAP tardaría unas 600 veces más y, en comparación con otros enfoques avanzados, se emplearía 200 veces más de tiempo. Además, la generación de explicaciones para imágenes de alta calidad es viable con este método. Con FastSHAP, se hace posible la adopción de los valores de Shapley en modelos a gran escala, resolviendo el problema del alto costo computacional.

2.3. Otro Enfoque Agnóstico y Técnicas Concretas Basadas en Shapley

Con el fin de ofrecer una perspectiva adicional sobre la interpretación de modelos de aprendizaje automático, exploraremos a continuación enfoques alternativos y específicos que utilizan el valor de Shapley.

2.3.1. LIME

LIME *(Local Interpretable Model-agnostic Explanations)* [14] es otra técnica desarrollada para la interpretación de modelos-agnósticos en *machine learning*. La clave de LIME es que en lugar de entrenar un modelo sustituto global, se enfoca en entrenar modelos sustitutos locales para poder explicar las predicciones de manera individual.

La idea subyacente es la siguiente: dejaremos de lado los datos de entrenamiento y, sin conocer los detalles internos del modelo, el objetivo es entender por qué se realizan ciertas predicciones. Para lograr esto, LIME simula cómo varían las predicciones al proporcionar diferentes variantes de los datos de entrada, generando un nuevo conjunto de datos utilizando muestras permutadas junto con las predicciones correspondientes. Con este nuevo conjunto de datos, LIME luego entrena un modelo interpretable, que se ajusta a las instancias cercanas a aquella que nos interesa explicar. El modelo aprendido debería ser capaz de ofrecer una buena aproximación de las predicciones del modelo inicial de "*caja negra*" a nivel local.

Desde una perspectiva matemática, los modelos sustitutos locales, con la restricción de ser interpretables, pueden expresarse así:

$$\text{explicación}(x) = \arg\min_{f \in F}(f, g, C_x) + \Omega(f)$$

Aquí, el modelo de explicación para la observación x es representado por f, el cual podríamos tomar como un modelo de regresión lineal que minimiza la pérdida L, por ejemplo, tomando como L el error cuadrático medio. Esta pérdida mide qué tan cerca está la explicación de la predicción del modelo original g, mientras se mantiene baja la complejidad del modelo $\Omega(f)$. F representa la familia de posibles explicaciones, como todos los modelos de regresión lineal posibles. La medida de proximidad C_x define qué tan grande es el vecindario alrededor de la instancia x que consideramos para la explicación. En la práctica, LIME generalmente solo optimiza la parte de la pérdida, dejando al usuario que determine la complejidad al seleccionar el número máximo de características que puede usar el modelo de regresión lineal.

Los pasos a seguir para entrenar modelos locales sustitutos son:

- Selecciona la instancia de interés para la que se desea tener una explicación de su predicción dentro del modelo de "*caja negra*".
- Perturba el conjunto de datos inicial y obtener las predicciones del modelo para estos nuevos puntos.
- Evalúa las nuevas muestras en función de su cercanía a la instancia de interés.
- Entrena un modelo interpretable con este nuevo conjunto de datos, considerando la proximidad a las instancias de interés.
- Interpreta la predicción a través del modelo localmente ajustado.

A la hora de seleccionar las características deseadas para nuestro modelo interpretable, podemos optar por utilizar un modelo Lasso con un parámetro de regularización λ, que controla cuándo se alcanzan las k características deseadas. Otras alternativas para esto serían el uso de criterios de selección de características hacia adelante o hacia atrás.

El modelo local e interpretable para la explicación es independiente del modelo de aprendizaje automático inicial que elijamos para el entrenamiento de nuestros datos, por lo que se consigue libertad a la hora de elegir el modelo predictivo y el modelo interpretable. Además, también permite tener cierta flexibilidad en la elección de las características interpretables que el modelo original puede no haber tenido en cuenta. LIME es uno de los pocos métodos que es capaz de adaptarse a múltiples escenarios, siendo funcional para datos tabulares, texto e imágenes. Otro punto a favor es su sencilla implementación.

Por el contrario, a la hora de trabajar con LIME se deben probar diferentes configuraciones de kernel para lograr una definición correcta del vecindario. Este es su principal problema para buscar explicaciones con sentido. El muestreo de los nuevos puntos de datos podría mejorarse, pues se ignora la posible correlación entre las características que puede derivar en puntos de datos poco probables. Aunque, los modelos sustitutos locales, con LIME como una implementación concreta, son prometedores, aún necesitan seguir desarrollándose, solucionando ciertos problemas de inestabilidad en las explicaciones.

El uso de LIME como método complementario a Shapley es altamente recomendable, ya que nos permite añadir más herramientas para la interpretabilidad de los modelos de aprendizaje automático.

Comparado con el uso de los valores de Shapley, que nos permite obtener una medida global o individual de la importancia de cada característica para la predicción, LIME se enfoca en generar explicaciones locales al aproximar el comportamiento del modelo en un vecindario específico. Mientras que los valores de Shapley explican la diferencia entre la predicción y la predicción promedio global, LIME explica la diferencia entre la predicción y una predicción promedio local. Esto hace que las interpretaciones realizadas con Shapley se consideren más robustas que las realizadas con LIME. Además, la solidez teórica proporcionada por la base de teoría de juegos a los valores de Shapley permite su generalización a una

mayor variedad de modelos. Por otro lado, LIME destaca por su rapidez y bajo coste computacional en comparación con Shapley. Dependiendo del tipo de explicaciones que busquemos, podemos destacar el uso de LIME para obtener respuestas breves y más accesibles. Sin embargo, si se requiere una explicación exhaustiva y detallada, los valores de Shapley claramente superan a LIME en términos de precisión y comprensión del modelo.

2.3.2. Métodos Específicos que Aprovechan el Uso del Valor de Shapley

En el contexto actual de la interpretación de modelos de aprendizaje automático, los enfoques basados en Shapley son muy populares por su capacidad para proporcionar explicaciones precisas y transparentes sobre las decisiones de los modelos. Para comprender mejor el potencial desarrollo que puede lograrse con Shapley y su relevancia en el campo del aprendizaje automático, exploraremos dos metodologías específicas. Eject, que se centra en la interpretación de modelos basados en árboles, y SVERAD, diseñado para el análisis de modelos de máquinas de vectores de soporte.

Eject

El método **Eject** [7] se propone como una nueva forma de calcular la utilidad en árboles de decisión. A diferencia de TreeSHAP, que considera todas las partes del árbol, Eject abandona el árbol temprano cuando se encuentra una característica ausente en la coalición, asegurando que las características fuera del camino de decisión sean jugadores nulos, lo que reduce la complejidad computacional.

Eject cumple con la propiedad del jugador ficticio local, asegurando que si una característica no se utiliza en la predicción de una instancia, su atribución sea cero. Así, logra que las explicaciones sean consistentes con la lógica del árbol de decisión. En términos de eficiencia computacional, Eject es más eficiente porque la suma exponencial sobre todos los subconjuntos de características se reduce a una suma sobre los subconjuntos de características únicas en el camino de decisión, lo cual es beneficioso en árboles de decisión potenciados. Al igual que TreeShap, Eject también se puede aprovechar de empujar todos los posibles subconjuntos S por el árbol simultáneamente.

En resumen, Eject se presenta como una alternativa eficiente y coherente para calcular la utilidad en árboles de decisión, asegurando explicaciones localmente precisas y computacionalmente eficientes. Sin embargo, TreeSHAP puede proporcionar explicaciones más detalladas al asignar atribuciones a características que podrían influir en la predicción, incluso si no se utilizan directamente en el camino de decisión.

SVERAD

La metodología **SVERAD** *(Shapley Values for Efficiently Rationalizing Activity Decisions)* [4] se enfoca en calcular valores de Shapley precisos para las predicciones de **Máquinas de Vectores de Soporte (SVM)** utilizando el kernel de función de base radial (RBF). Estos valores de Shapley ofrecen una comprensión detallada de cómo cada característica contribuye a las predicciones del modelo SVM.

Para realizar los cálculos, se evalúa el cambio en el valor del kernel RBF al agregar o quitar características del conjunto. SVERAD simplifica este proceso y reduce la complejidad computacional al considerar las características de intersección y diferencia simétrica por separado. Además, se desarrollan fórmulas matemáticas específicas para calcular los valores de Shapley para cada tipo de característica, lo que permite una interpretación precisa y eficiente de las contribuciones.

SVERAD se valida mediante pruebas de concepto y comparaciones con métodos de aproximación de Shapley. Los resultados demuestran que SVERAD produce valores de Shapley más precisos y una mejor correlación con los resultados reales de SVM.

En resumen, SVERAD puede verse como una herramienta eficaz y precisa para calcular los valores de Shapley en las predicciones de SVM. Es especialmente útil para interpretar y comprender las decisiones de los modelos de SVM, los cuales, debido a la naturaleza del algoritmo de optimización que utilizan para encontrar el hiperplano óptimo, pueden considerarse complejos en su funcionamiento interno.

Capítulo 3

Aplicación del Algoritmo de Shapley en el Análisis de la Liga Española de Fútbol

El propósito de este último capítulo es reforzar el trabajo que se ha ido desarrollando. Para ello, usaremos un ejemplo con el que facilitaremos y asentaremos la comprensión de los conceptos expuestos anteriormente. En nuestro ejemplo, utilizaremos datos de la liga española de fútbol, con los que vamos a realizar una aplicación práctica y contextualizada del uso de Shapley. Hemos decidido decantarnos por el uso de SHAP porque nos permitirá analizar cómo cada característica afecta nuestras predicciones de manera individual y global, mejorando la interpretabilidad y destacando la importancia de los datos para crear modelos predictivos efectivos.

3.1. Introducción

La inteligencia artificial y más en específico el aprendizaje automático, están ganando rápidamente popularidad como métodos para predecir el rendimiento a nivel deportivo. Estas tecnologías no han pasado desapercibidas en uno de los deportes más populares que existe, el fútbol. Hoy en día, se recopilan grandes cantidades de datos técnicos durante los partidos, lo que ha provocado que los equipos profesionales se hayan visto en la necesidad de contar con expertos en ciencia de datos. Esto se debe a que los datos recopilados proporcionan una valiosa información acerca del estilo de juego y las tácticas de un equipo. Por lo que, al analizar los datos, los entrenadores y analistas pueden adquirir cierta comprensión de las fortalezas y debilidades de un jugador o un equipo, utilizando esta información para tomar decisiones mejor fundamentadas sobre la contratación de jugadores o el planteamiento de un partido.

El desafío que ha surgido es poder aplicar el uso de la IA para predecir el rendimiento de equipos. Esta área de investigación en crecimiento busca llevar a otro nivel el análisis moderno de partidos de fútbol. Con las grandes cantidades de datos que se generan sobre el rendimiento de equipos y jugadores, se entrenan múltiples algoritmos con diferentes enfoques para hacer predicciones. Sin embargo, factores como la naturaleza de las características utilizadas, la complejidad del problema predictivo y el tamaño del conjunto de datos, pueden influir significativamente en la precisión de la predicción.

Para poder realizar un análisis eficaz del comportamiento de un equipo, es fundamental resumir su estilo de juego de una manera compresible para los humanos, sin dejar de lado la relevancia del propio análisis de datos. Uno de los principales retos al utilizar la IA en el análisis del rendimiento deportivo es la falta de transparencia e interpretabilidad de los resultados. El objetivo de este ejemplo será aplicar uno de los métodos que hemos desarrollado anteriormente, SHAP, visto en la sección 2.1.3. Nos vamos a enfocar en analizar el rendimiento de los equipos, centrándonos en la diferencia de goles en un partido o en el resultado del mismo, con el fin de proporcionar una explicación de las características más relevantes.

3.2. Metodología

Como hemos mencionado antes, nuestro enfoque se centra en analizar el rendimiento de los equipos de fútbol, específicamente en la diferencia de goles o el resultado de los partidos. Para lograrlo, seguiremos el siguiente procedimiento. Primero trataremos de obtener un conjunto de datos adecuado para nuestro objetivo. Luego, ajustaremos un modelo de regresión lineal o entrenaremos el modelo de clasificación. Finalmente, aplicaremos SHAP para analizar las predicciones y la contribución de cada característica.

Procedemos con la descripción de nuestro conjunto de datos. Hemos recopilado todos los partidos de las temporadas 2022-2023 y 2023-2024 de la primera división española, considerando únicamente los equipos que estuvieron en dicha categoría ambos años, según el procedimiento descrito en (II.1). Para cada equipo, se tomarán en cuenta todos los partidos excepto aquellos en los que se enfrentaron a equipos que no estuvieron ambos años en la liga. Al final, contamos con un total de 1088 partidos para nuestro análisis. Este conjunto de datos está compuesto por 44 variables que abarcan información ofensiva, defensiva y de control del juego, incluyendo algunas variables categóricas. Los datos han sido obtenidos de (`https://fbref.com/es/comps/12/Estadisticas-de-La-Liga`).

Dado que nuestras variables fueron seleccionadas manualmente al crear el conjunto de datos, los modelos predictivos que hemos desarrollado han utilizado todas las variables posibles dentro de las posibilidades, tanto en el modelo de regresión como en el de clasificación (ver II.3). Aunque no hayamos empleado ninguna técnica de selección de características, en (II.2) describimos los posibles procedimientos a seguir. Además, se observa el potencial de predicción de nuestro conjunto de datos, que mostró un $R^2 > 0{,}9$.

Para cuantificar los efectos de cada característica en el resultado final de nuestro estudio, elegimos SHAP como herramienta de explicabilidad post-hoc. El uso del método SHAP proporciona una explicación detallada, precisa y fácilmente interpretable del funcionamiento interno de nuestro modelo. Los valores SHAP asignan una contribución específica a cada característica para una predicción específica. Los valores positivos indican que la característica aumentó la predicción, mientras que los valores negativos indican que la característica disminuyó la predicción. Esto nos ayuda a comprender cómo cada característica afecta la predicción final y cómo se comparan entre sí.

3.3. Análisis de Resultados

A continuación, vamos a presentar los resultados obtenidos mediante nuestro ejemplo, incluyendo las explicaciones generadas por el algoritmo SHAP. Antes de realizar un análisis detallado de un equipo, primero exploraremos la visión global del modelo.

La Figura 3.1 muestra un gráfico que compara los valores reales (eje x) con los valores predichos (eje y). En este gráfico, distinguiremos dos tipos de puntos. Los puntos verdes representan la comparación para los datos promedio de las diferentes variables para cada uno de los 17 equipos, incluyendo el resultado del rendimiento del equipo. Por ejemplo, un rendimiento promedio del equipo de $+1{,}3$ indica que, en promedio, el equipo anotó 1,3 goles más de los que recibió. Los puntos azules, por otro lado, muestran la comparación para partidos aleatorios en el conjunto de datos de prueba del modelo, para hacernos una mejor idea del rendimiento del modelo. Hemos incluido una línea de mejor ajuste en el gráfico para visualizar qué tan cerca están las predicciones de los resultados reales.

Para entender los factores que influyen en las predicciones del modelo, visualizaremos la Figura 3.2. Las características se presentan según su importancia, aunque no distingue entre si tiene efecto positivo o negativo en la predicción.

Nuestro análisis revela el valor de variables relacionadas con el pase, destacando el uso y efectividad del pase corto y medio frente al largo. Asimismo, variables como el número de acciones de goles creadas, los Expected Goals (xG), la cantidad de tiros a puerta y los goles por disparo tienen un gran peso en el aspecto ofensivo. Por el contrario, las variables defensivas que destacan son los disparos recibidos a puerta y las paradas.

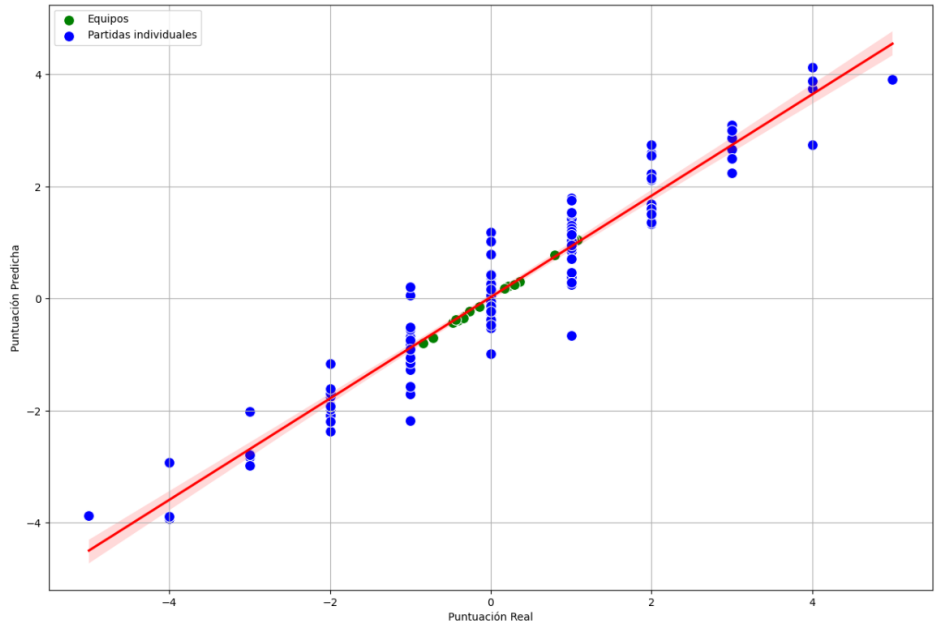


Figura 3.1: Comparación entre Valores Reales y Predichos de la Diferencia de Goles

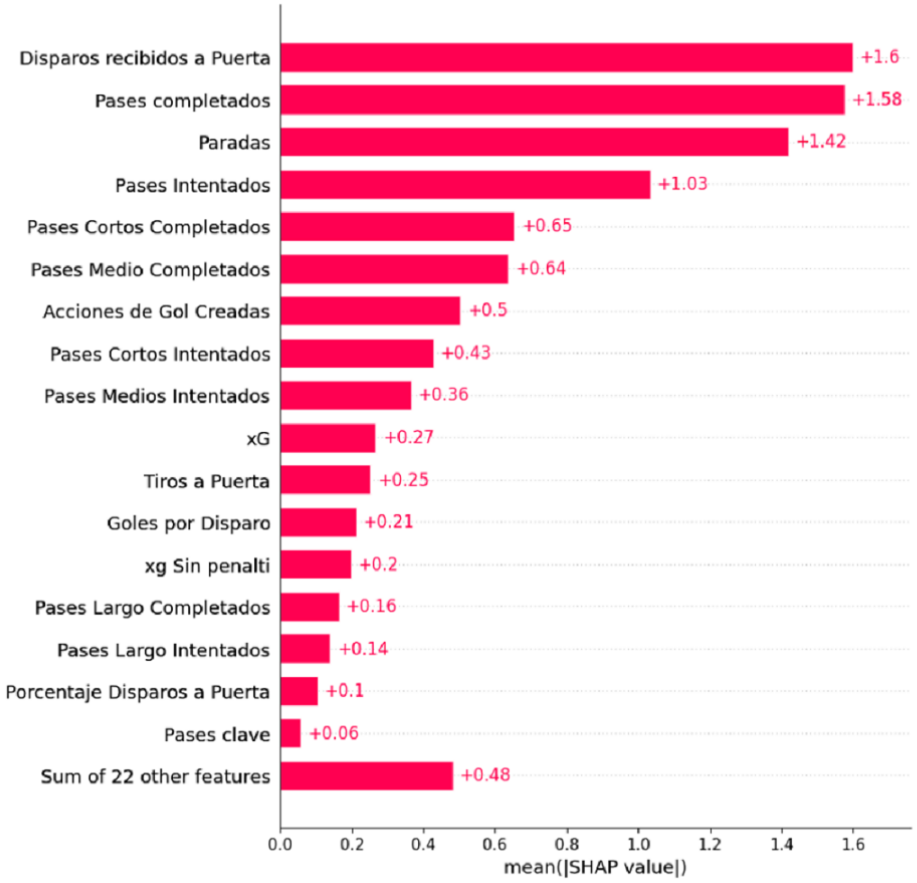


Figura 3.2: Resumen Global de la Importancia de Variables según SHAP

El siguiente paso consiste en el análisis del rendimiento local. Las siguientes gráficas Figuras 3.3, 3.4, 3.5 y 3.6 nos van a permitir ver cómo contribuyeron las diferentes variables a la predicción del modelo para equipos específicos. Para nuestro ejemplo, queremos estudiar equipos en diferentes situaciones. Los seleccionados son: el Almería, como un equipo de tabla baja; el Real Betis, representante de un equipo equilibrado; y el Real Madrid, como un equipo de tabla alta. Para este análisis, incluiremos la gráfica de SHAP para el modelo de regresión que predice la diferencia de goles basándonos en los datos promediados de los equipos. Además, en el caso del Real Madrid, también añadiremos un gráfico del modelo de clasificación de resultados.

En el caso del Almería (Figura 3.3), al tratarse de un equipo que se encuentra entre las últimas posiciones de la liga, la predicción de la diferencia de goles es negativa, como cabría esperar. Esto también queda claramente representado al examinar la importancia de las variables, ya que nos damos cuenta de la ausencia de variables puramente ofensivas. En su lugar, destacan los disparos recibidos a puerta con una connotación negativa, y las paradas con un efecto positivo. Además, al analizar el peso de las variables de pase, observamos un posible efecto negativo, posiblemente debido al bajo acierto en los pases largos.

Para el caso del Real Betis (Figura 3.4), la predicción en la diferencia de goles es casi cero, dejando clara la nivelación de los resultados. En este equipo, se observa la presencia de múltiples ámbitos de variables, entre las que destacan positivamente el número de disparos recibidos y el pase medio, mientras que el pase corto y las paradas tienen un impacto negativo.

Con respecto al caso del Real Madrid (Figura 3.5), la predicción que tenemos es ligeramente superior a uno, lo que justifica que sea uno de los principales candidatos a ganar la liga. En cuanto al análisis de sus características, es cierto que aparecen bastantes variables relacionadas con el pase, pero su efecto general es mínimo. La mayor parte de la carga explicativa recae en las acciones de gol creadas, el xG y la comparación de los efectos de los disparos recibidos a puerta y las paradas

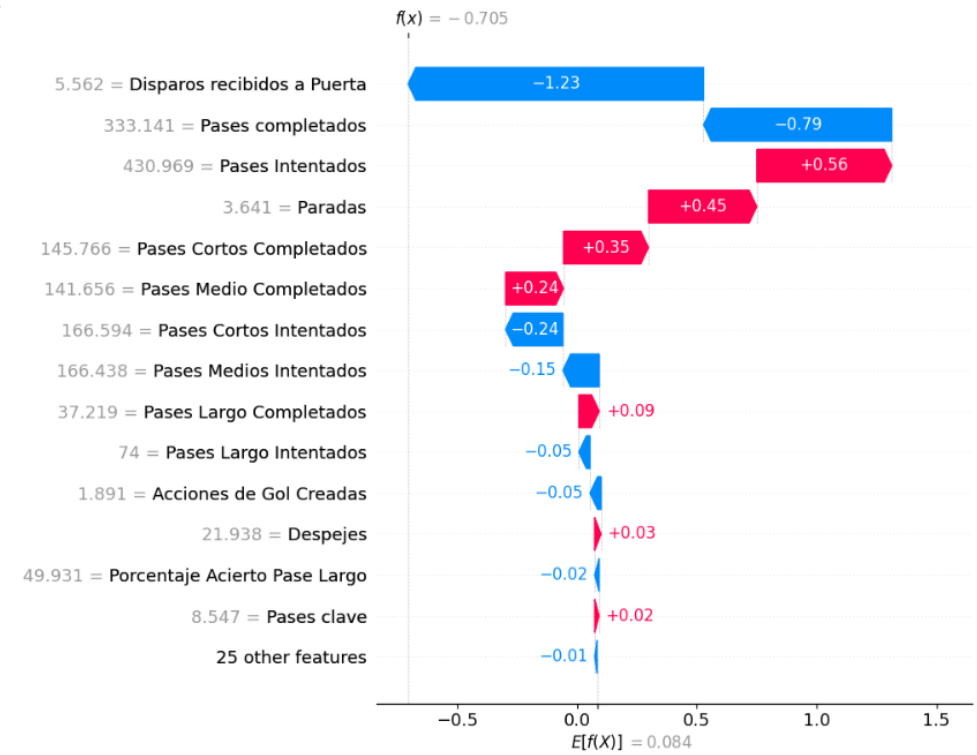


Figura 3.3: Gráfico de SHAP que indica las variables clave de rendimiento del Almería

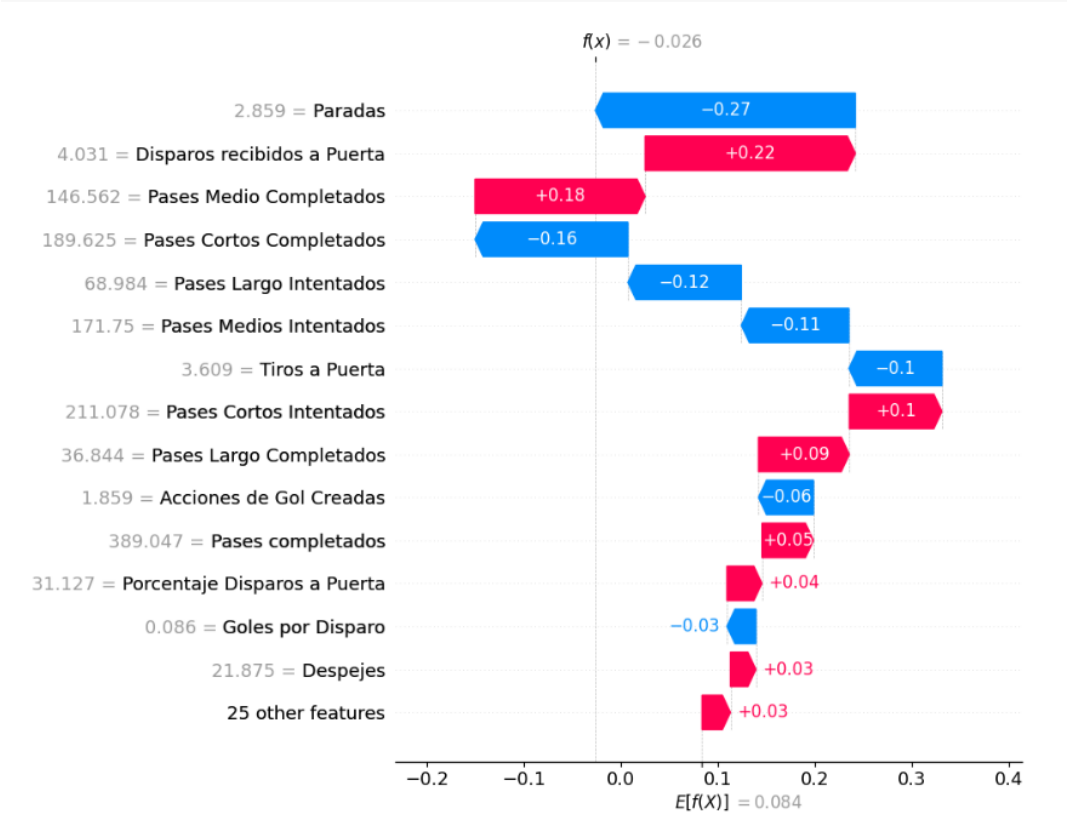


Figura 3.4: Gráfico de SHAP que indica las variables clave de rendimiento del Real Betis

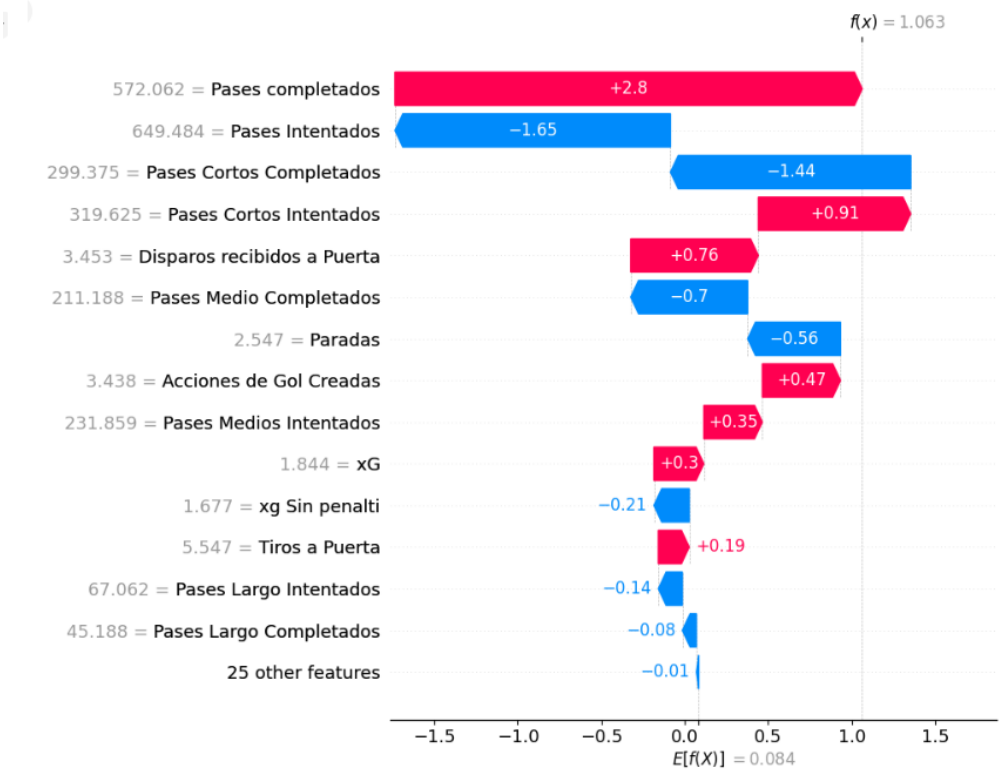


Figura 3.5: Gráfico de SHAP que indica las variables clave de rendimiento del Real Madrid

La siguiente gráfica pretende mostrar los resultados que podemos conseguir a través de un modelo de clasificación, en este caso para determinar si un partido se ha ganado, empatado o perdido. En la Figura 3.6 se puede comparar la importancia de las variables en una victoria del Real Madrid, que suele ser lo habitual, frente a cómo se distribuyen estas importancias en los casos contrarios. La lista que se encuentra en II.3 muestra el nombre de las características.

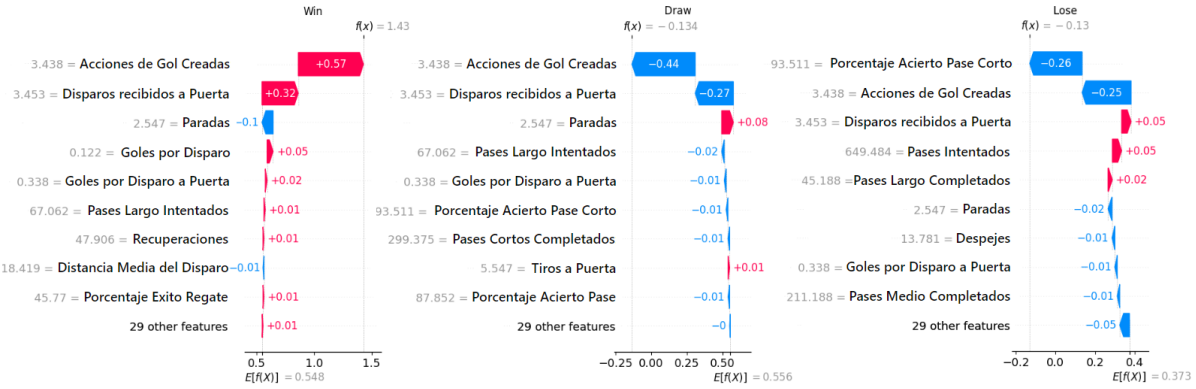


Figura 3.6: Gráfico de SHAP sobre la importancia de las variables en el modelo de clasificación para el equipo del Real Madrid

Conclusiones finales

Durante el ejemplo, hemos desarrollado una visión valiosa de cómo los valores de Shapley aportan herramientas sólidas para la interpretación de modelos, en nuestro contexto para el fútbol. El enfoque que hemos desarrollado no tendría un buen rendimiento en el análisis individual de un solo partido, pero es útil para comprender cómo diversas características influyen en el desempeño promedio de los equipos a lo largo de una temporada. Aunque en el modelo se incluyen muchas características relacionadas, el valor que se les asigna y las comparaciones que se pueden hacer entre ellas nos resultan convenientes para comprender mejor el rendimiento del equipo. Es importante destacar que este análisis se basa en datos anteriores y no predice el rendimiento futuro. No obstante, muestra el gran potencial de esta metodología si mejoramos el conjunto de datos añadiendo más variables y observaciones, y desarrollamos modelos más complejos.

En resumen, los enfoques que utilizan los valores de Shapley y la inteligencia artificial explicada están en constante evolución. Con el tiempo, los avances que se pueden alcanzar en la interpretación para diferentes modelos y usos, pueden llegar a ser sorprendentes.

Bibliografía

[1] AAS, K.; JULLUM, M. Y LØLAND, A., *Explaining Individual Predictions When Features Are Dependent: More Accurate Approximations to Shapley Values*, 2019, https://arxiv.org/pdf/1903.10464

[2] ALI, S.; ABUHMED, T.; EL-SAPPAGH, S.; MUHAMMAD, K.; ALONSO, J.; CONFALONIERI, R.; GUIDOTTI, R.; DEL SER, J.; DÍAZ-RODRÍGUEZ, N. Y HERRERA, F., *Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence*, Information Fusion, 2023, https://www.researchgate.net/publication/370111593_Explainable_Artificial_Intelligence_XAI_What_we_know_and_what_is_left_to_attain_Trustworthy_Artificial_Intelligence

[3] AZZALINI, A. Y SCARPA, B., *Data Analisis and Data Mining*, Oxford University Press, 2012, https://dbo.raharja.ac.id/331/1/Data%20Analysis%20and%20Data%20Mining.pdf

[4] BAJORATH, J.; FELDMANN, C. Y MASTROPIETRO, A., *Calculation of exact Shapley values for explaining support vector machine models using the radial basis function kernel*, Scientific Reports, 2023, https://www-nature-com.cuarzo.unizar.es:9443/articles/s41598-023-46930-2

[5] BLACK, J. E.; KUEPER, J. K. Y WILLIAMSON, T. S., *An introduction to machine learning for classification and prediction*, *Family Practice*, Volumen 40, Febrero 2023, pp. 200–204, https://doi.org/10.1093/fampra/cmac104

[6] BURZYKOWSKI, T.; ROUSSEAU, A. J.; GEUBBELMANS, M. Y VALKENBORG, D., *Introduction to machine learning*, American Journal of Orthodontics and Dentofacial Orthopedics, Volume 163, 2023, pp. 732-734, https://www-sciencedirect-com.cuarzo.unizar.es:9443/science/article/pii/S0889540623000793#cebib0010

[7] CAMPBELL, T. W.; GEORGANTAS III, R. W.; RODER, H. Y RODER, J., *Exact Shapley values for local and model-true explanations of decision tree ensembles, Machine Learning with Applications*, Volumen 9, 2022, https://www-sciencedirect-com.cuarzo.unizar.es:9443/science/article/pii/S2666827022000500

[8] COMISIÓN EUROPEA, *Proyecto de ley de inteligencia artificial*, Legislación de la UE en progreso, 2024, https://www.europarl.europa.eu/RegData/etudes/BRIE/2021/698792/EPRS_BRI(2021)698792_EN.pdf Consultado a fecha de 07/06/2024

[9] COVERT, I.; JETHANI, N.; LEE, S.-I.; RANGANATH, R. Y SUDARSHAN, M., *FastSHAP: Real-Time Shapley Value Estimation*, 2021, https://arxiv.org/pdf/2107.07436

[10] DEVOS, M. Y KENT, D. A., *Game Theory A Playful Introduction*, American Mathematical Society, Volumen 80, 2016, https://bookstore.ams.org/stml-80

[11] GOLD, R. Z.; LINDEMAN, R. H. Y MERENDA, P. F., *Introduction to bivariate and multivariate analysis*, Scott Foresman and Company, 1980, https://search.worldcat.org/es/title/Introduction-to-bivariate-and-multivariate-analysis/oclc/5310754

[12] HASTIE, T.; TIBSHIRANI, R. Y FRIEDMAN, J., *The Elements of Statistical Learning*, Data Mining, Inference, and Prediction, Segunda edición, 2009, https://hastie.su.domains/Papers/ESLII.pdf

[13] KOKKOTIS, C.; MOUSTAKIDIS, S.; PLAKIAS, S.; TSATALAS, T. Y TSAOPOULOS, D., *Predicting Football Team Performance with Explainable AI: Leveraging SHAP to Identify Key Team-Level Performance Metrics*, Future Internet, 2023, https://www.researchgate.net/publication/370527093_Predicting_Football_Team_Performance_with_Explainable_AI_Leveraging_SHAP_to_Identify_Key_Team-Level_Performance_Metrics

[14] MOLNAR, C. *Interpretable machine learning. A Guide for Making Black Box Models Explainable*, 2019, https://christophm.github.io/interpretable-ml-book/

[15] MOLNAR, C., *Interpreting Machine Learning Models With SHAP*, A Guide With Python Examples And Theory On Shapley Values, Primera Edición, 2023, https://christophmolnar.com/books/shap/

[16] OWEN, A. B., *Sobol' indices and Shapley value*, Stanford University, Journal on Uncertainty Quantification, pp. 245–251, 2014, https://artowen.su.domains/reports/sobolshapley.pdf

[17] OWEN, A. B. Y PRIEUR, C., *On Shapley value for measuring importance of dependent inputs*, 2016, https://arxiv.org/pdf/1610.02080v3

[18] PEÑA, D., *Regresión y diseño de experimentos*, Alianza Editorial, Febrero 2010, https://www.alianzaeditorial.es/libro/manuales/regresion-y-diseno-de-experimentos-daniel-pena-9788420693897/

[19] PÉREZ, J.; JIMENO, J. L. Y CERDÁ, E., *Teoría de Juegos*, Pearson-Prentice Hall, Madrid, 2004, https://elvisjgblog.wordpress.com/wp-content/uploads/2018/02/teorc3ada-de-juegos-joaquc3adn-pc3a9rez-2004.pdf

[20] SHAPLEY, L. S., *A Value for n-Person Games*, Contributions to the Theory of Games II, Princeton University Press, 1953, https://www.rand.org/content/dam/rand/pubs/papers/2021/P295.pdf

Anexos

Apéndice I

Demostraciones de teoremas pendientes

I.1. Prueba Teorema 2.1

Teorema 2.1. *Sea la descomposición ANOVA de una función con k entradas independientes y componentes de varianza σ_S^2 para $S \subseteq K$. Si el valor de un subconjunto S de variables es $val(S) = \underline{\tau}_S^2$, entonces el valor de Shapley de la variable j es:*

$$\phi_j = \sum_{S \subseteq K,\ j \in S} \frac{\sigma_S^2}{|S|}$$

Demostración:

Usando la linealidad del valor de Shapley, podemos descomponer la función de valor en componentes más simples:

$$\text{val}(S) = \sum_{s \subseteq K, s \neq \emptyset} \text{val}^{(s)}(S)$$

Aquí, $\text{val}^{(s)}(S)$ representa la contribución de la coalición s al valor de S. La función $\sigma_s 1_{S=s}$ indica que esta contribución es σ_s si S es igual a s, y 0 en caso contrario.

Según la propiedad de pasividad, si un jugador j no pertenece a la coalición s, su valor de Shapley es cero, y debido a la simetría, si un jugador j pertenece a la coalición s, su valor de Shapley es $\phi_j^{(s)} = \sigma_s^2/|s|$.

Además, la suma de los valores de Shapley de todos los jugadores en la coalición s debe ser igual al valor total de la coalición s:

$$\sum_{j \in s} \phi_j^{(s)} = \text{val}^{(s)} \quad \text{(eficiencia)}$$

La conclusión se sigue entonces de la aditividad del valor de Shapley, que permite agregar los valores de Shapley de todas las subcoaliciones s para obtener el valor total de una coalición S.

Las expresiones de los índices de Sobol para conjuntos de un único elemento son:

$$\underline{\tau}_{\{j\}}^2 = \sigma_{\{j\}}^2 \quad \text{y} \quad \bar{\tau}_{\{j\}}^2 = \sum_{s: j \in s} \sigma_s^2$$

Ninguno de los índices de Sobol coinciden con el valor de Shapley porque no suman a $\underline{\tau}_K^2 = \sigma^2$, y por lo tanto, no cumplen con la primera propiedad de eficiencia. A continuación, mostraremos que este problema no puede solucionarse simplemente buscando realizar un reescalado.

El índice $\underline{\tau}_S^2$ no considera la contribución de la variable j a los componentes de varianza σ_S^2 cuando $j \in S$ y $|S| > 1$. Por otro lado, el índice $\bar{\tau}_S^2$ cuenta múltiples veces los componentes de varianza: la contribución de σ_S^2 para $|S| > 1$ se incluye en $\bar{\tau}_{\{j\}}^2$ para cada $j \in S$.

Ninguno de estos problemas puede resolverse reescalando, se necesitaría un reescalado diferente para cada caso. En resumen, los índices de Sobol solo acotan el valor de Shapley, lo que significa que el

valor de Shapley proporciona una medida más adecuada y consistente en comparación con los índices de Sobol.

$$\underline{\tau}^2_{\{j\}} \leq \phi_j \leq \bar{\tau}^2_{\{j\}}$$

Esta propiedad de encasillado se cumple porque cada $\sigma^2_S \geq 0$. Estos componentes de varianza pueden expresarse en términos de $\underline{\tau}^2_S$ mediante un proceso llamado inversión de Mobius. La inversión de Mobius es una técnica matemática que permite relacionar funciones aritméticas en teoría de números. En este contexto, se utiliza para expresar los componentes de varianza en función de los índices de Sobol mediante una relación de inclusión-exclusión.

$$\sigma^2_S = \sum_{s \subseteq S} (-1)^{|S-s|} \underline{\tau}^2_s$$

En contextos económicos, σ^2_S es similar a β_S, que se define como:

$$\beta_S = \sum_{s \subseteq S} (-1)^{|S-s|} val(s)$$

Esta medida puede ser negativa, lo que indica que la inclusión de ciertos jugadores puede disminuir el rendimiento del equipo. Por ejemplo, podríamos tener el caso en el que la contribución combinada de dos jugadores es menor que la suma de sus contribuciones individuales. Un juego en el que todas las β_S sean mayores o iguales a cero se considera un juego totalmente monótono. Esta propiedad se aplica a una amplia gama de juegos, incluidas las funciones de creencia en teoría de la decisión.

I.2. Prueba Teorema 2.2

Teorema 2.2. *Si $f(x) = \beta_0 + \beta^\top x$ para $x \sim N(\mu, \Sigma)$ donde $\Sigma \in \mathbb{R}^{k\times k}$ es una matriz simétrica semidefinida positiva de rango completo, entonces el efecto de Shapley para la variable j es:*

$$\phi_j = \frac{1}{k} \sum_{S \subseteq -\{j\}} \binom{k-1}{|S|}^{-1} \frac{\mathrm{Cov}(x_j, x_{-S}^\top \beta_{-S} \mid x_S)^2}{\mathrm{Var}(x_j \mid x_S)}$$

Demostración:

Empecemos expresando $\mathrm{var}(x_{-S}|x_S)$ de la siguiente manera:

$$\mathrm{var}(x_{-S}|x_S) = \Sigma_{-S,-S} \; - \; \Sigma_{-S,S}\, \Sigma_{S,S}^{-1}\, \Sigma_{S,-S} \tag{1}$$

Mediante el uso de (1) para $\mathrm{var}(f(x)|x_S)$ obtenemos:

$$\mathrm{var}(f(x)|x_S) = \mathrm{var}(x_S^\top \; \beta_S \; + \; x_{-S}^\top \, \beta_{-S} \mid x_S) = \mathrm{var}(x_{-S}^\top \, \beta_{-S} \mid x_S)$$

$$= \beta_{-S}^\top \left(\Sigma_{-S,-S} \; - \; \Sigma_{-S,S}\, \Sigma_{S,S}^{-1}\, \Sigma_{S,-S} \right) \beta_{-S} \tag{2}$$

Usaremos $s = s(j,S) = -S - \{j\}$ que nos ayudará a visualizar la matriz de covarianza particionada si los índices han sido ordenados para que los S precedan a los j, y estos a su vez precedan a los de s.

$$\Sigma = \begin{pmatrix} \Sigma_{SS} & \Sigma_{Sj} & \Sigma_{Ss} \\ \Sigma_{jS} & \Sigma_{jj} & \Sigma_{js} \\ \Sigma_{sS} & \Sigma_{sj} & \Sigma_{ss} \end{pmatrix}$$

Solo para esta sección, hacemos una compresión notacional adicional abreviando $S + \{j\}$ a $S + j$. Prosigamos con el uso de la expresión (2):

$$\underline{\tau}_{S+j}^2 - \underline{\tau}_S^2 = \mathrm{var}(f(x)|x_S) - \mathrm{var}(f(x)|x_{S+j})$$

$$= \beta_{-S}^\top \left(\Sigma_{-S,-S} \; - \; \Sigma_{-S,S}\, \Sigma_{S,S}^{-1}\, \Sigma_{S,-S} \right) \beta_{-S} \; - \; \beta_{-s}^\top \left(\Sigma_{ss} \; - \; \Sigma_{s,S+j}\, \Sigma_{S+j,S+j}^{-1}\, \Sigma_{S+j,s} \right) \beta_s \tag{3}$$

A continuación, recordemos la fórmula de la inversa de una matriz particionada:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BECA^{-1} & -A^{-1}BE \\ -ECA^{-1} & E \end{pmatrix} \quad \text{con} \quad E = (D - CA^{-1}B)^{-1}$$

Para nuestro caso, tenemos lo siguiente:

$$\Sigma_{S+j,S+j}^{-1} = \begin{pmatrix} \Sigma_{SS} & \Sigma_{Sj} \\ \Sigma_{jS} & \Sigma_{jj} \end{pmatrix}^{-1} = \begin{pmatrix} \Sigma_{SS}^{-1} + \Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1} & -\Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S) \\ -E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1} & E_j(S) \end{pmatrix} \tag{4}$$

Siendo $E_j(S) = (\Sigma_{jj} - \Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Sj})^{-1} = \mathrm{var}(x_j|x_S)^{-1}$ que existe porque Σ tiene rango completo. Ahora con (4):

$$\Sigma_{s,S+j}\Sigma_{S+j,S+j}^{-1}\Sigma_{S+j,s} = \begin{pmatrix} \Sigma_{sS} & \Sigma_{sj} \end{pmatrix} \begin{pmatrix} \Sigma_{SS}^{-1} + \Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1} & -\Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S) \\ -E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1} & E_j(S) \end{pmatrix} \begin{pmatrix} \Sigma_{Sj} \\ \Sigma_{js} \end{pmatrix}$$

$$= \begin{pmatrix} \Sigma_{sS} & \Sigma_{sj} \end{pmatrix} \begin{pmatrix} \Sigma_{SS}^{-1}\Sigma_{Ss} + \Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} - \Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S)\Sigma_{js} \\ -E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} + E_j(S)\Sigma_{js} \end{pmatrix}$$

$$= \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Ss} + \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} - \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj}E_j(S)\Sigma_{js} - \Sigma_{sj}E_j(S)\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} + \Sigma_{sj}E_j(S)\Sigma_{js}$$

$$= \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Ss} + E_j(S)\left(\Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj}\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} - \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj}\Sigma_{js} - \Sigma_{sj}\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} + \Sigma_{sj}\Sigma_{js} \right)$$

$$= \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Ss} + E_j(S)\left(\Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj}\left(\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} - \Sigma_{js}\right) - \Sigma_{sj}\left(\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} - \Sigma_{js}\right)\right)$$

$$= \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Ss} + E_j(S)\left(\Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj} - \Sigma_{sj}\right)\left(\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss} - \Sigma_{js}\right)$$

$$= \Sigma_{sS}\ \Sigma_{SS}^{-1}\ \Sigma_{Ss}\ +\ E_j(S)\ \mathrm{cov}(x_s, x_j|x_S)\ \mathrm{cov}(x_j, x_s|x_S) \tag{5}$$

Para llegar a (5) hemos usado que $\mathrm{cov}(x_A, x_B|x_C) = \Sigma_{A,B}\ -\ \Sigma_{A,C}\ \Sigma_{C,C}^{-1}\ \Sigma_{C,B}$.
Partiendo del resultado conseguido para $(\underline{\tau}_{S+j}^2 - \underline{\tau}_S^2)$ en (3) y usando la expresión (5) logramos:

$$\underline{\tau}_{S+j}^2 - \underline{\tau}_S^2$$

$$= \beta_{-S}^\top\ \mathrm{cov}(x_{-S}|x_S)\ \beta_{-S}\ -\ \beta_s^\top\ \Sigma_{ss}\ \beta_s\ +\ \beta_s^\top\left(\Sigma_{sS}\ \Sigma_{SS}^{-1}\ \Sigma_{Ss}\ +\ E_j(S)\ \mathrm{cov}(x_s, x_j|x_S)\ \mathrm{cov}(x_j, x_s|x_S)\right)\beta_s$$

$$= \beta_{-S}^\top\ \mathrm{cov}(x_{-S}|x_S)\ \beta_{-S}\ -\ \beta_s^\top\ \mathrm{cov}(x_s|x_S)\ \beta_s\ +\ E_j(S)\ \beta_s^\top\ \mathrm{cov}(x_s, x_j|x_S)\ \mathrm{cov}(x_j, x_s|x_S)\beta_s$$

$$= \begin{pmatrix}\beta_s & \beta_j\end{pmatrix}\begin{pmatrix}\Sigma ss & \Sigma_{sj}\\ \Sigma js & \Sigma_{jj}\end{pmatrix}\begin{pmatrix}\beta_s\\ \beta_j\end{pmatrix} - \begin{pmatrix}\beta_s & \beta_j\end{pmatrix}\begin{pmatrix}\Sigma_{sS}\\ \Sigma_{jS}\end{pmatrix}\Sigma_{SS}^{-1}\begin{pmatrix}\Sigma_{Ss} & \Sigma_{Sj}\end{pmatrix}\begin{pmatrix}\beta_s\\ \beta_j\end{pmatrix}$$
$$-\beta_s^\top\left(\Sigma_{ss} - \Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Ss}\right)\beta_s + E_j(S)\ \beta_s^\top\ \mathrm{cov}(x_s, x_j|x_S)\ \mathrm{cov}(x_j, x_s|x_S)\beta_s$$

$$= \beta_j^2\Sigma_{jj} + \beta_j\Sigma_{js}\beta_s + \beta_s^\top\Sigma_{sj}\beta_j - \beta_s^\top\Sigma_{sS}\Sigma_{SS}^{-1}\Sigma_{Sj}\beta_j - \beta_j\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Ss}\beta_s\beta_j^2\Sigma_{jS}\Sigma_{SS}^{-1}\Sigma_{Sj}$$
$$+ E_j(S)\ \beta_s^\top\ \mathrm{cov}(x_s, x_j|x_S)\ \mathrm{cov}(x_j, x_s|x_S)\beta_s$$

$$= \beta_j^2\mathrm{var}(x_j|x_S) + 2\beta_j\mathrm{cov}(x_j, x_s|x_S)\beta_s + E_j(S)\beta_s^\top\mathrm{cov}(x_s, x_j|x_S)\mathrm{cov}(x_j, x_s|x_S)\beta_s$$

$$= \beta_j^2\mathrm{var}(x_j|x_S) + 2\beta_j\mathrm{cov}(x_j, x_{-S-j}|x_S)\beta_{-S-j} + \mathrm{var}(x_j|x_S)^{-1}$$
$$\beta_{-S-j}^\top\mathrm{cov}(x_{-S-j}, x_j|x_S)\mathrm{cov}(x_j, x_{-S-j}|x_S)\beta_{-S-j} \tag{6}$$

En estos últimos pasos nos hemos aprovechado de la notación que habíamos introducido antes, $s = s(j,S) = -S - \{j\}$.

A partir de la expresión lograda en (6) y sustituyéndola en (2.7), el valor de Shapley para la variable j se escribe como:

$$\phi_j = \frac{1}{k}\sum_{S\subseteq -\{j\}}\binom{k-1}{|S|}^{-1}\left(\beta_j^2\mathrm{var}(x_j|x_S) + 2\beta_j\mathrm{cov}(x_j, x_{-S-j}|x_S)\beta_{-S-j}\right.$$
$$\left. + \mathrm{var}(x_j|x_S)^{-1}\beta_{-S-j}^\top\mathrm{cov}(x_{-S-j}, x_j|x_S)\mathrm{cov}(x_j, x_{-S-j}|x_S)\beta_{-S-j}\right) \tag{7}$$

Si elevamos al cuadrado la siguiente expresión:

$$\mathrm{cov}(x_j, x_{-S}^\top\beta_{-S}|x_S) = \mathrm{cov}(x_j, x_{-S-j}^\top\beta_{-S-j}|x_S) + \beta_j\mathrm{var}(x_j|x_S) \tag{8}$$

Cuando realicemos los cambios pertinentes en (7) aprovechando lo obtenido en (8), lograremos probar el resultado.

Apéndice II

Estudio práctico detallado y código

II.1. Obtención del Conjunto de Datos sobre La Liga Española

```
# Necesario para trabajar con archivos en Drive
from google.colab import drive
drive.mount('/content/drive')
import os
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
```

Con las funciones que voy a ir escribiendo, a continuación voy recopilando qué datos me interesan de las páginas web correspondientes.

```
def data1(url, equipo):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
    columnas = ['-', '-', '-', '-', 'Local/Visitante', 'Resultado',
                'Goles a favor', 'Goles en contra', 'Rival','-' ,'Disparos',
                'Disparos a puerta', 'Porcentaje de Disparos a puerta',
                'Goles por Disparo', 'Goles por Disparo A puerta',
                'Distancia Media de los Disparos', '-',
                '-', '-','xG', 'xG sin penaltis', '-', '-',
                '-', '-']

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Incluyo la columna 'equipo' con el valor especificado
    df['equipo'] = equipo

    # Reordenar las columnas para que 'equipo' sea la primera
    columnas_reordenadas = ['equipo'] + columnas_deseadas
    df = df[columnas_reordenadas]

```

```
    # Obtener los valores del DataFrame como un array
    dat = df.values

    return dat

def data2(url):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
    columnas = ['-', '-', '-', '-','-', '-', '-', '-','-', 'Disparos A puerta en Contra','-' ,'Paradas',
                '-','-', '-', '-','-', '-', '-', '-','-', '-', '-','-', '-', '-', '-', '-','-', '-', '-','-']

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Obtener los valores del DataFrame como un array
    dat = df.values

    return dat

def data3(url):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
    columnas = ['-', '-', '-', '-','-', '-', '-', '-','-', 'Pases completados','Pases intentados','Porcentaje acierto pase','-','-',
                'Pases cortos comletados','Pases cortos intentados','Porcentaje acierto pase corto','Pases medios comletados','Pases medios intentados',
                'Porcentaje acierto pase medio','Pases largos comletados','Pases largos intentados','Porcentaje acierto pase largo',
                '-','-','-','Pases clave','-','-','-','-','-']

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Obtener los valores del DataFrame como un array
    dat = df.values

    # Configurar la forma de impresion de NumPy para evitar la notacion cientifica
```

```
    np.set_printoptions(suppress=True, precision=1, floatmode='fixed')
    return dat

def data4(url):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
    columnas = ['-', '-', '-', '-','-', '-', '-', '-','-','-',
          '-', '-', '-','-','-','-','Acciones de gol creadas','-', '-', '-','-',
              '-','-','-']

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Obtener los valores del DataFrame como un array
    dat = df.values

    return dat

def data5(url):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
    columnas = ['-', '-', '-', '-','-', '-', '-', '-','-','-','-', '-',
        '-', '-','-', '-', '-', '-','-','-', '-', '-','Intercepciones','-','
            Despejes','-','-' ]

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Obtener los valores del DataFrame como un array
    dat = df.values

    return dat

def data6(url):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
```

```
    columnas = ['-', '-', '-', '-','-', '-', '-', '-','-','Porcentaje de posesion',
                '-', '-', '-', '-','-', '-', '-','Regates intentados','Regates exitosos',
                'Porcentaje de regate existoso','-', '-', '-', '-','-', '-', 'Llegadas 1/3',
                'Entradas al area rival','-', '-', '-', '-','-']

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Obtener los valores del DataFrame como un array
    dat = df.values

    return dat

def data7(url):
    # Obtener los datos de la pagina web
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id='matchlogs_for')
    a = pd.read_html(str(table))[0]

    # Verificar los nombres de las columnas
    columnas = ['-', '-', '-', '-','-', '-', '-', '-','-','-', '-','-','Faltas cometidas',
                'Faltas recibidas','-', '-', '-', '-','-', '-', '-',
                'Recuperaciones','Duelos aereos ganados','Duelos aereos perdidos',
                'Porcentaje de victoria en duelos aereos','-']

    # Asignar los nuevos nombres de columnas al DataFrame
    a.columns = columnas

    # Filtrar las columnas no deseadas
    columnas_deseadas = [col for col in columnas if col != '-']

    # Crear el DataFrame solo con las columnas deseadas
    df = a[columnas_deseadas]

    # Obtener los valores del DataFrame como un array
    dat = df.values

    return dat
```

```
    #Con esta funcion junto la recopilacion de datos de las funciones anteriores para un equipo dado
def obtener_datos(codigo_equipo,nombre_equipo):
    # Base URL
    base_url = 'https://fbref.com/en/squads/' + codigo_equipo

    #Si cambias en la url de abajo donde pone 2023-2024, que indica temporada actual, y lo cambias a 2022-2023 carga la anterior

    # Indicar el nombre del equipo y el resto de la URL
    url1 = base_url + '/2022-2023/matchlogs/c12/shooting/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'
```

```
    url2 = base_url + '/2022-2023/matchlogs/c12/keeper/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'
    url3 = base_url + '/2022-2023/matchlogs/c12/passing/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'
    url4 = base_url + '/2022-2023/matchlogs/c12/gca/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'
    url5 = base_url + '/2022-2023/matchlogs/c12/defense/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'
    url6 = base_url + '/2022-2023/matchlogs/c12/possession/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'
    url7 = base_url + '/2022-2023/matchlogs/c12/misc/' + nombre_equipo.replace(' ', '-') + '-Match-Logs-La-Liga'

    dat1 = data1(url1, nombre_equipo)
    dat2 = data2(url2)
    dat3 = data3(url3)
    dat4 = data4(url4)
    dat5 = data5(url5)
    dat6 = data6(url6)
    dat7 = data7(url7)

    misdatos = np.concatenate((dat1, dat2[:, 0:], dat3[:, 0:], dat4[:, 0:], dat5[:, 0:], dat6[:, 0:], dat7[:, 0:]), axis=1)
    datos=misdatos[0:38]
    return datos
```

Para obtener los códigos y los nombres adecuados que posteriormente se deben pasar a la función, los busco manualmente, navegando en la página web que utilizamos durante el programa. Gracias a esto, logro descargar en mi unidad de Google Drive el conjunto de variables que he indicado para los partidos de un equipo durante la temporada seleccionada. Cabe destacar que el programa solo proporciona los datos seleccionados de un equipo y una temporada, por lo que es necesario recopilar los datos de todos los equipos uno por uno, y hacerlo igualmente para diferentes temporadas. Además, estos datos deben ser procesados en Excel, configurando ciertos formatos y utilizando la función 'Texto en columnas'.

```
#Aqui indico los datos necesarios para ir seleccionado que equipo me voy descargando
datos_equipo = obtener_datos('78ecf4bb','Almeria')

nombre_archivo = 'Almeria.csv'
ruta_archivo = '/content/drive/MyDrive/La liga'

# Guardar los datos en un archivo CSV
np.savetxt(os.path.join(ruta_archivo, nombre_archivo), datos_equipo, delimiter=',', fmt='%s')
```

```
#Estos son los nombres ordenados de las 44 variables que he ido seleccionando.
#Equipo | Local/Visitante | Resultado | Goles A Favor | Goles en contra | Rival | Tiros | Tiros a Puerta | Porcentaje Disparos a Puerta | Goles por Disparo | Goles por Disparo a Puerta | Distancia Media del Disparo | xG | xG Sin penalti | Disparos recibidos a Puerta | Paradas | Pases completados | Pases Intentados | Porcentaje Acierto Pase | Pases Cortos Completados | Pases Cortos Intentados | Porcentaje Acierto Pase Corto | Pases Medio Completados | Pases Medios Intentados | Porcentaje Acierto Pase Medio | Pases Largo Completados | Pases Largo Intentados | Porcentaje Acierto Pase Largo | Pases clave | Acciones de Gol Creadas | Intercepciones | Despejes | Porcentaje Posesion | Regates intentados | Regates Exitosos | Porcentaje Exito Regate | Llegadas Ultimo Tercio (No area) | Llegadas Area Rival | Faltas Cometidas | Faltas Recibidas | Recuperaciones | Duelos Areos Ganados | Duelos Areos Perdidos | Porcentaje de Victoria Duelos Aereos
```

II.2. Análisis del Modelo

Para el análisis de datos de la liga de fútbol, usaremos el archivo "*La liga DATOS.csv*", el cual hemos creado mediante un programa para obtener el conjunto de datos. Una vez cargados los datos, se realiza un preprocesamiento para excluir ciertos rivales del conjunto de datos original, como Valladolid, Granada, Elche, Alaves, Espanyol y Las Palmas, con el objetivo de quedarnos únicamente con los partidos que involucran a los 17 equipos que estuvieron en primera durante las temporadas 22-23 y 23-24. A continuación, procederemos con un análisis utilizando RStudio para desarrollar un posible modelo de regresión. En este análisis, exploraremos las posibilidades de poder desarrollar un sólido modelo de regresión utilizando nuestros datos. Tenemos confianza en la calidad de nuestros datos y en su potencial, lo que nos hace querer mostrar que pueden ser aprovechados para construir un modelo robusto y preciso.

```
library(readr)

DATOS <- read_delim("La liga DATOS.csv",
                            delim = ";", escape_double = FALSE,
                            trim_ws = TRUE, locale = locale(decimal_mark = ","))
LIGA=DATOS[-which(DATOS$Rival=="Valladolid"|DATOS$Rival=="Granada"|DATOS$Rival=="Elche"|
                  DATOS$Rival=="Alaves"|DATOS$Rival=="Espanyol"|DATOS$Rival=="Las Palmas"),,drop=FALSE]
```

```
#Modelo regresion
library("Rcmdr")

modeloRLM <- lm(Goles.A.Favor -Goles.en.contra ~
                Acciones.de.Gol.Creadas +Despejes +Disparos.recibidos.a.Puerta
                +Distancia.Media.del.Disparo +Duelos.Areos.Ganados +Duelos.Areos.Perdidos +
                Faltas.Cometidas +Faltas.Recibidas +Goles.por.Disparo
                +Goles.por.Disparo.a.Puerta +Intercepciones +Llegadas.Area.Rival
                +Llegadas.Ultimo.Tercio..No.area. +Local.Visitante +Paradas +Pases.clave
                +Pases.completados +Pases.Cortos.Completados +Pases.Cortos.Intentados
                +Pases.Intentados +Pases.Largo.Completados +Pases.Largo.Intentados
                +Pases.Medio.Completados +Pases.Medios.Intentados +Porcentaje.Acierto.Pase
                +Porcentaje.Acierto.Pase.Corto +Porcentaje.Acierto.Pase.Largo
                +Porcentaje.Acierto.Pase.Medio +Porcentaje.de.Victoria.Duelos.Aereos
                +Porcentaje.Disparos.a.Puerta +Porcentaje.Exito.Regate +Porcentaje.Posesion
                +Recuperaciones +Regates.Exitosos +Regates.intentados +Tiros
                +Tiros.a.Puerta +xG +xg.Sin.penalti, data = LIGA)
summary(modeloRLM)
```

```
    Residuals:
    Min      1Q  Median      3Q     Max
-1.7247 -0.1996 -0.0351  0.1505  1.6104

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                        0.0049783  0.8928701   0.006 0.995552
Acciones.de.Gol.Creadas            0.3411976  0.0127083  26.848  < 2e-16 ***
Despejes                           0.0065690  0.0019551   3.360 0.000807 ***
Disparos.recibidos.a.Puerta       -0.9576084  0.0114225 -83.835  < 2e-16 ***
Distancia.Media.del.Disparo        0.0007879  0.0044379   0.178 0.859117
Duelos.Areos.Ganados              -0.0032256  0.0048607  -0.664 0.507087
Duelos.Areos.Perdidos              0.0020243  0.0048497   0.417 0.676463
```

```
Faltas.Cometidas                          0.0066073  0.0032303   2.045 0.041066 *
Faltas.Recibidas                         -0.0014080  0.0033753  -0.417 0.676650
Goles.por.Disparo                         2.7645526  0.3422976   8.076 1.82e-15 ***
Goles.por.Disparo.a.Puerta                0.1120248  0.0960817   1.166 0.243907
Intercepciones                           -0.0019196  0.0037464  -0.512 0.608490
Llegadas.Area.Rival                       0.0053785  0.0056712   0.948 0.343149
Llegadas.Ultimo.Tercio..No.area.         -0.0037496  0.0028648  -1.309 0.190875
Local.Visitante[T.Home]                   0.0981594  0.0263588   3.724 0.000207 ***
Paradas                                   0.9318973  0.0137236  67.905  < 2e-16 ***
Pases.clave                              -0.0170188  0.0083182  -2.046 0.041010 *
Pases.completados                         0.0135368  0.0052809   2.563 0.010505 *
Pases.Cortos.Completados                 -0.0042293  0.0086762  -0.487 0.626036
Pases.Cortos.Intentados                   0.0011363  0.0074718   0.152 0.879150
Pases.Intentados                         -0.0093248  0.0039791  -2.343 0.019292 *
Pases.Largo.Completados                  -0.0233920  0.0098979  -2.363 0.018293 *
Pases.Largo.Intentados                    0.0150091  0.0057294   2.620 0.008928 **
Pases.Medio.Completados                  -0.0159604  0.0076623  -2.083 0.037496 *
Pases.Medios.Intentados                   0.0106456  0.0064371   1.654 0.098471 .
Porcentaje.Acierto.Pase                  -0.0059718  0.0183699  -0.325 0.745180
Porcentaje.Acierto.Pase.Corto            -0.0105839  0.0138522  -0.764 0.445005
Porcentaje.Acierto.Pase.Largo             0.0077817  0.0069503   1.120 0.263132
Porcentaje.Acierto.Pase.Medio             0.0100086  0.0112950   0.886 0.375764
Porcentaje.de.Victoria.Duelos.Aereos     -0.0004359  0.0023071  -0.189 0.850177
Porcentaje.Disparos.a.Puerta             -0.0103030  0.0019511  -5.281 1.57e-07 ***
Porcentaje.Exito.Regate                   0.0019806  0.0023768   0.833 0.404856
Porcentaje.Posesion                       0.0008990  0.0034242   0.263 0.792960
Recuperaciones                            0.0034620  0.0017482   1.980 0.047933 *
Regates.Exitosos                          0.0013631  0.0143534   0.095 0.924357
Regates.intentados                        0.0001333  0.0071711   0.019 0.985173
Tiros                                    -0.0128294  0.0092233  -1.391 0.164527
Tiros.a.Puerta                            0.1424456  0.0171417   8.310 2.94e-16 ***
xG                                        0.4303636  0.0468431   9.187  < 2e-16 ***
xg.Sin.penalti                           -0.3266042  0.0561358  -5.818 7.90e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3906 on 1048 degrees of freedom
Multiple R-squared:  0.945,     Adjusted R-squared:  0.943
F-statistic: 462.1 on 39 and 1048 DF,  p-value: < 2.2e-16
```

Listing II.1: summary(modeloRLM)

```
    #Test de bonferroni de valores atipicos
#ningun valor es atipico vs algun valor es atipico
outlierTest(modeloRLM)
#homocedasticidad vs #heterocedasticidad
library(zoo, pos=20)
library(lmtest, pos=20)
bptest(Goles.A.Favor - Goles.en.contra ~ Acciones.de.Gol.Creadas + Despejes
       + Disparos.recibidos.a.Puerta + Distancia.Media.del.Disparo +
         Duelos.Areos.Ganados + Duelos.Areos.Perdidos + Faltas.Cometidas +
         Faltas.Recibidas + Goles.por.Disparo + Goles.por.Disparo.a.Puerta +
         Intercepciones + Llegadas.Area.Rival + Llegadas.Ultimo.Tercio..No.area.
             +
         Local.Visitante + Paradas + Pases.clave + Pases.completados +
         Pases.Cortos.Completados + Pases.Cortos.Intentados + Pases.Intentados +
         Pases.Largo.Completados + Pases.Largo.Intentados + Pases.Medio.
             Completados +
         Pases.Medios.Intentados + Porcentaje.Acierto.Pase +
         Porcentaje.Acierto.Pase.Corto + Porcentaje.Acierto.Pase.Largo +
         Porcentaje.Acierto.Pase.Medio + Porcentaje.de.Victoria.Duelos.Aereos +
         Porcentaje.Disparos.a.Puerta + Porcentaje.Exito.Regate +
         Porcentaje.Posesion + Recuperaciones + Regates.Exitosos +
         Regates.intentados + Tiros + Tiros.a.Puerta + xG + xg.Sin.penalti,
```

```
       varformula = ~ fitted.values(modeloRLM), studentize=FALSE, data=LIGA)

#Normalidad vs no normalidad
shapiro.test(rstandard(modeloRLM))
```

```
outlierTest(modeloRLM)
     rstudent unadjusted p-value Bonferroni p
409 -4.561949       0.0000056672     0.006166
852  4.227454       0.0000256990     0.027960

Breusch-Pagan test
BP = 1.0005, df = 1, p-value = 0.3172

Shapiro-Wilk normality test

data:  rstandard(modeloRLM)
W = 0.94127, p-value < 2.2e-16
```

```
stepwise(modeloRLM, direction='backward/forward', criterion='AIC')
#hacia atras/adelante
```

Con este método de selección de variables obtenemos el siguiente modelo.

```
A=lm(formula = Goles.A.Favor - Goles.en.contra ~ Acciones.de.Gol.Creadas +
     Despejes + Disparos.recibidos.a.Puerta + Faltas.Cometidas +
     Goles.por.Disparo + Local.Visitante + Paradas + Pases.clave +
     Pases.completados + Pases.Intentados + Pases.Largo.Completados +
     Pases.Largo.Intentados + Pases.Medio.Completados + Pases.Medios.Intentados
         +
     Porcentaje.Acierto.Pase.Corto + Porcentaje.Disparos.a.Puerta +
     Porcentaje.Exito.Regate + Recuperaciones + Tiros + Tiros.a.Puerta +
     xG + xg.Sin.penalti + Porcentaje.de.Victoria.Duelos.Aereos,
   data = LIGA)
summary(A)
AIC(modeloRLM)
BIC(modeloRLM)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-1.73737 -0.19580 -0.03856  0.14704  1.59185

Coefficients:
                                    Estimate Std. Error t value
(Intercept)                        0.8898777  0.4806201   1.852
Acciones.de.Gol.Creadas            0.3451822  0.0125012  27.612
Despejes                           0.0062215  0.0018559   3.352
Disparos.recibidos.a.Puerta       -0.9592889  0.0110858 -86.533
Faltas.Cometidas                   0.0063489  0.0031413   2.021
Goles.por.Disparo                  2.9955291  0.2409799  12.431
Local.Visitante[T.Home]            0.1000289  0.0260295   3.843
Paradas                            0.9347516  0.0133983  69.767
Pases.clave                       -0.0164537  0.0080909  -2.034
Pases.completados                  0.0090158  0.0022230   4.056
Pases.Intentados                  -0.0077742  0.0020374  -3.816
Pases.Largo.Completados           -0.0102992  0.0030429  -3.385
Pases.Largo.Intentados             0.0092251  0.0025862   3.567
Pases.Medio.Completados           -0.0067281  0.0032543  -2.067
Pases.Medios.Intentados            0.0048189  0.0030502   1.580
Porcentaje.Acierto.Pase.Corto     -0.0107440  0.0050052  -2.147
Porcentaje.Disparos.a.Puerta      -0.0111321  0.0018801  -5.921
Porcentaje.Exito.Regate            0.0022946  0.0008859   2.590
Recuperaciones                     0.0032769  0.0015887   2.063
Tiros                             -0.0140295  0.0089006  -1.576
```

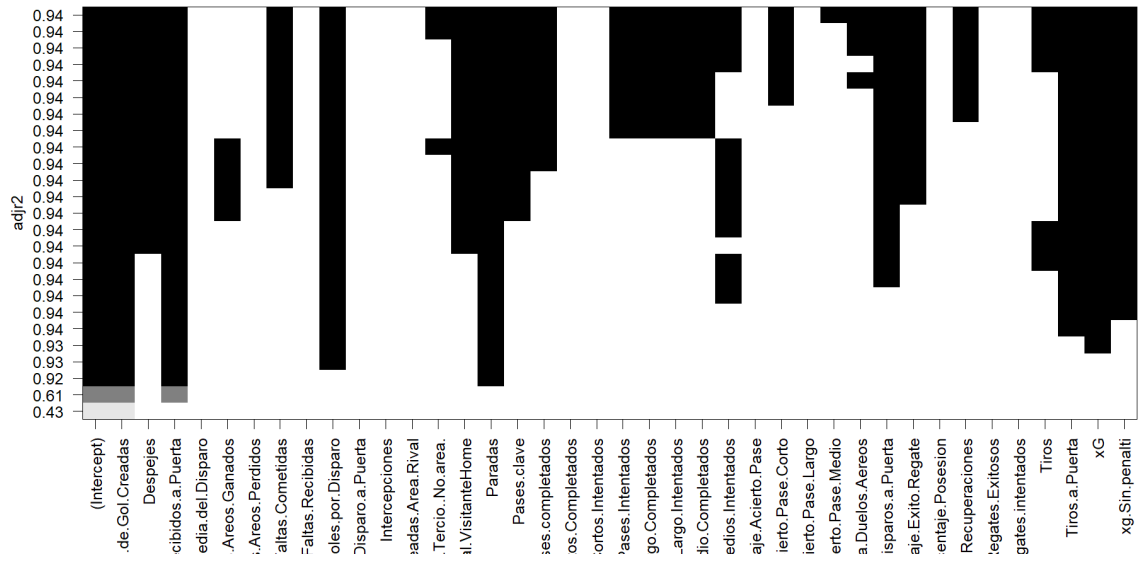
```
Tiros.a.Puerta                          0.1435504   0.0168841    8.502
xG                                      0.4314378   0.0460842    9.362
xg.Sin.penalti                         -0.3234032   0.0529075   -6.113
Porcentaje.de.Victoria.Duelos.Aereos   -0.0015934   0.0010660   -1.495
                                            Pr(>|t|)
(Intercept)                                 0.064372 .
Acciones.de.Gol.Creadas                      < 2e-16 ***
Despejes                                    0.000830 ***
Disparos.recibidos.a.Puerta                  < 2e-16 ***
Faltas.Cometidas                            0.043518 *
Goles.por.Disparo                            < 2e-16 ***
Local.Visitante[T.Home]                     0.000129 ***
Paradas                                      < 2e-16 ***
Pases.clave                                 0.042239 *
Pases.completados                      0.00005362730 ***
Pases.Intentados                            0.000144 ***
Pases.Largo.Completados                     0.000739 ***
Pases.Largo.Intentados                      0.000377 ***
Pases.Medio.Completados                     0.038930 *
Pases.Medios.Intentados                     0.114432
Porcentaje.Acierto.Pase.Corto               0.032053 *
Porcentaje.Disparos.a.Puerta           0.00000000432 ***
Porcentaje.Exito.Regate                     0.009726 **
Recuperaciones                              0.039396 *
Tiros                                       0.115268
Tiros.a.Puerta                               < 2e-16 ***
xG                                           < 2e-16 ***
xg.Sin.penalti                         0.00000000137 ***
Porcentaje.de.Victoria.Duelos.Aereos        0.135285
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3895 on 1064 degrees of freedom
Multiple R-squared:  0.9445,    Adjusted R-squared:  0.9433
F-statistic: 787.6 on 23 and 1064 DF,  p-value: < 2.2e-16

> AIC(modeloRLM)
[1] 1083.424
> BIC(modeloRLM)
[1] 1288.1
```

Listing II.2: summary(A)

Este código es el que nos va a permitir poder ver una gráfica acerca de cómo podemos seleccionar las variables; esto nos puede dar una idea de cuáles van a resultar más relevantes.

```
library(leaps, pos=23)
plot(regsubsets(Goles.A.Favor - Goles.en.contra ~ Acciones.de.Gol.Creadas +
                  Despejes + Disparos.recibidos.a.Puerta + Distancia.Media.del.
                      Disparo +
                  Duelos.Areos.Ganados + Duelos.Areos.Perdidos + Faltas.
                      Cometidas +
                  Faltas.Recibidas + Goles.por.Disparo + Goles.por.Disparo.a.
                      Puerta +
                  Intercepciones + Llegadas.Area.Rival + Llegadas.Ultimo.Tercio
                      ..No.area. +
                  Local.Visitante + Paradas + Pases.clave + Pases.completados +
                  Pases.Cortos.Completados + Pases.Cortos.Intentados + Pases.
                      Intentados +
                  Pases.Largo.Completados + Pases.Largo.Intentados + Pases.Medio
                      .Completados
                + Pases.Medios.Intentados + Porcentaje.Acierto.Pase +
                  Porcentaje.Acierto.Pase.Corto + Porcentaje.Acierto.Pase.Largo
                      +
```

```
12                    Porcentaje.Acierto.Pase.Medio + Porcentaje.de.Victoria.Duelos.
                          Aereos +
13                    Porcentaje.Disparos.a.Puerta + Porcentaje.Exito.Regate +
14                    Porcentaje.Posesion + Recuperaciones + Regates.Exitosos +
15                    Regates.intentados + Tiros + Tiros.a.Puerta + xG + xg.Sin.
                          penalti,
16                  data=LIGA, nbest=1, nvmax=25), scale='adjr2')
```



II.3. Usos del valor de Shapley

Vamos a preparar nuestros datos para su uso en modelos de regresión y clasificación en Python, de manera que le podamos aplicar el cálculo de un enfoque de los valores de Shapley, en este caso, SHAP. Esto nos permitirá comprender mejor cómo cada característica influye en las predicciones de nuestros modelos, ofreciéndonos una visión que nos ayude a entender mejor la relevancia y utilidad de los algoritmos de Shapley para calcular la importancia de cada variable en el proceso de toma de decisiones.

```
import pandas as pd
from google.colab import files
# Sube el archivo
uploaded = files.upload()

# Asegurate de que el nombre del archivo coincide con el que has subido
filename = "La liga DATOS.csv"

# Leer el archivo CSV
DATOS = pd.read_csv(filename, delimiter=';', decimal=',')

# Filtrar los datos
LIGA = DATOS[~DATOS['Rival'].isin(["Valladolid", "Granada", "Elche", "Alaves", "
    Espanyol", "Las Palmas"])]
```

```
La liga DATOS.csv
La liga DATOS.csv(text/csv) - 207375 bytes, last modified: 5/6/2024 - 100% done
Saving La liga DATOS.csv to La liga DATOS.csv
```

```
from tabulate import tabulate
summary = LIGA.describe().transpose().round(2)
summary = summary.drop("count", axis=1)
markdown_table = tabulate(summary, headers='keys', tablefmt='pipe')
print(markdown_table)
```

```
|                                    |   mean |    std |   min |
|:-----------------------------------|-------:|-------:|------:|
| Goles A Favor                      |   1.28 |   1.18 |   0   |
| Goles en contra                    |   1.28 |   1.18 |   0   |
| Tiros                              |  12.12 |   4.96 |   1   |
| Tiros a Puerta                     |   4.03 |   2.36 |   0   |
| Porcentaje Disparos a Puerta       |  33.64 |  15.92 |   0   |
| Goles por Disparo                  |   0.1  |   0.1  |   0   |
| Goles por Disparo a Puerta         |   0.28 |   0.27 |   0   |
| Distancia Media del Disparo        |  17.94 |   3.19 |   8.5 |
| xG                                 |   1.28 |   0.76 |   0   |
| xg Sin penalti                     |   1.18 |   0.7  |   0   |
| Disparos recibidos a Puerta        |   4.15 |   2.38 |   0   |
| Paradas                            |   2.89 |   1.96 |   0   |
| Pases completados                  | 386.14 | 124.38 | 121   |
| Pases Intentados                   | 486.1  | 121.15 | 206   |
| Porcentaje Acierto Pase            |  78    |   7.28 |  50.4 |
| Pases Cortos Completados           | 178.56 |  67.89 |  50   |
| Pases Cortos Intentados            | 200.94 |  68.79 |  62   |
| Porcentaje Acierto Pase Corto      |  87.77 |   4.85 |  67.4 |
| Pases Medio Completados            | 157.07 |  56.22 |  32   |
| Pases Medios Intentados            | 183.5  |  57.27 |  51   |
| Porcentaje Acierto Pase Medio      |  84.25 |   6.43 |  54.2 |
| Pases Largo Completados            |  40.41 |  10.87 |  15   |
| Pases Largo Intentados             |  75.16 |  14.05 |  34   |
| Porcentaje Acierto Pase Largo      |  53.84 |  10.61 |  25.4 |
| Pases clave                        |   9.08 |   4.14 |   1   |
```

Acciones de Gol Creadas	2.12	2.1	0	
Intercepciones	8.03	3.46	1	
Despejes	18.88	8.56	2	
Porcentaje Posesion	50	11.12	20	
Regates intentados	17.88	6.47	2	
Regates Exitosos	8.35	3.99	0	
Porcentaje Exito Regate	46.19	13.61	0	
Llegadas Ultimo Tercio (No area)	13.09	6.21	1	
Llegadas Area Rival	4.24	2.89	0	
Faltas Cometidas	12.95	4.18	3	
Faltas Recibidas	12.33	4.02	3	
Recuperaciones	49.75	8.85	24	
Duelos Areos Ganados	13.77	6.47	0	
Duelos Areos Perdidos	13.77	6.47	0	
Porcentaje de Victoria Duelos Aereos	50	11.78	0	

	25%	50%	75%	max
Goles A Favor	0	1	2	7
Goles en contra	0	1	2	7
Tiros	9	12	15	35
Tiros a Puerta	2	4	5	15
Porcentaje Disparos a Puerta	22.2	33.3	43.12	100
Goles por Disparo	0	0.08	0.15	0.67
Goles por Disparo a Puerta	0	0.25	0.5	1
Distancia Media del Disparo	15.8	17.7	19.73	31.9
xG	0.7	1.2	1.7	4.6
xg Sin penalti	0.7	1.1	1.6	4.5
Disparos recibidos a Puerta	3	4	6	15
Paradas	1	3	4	13
Pases completados	293	378	472	788
Pases Intentados	396.75	477.5	567	878
Porcentaje Acierto Pase	73.47	78.9	83.3	92.5
Pases Cortos Completados	130.75	168	216	412
Pases Cortos Intentados	152.75	192	239	441
Porcentaje Acierto Pase Corto	85	88.5	91.3	97.4
Pases Medio Completados	113	156	197	355
Pases Medios Intentados	138	183	223	377
Porcentaje Acierto Pase Medio	80.9	85.3	88.9	95.9
Pases Largo Completados	33	40	47	80
Pases Largo Intentados	65	74	84	127
Porcentaje Acierto Pase Largo	46.4	53.6	61.4	85.7
Pases clave	6	9	11	29
Acciones de Gol Creadas	0	2	4	12
Intercepciones	5	8	10	21
Despejes	13	17	24	56
Porcentaje Posesion	42	50	58	80
Regates intentados	13	17	22	48
Regates Exitosos	5	8	11	26
Porcentaje Exito Regate	37.5	46.2	55	100
Llegadas Ultimo Tercio (No area)	9	12	16	45
Llegadas Area Rival	2	4	6	16
Faltas Cometidas	10	13	16	31
Faltas Recibidas	10	12	15	30
Recuperaciones	44	50	55	77
Duelos Areos Ganados	9	13	18	41
Duelos Areos Perdidos	9	13	18	41
Porcentaje de Victoria Duelos Aereos	42.3	50	57.7	100

Listing II.3: Resumen de las variables numéricas

Código pensado para el modelo de regresión lineal

```
from sklearn.model_selection import train_test_split
```

```

# Definir las caracteristicas predictoras excluyendo las ya usadas y las que no se desean
#Para el modelo de regresion (difencia de goles)
y = LIGA['Goles A Favor'] - LIGA['Goles en contra']
X = LIGA.drop(columns=['Goles A Favor', 'Goles en contra', 'Equipo', 'Rival', 'Resultado'])



# Convertir variables categoricas a variables dummy (one-hot encoding)
X = pd.get_dummies(X, drop_first=True)

# Verificar que todas las columnas son numericas
assert all(X.dtypes != 'object'), "There are still non-numeric columns in X"

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=42)

# Mostrar las dimensiones de los conjuntos de datos
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
# Convertir columnas categoricas en variables dummy
dat1 = LIGA.drop(columns=[ 'Rival', 'Resultado'])
dat2 = LIGA.drop(columns=[ 'Rival','Equipo' ,'Resultado'])

# Convertir la variable 'Local/Visitante' en una variable dummy
datos1 = pd.get_dummies(dat1, columns=['Local/Visitante'], drop_first=True)
datos2 = pd.get_dummies(dat2, columns=['Local/Visitante'], drop_first=True)

# Calcular las medias
teams = datos1.groupby('Equipo').mean()
total=datos2.mean()

# Convertir la serie total en un dataframe
total_df = total.to_frame().T

# Concatenar total_df y teams
promedios = pd.concat([total_df, teams], axis=0)
# print(promedios)

# Extraer los nombres de los equipos como el indice del dataframe teams
orden = promedios.index

# Ponemos los promedios en el formato de los datos de entrenamiento
y_pro = promedios['Goles A Favor'] - promedios['Goles en contra']
X_pro = promedios.drop(columns=['Goles A Favor', 'Goles en contra'])

#Los uno para tener los promedios en el calculo de los valores de Shapley
y_test = pd.concat([y_test, y_pro], ignore_index=True)
X_test = pd.concat([X_test, X_pro], ignore_index=True)

# Mostrar las dimensiones de los nuevos conjuntos de datos
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_absolute_error


# Crear y entrenar el modelo de regresion lineal
model = LinearRegression()
model.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_test)

# Calcular el error absoluto medio (MAE)
mae = mean_absolute_error(y_test, y_pred)
print(f"MAE: {mae:.2f}")
```

```
MAE: 0.29
```

```
import numpy as np

# Obtener los coeficientes del modelo
coefs = pd.DataFrame({
    'feature': X.columns.values,
    'coefficient': np.round(model.coef_, 3)
})

# Imprimir los coeficientes en formato markdown
print(coefs.to_markdown(index=False))
```

```
| feature                              |   coefficient |
|:-------------------------------------|--------------:|
| Tiros                                |        -0.011 |
| Tiros a Puerta                       |         0.145 |
| Porcentaje Disparos a Puerta         |        -0.01  |
| Goles por Disparo                    |         2.829 |
| Goles por Disparo a Puerta           |         0.146 |
| Distancia Media del Disparo          |        -0     |
| xG                                   |         0.475 |
| xg Sin penalti                       |        -0.386 |
| Disparos recibidos a Puerta          |        -0.947 |
| Paradas                              |         0.922 |
| Pases completados                    |         0.015 |
| Pases Intentados                     |        -0.01  |
| Porcentaje Acierto Pase              |        -0.008 |
| Pases Cortos Completados             |        -0.012 |
| Pases Cortos Intentados              |         0.008 |
| Porcentaje Acierto Pase Corto        |        -0.001 |
| Pases Medio Completados              |        -0.014 |
| Pases Medios Intentados              |         0.008 |
| Porcentaje Acierto Pase Medio        |         0.005 |
| Pases Largo Completados              |        -0.021 |
| Pases Largo Intentados               |         0.014 |
| Porcentaje Acierto Pase Largo        |         0.006 |
| Pases clave                          |        -0.018 |
| Acciones de Gol Creadas              |         0.333 |
| Intercepciones                       |        -0     |
| Despejes                             |         0.007 |
| Porcentaje Posesion                  |         0.003 |
| Regates intentados                   |        -0     |
| Regates Exitosos                     |         0.008 |
| Porcentaje Exito Regate              |         0     |
| Llegadas Ultimo Tercio (No area)     |        -0.003 |
| Llegadas Area Rival                  |         0.001 |
| Faltas Cometidas                     |         0.004 |
```

```
| Faltas Recibidas                    |         -0.007 |
| Recuperaciones                      |          0.003 |
| Duelos Areos Ganados                |         -0.007 |
| Duelos Areos Perdidos               |          0.005 |
| Porcentaje de Victoria Duelos Aereos |          0.001 |
| Local/Visitante_Home                |          0.111 |
```

Listing II.4: Tabla de coeficientes

```
!pip install shap

# Importar las bibliotecas necesarias
import shap

# Calcular los valores SHAP
explainer = shap.LinearExplainer(model, X_train)

# Calcular los valores SHAP llamando al explainer con los datos a explicar.
shap_values = explainer(X_test)

#Otra forma
# try1 = shap.Explainer(model, X_test)
# shap_values = try1(X_test)
```

```
import matplotlib.pyplot as plt
from google.colab import files
# Crear y guardar la grfica
plt.figure()
#El indice cero es el primero de la lista de los valores guardados para test
shap.plots.waterfall(shap_values[0], max_display=15, show=True)
plt.savefig('shap_waterfall_plot.png')
plt.close()

# Descargar la grafica
files.download('shap_waterfall_plot.png')
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Crear el grafico de dispersion
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test[165:], y=y_pred[165:], s=100, color='green', label='
    Equipos')
sns.scatterplot(x=y_test[:164], y=y_pred[:164], s=100, color='blue', label='
    Partidas individuales')


# Linea de mejor ajuste
sns.regplot(x=y_test, y=y_pred, scatter=False, color='red', line_kws={"label":"
    Linea de mejor ajuste"})

# Titulos y etiquetas
plt.title('Comparacion del Rendimiento Promedio de Puntuacion de Equipos')
plt.xlabel('Puntuacion Promedio Real de Equipos')
plt.ylabel('Puntuacion Promedio Predicha de Equipos')

# Mostrar la leyenda
plt.legend()

# Mostrar el grafico
plt.grid(True)
plt.show()
```

Código pensado para el modelo de clasificación

```
# !pip install shap
import numpy as np
import xgboost as xgb
import shap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Convertir la columna 'Resultado' a etiquetas numericas
label_encoder = LabelEncoder()
LIGA['Resultado'] = label_encoder.fit_transform(LIGA['Resultado'])

#Win: W => 2
#Lose: L => 1
#Draw: D => 0
```

```
#Para poder crear el modelo elimina la siguiente variable
#Cuidado que si la hemos convertido ya a dummy puede tener otro nombre
LIGA= LIGA.drop(columns=['Local/Visitante'])

# Separar las caracteristicas y el objetivo
y = LIGA['Resultado']
X = LIGA.drop(columns=['Resultado','Equipo','Rival','Goles A Favor', 'Goles en
    contra'])

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,
    random_state=42)

# Convertir columnas categoricas en variables dummy
dat1 = LIGA.drop(columns=[ 'Rival','Goles A Favor', 'Goles en contra'])

# Calcular las medias y seleccionar un equipo
teams = dat1.groupby('Equipo').mean()
real_madrid = teams.loc['Real Madrid']

# Ponemos el equipo en el formato de los datos de entrenamiento
y_rm = real_madrid['Resultado']
X_rm = real_madrid[1:]


# Convertir la serie total en un dataframe
X_rmdat = X_rm.to_frame().T
y_rmdat = pd.Series(y_rm)

# Los uno para tener los promedios en el calculo de los valores de Shapley
y_test = pd.concat([y_test, y_rmdat], ignore_index=True)
X_test = pd.concat([X_test, X_rmdat], ignore_index=True)

# Mostrar las dimensiones de los nuevos conjuntos de datos
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")

# Crear el DMatrix de XGBoost
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)

# Entrenar el modelo XGBoost
params = {
    'objective': 'multi:softprob',
    'num_class': 3,
```

```
    'learning_rate': 0.01,
    'eval_metric': 'mlogloss'
}
bst = xgb.train(params, dtrain, num_boost_round=100)

# Predecir y explicar el modelo con SHAP
explainer = shap.TreeExplainer(bst)
shap_values = explainer(dtest)
```

```
#Vamos a necesitar saber cuales son los nombre de cara a la interpretacion de la grafica
feat_names = list(X.columns)
for i, feat_name in enumerate(feat_names):
    print(f"Feature {i}: {feat_name}")
```

```
Feature 0: Tiros
Feature 1: Tiros a Puerta
Feature 2: Porcentaje Disparos a Puerta
Feature 3: Goles por Disparo
Feature 4: Goles por Disparo a Puerta
Feature 5: Distancia Media del Disparo
Feature 6: xG [suma de las probabilidades estimadas de gol de un disparo]
Feature 7: xg Sin penalti
Feature 8: Disparos recibidos a Puerta
Feature 9: Paradas
Feature 10: Pases completados
Feature 11: Pases Intentados
Feature 12: Porcentaje Acierto Pase
Feature 13: Pases Cortos Completados
Feature 14: Pases Cortos Intentados
Feature 15: Porcentaje Acierto Pase Corto
Feature 16: Pases Medio Completados
Feature 17: Pases Medios Intentados
Feature 18: Porcentaje Acierto Pase Medio
Feature 19: Pases Largo Completados
Feature 20: Pases Largo Intentados
Feature 21: Porcentaje Acierto Pase Largo
Feature 22: Pases clave
Feature 23: Acciones de Gol Creadas
Feature 24: Intercepciones
Feature 25: Despejes
Feature 26: Porcentaje Posesion
Feature 27: Regates intentados
Feature 28: Regates Exitosos
Feature 29: Porcentaje Exito Regate
Feature 30: Llegadas Ultimo Tercio (No area)
Feature 31: Llegadas Area Rival
Feature 32: Faltas Cometidas
Feature 33: Faltas Recibidas
Feature 34: Recuperaciones
Feature 35: Duelos Areos Ganados
Feature 36: Duelos Areos Perdidos
Feature 37: Porcentaje de Victoria Duelos Aereos
```

Listing II.5: Orden de características

```
import shap
import matplotlib.pyplot as plt

# Asume que `shap_values` ya esta definido
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Primera grafica
```

```
plt.sca(axes[0])
shap.plots.waterfall(shap_values[1,:,2], show=False)
axes[0].set_title('Nombre 1')

# Segunda grafica
plt.sca(axes[1])
shap.plots.waterfall(shap_values[1,:,0], show=False)
axes[1].set_title('Nombre 2')

# # Tercera grafica
# plt.sca(axes[2])
# shap.plots.waterfall(shap_values[0,:,1], show=False)
# axes[2].set_title('Nombre 3')

# Ajustar los graficos
plt.subplots_adjust(wspace=0.7)  # Incrementa el espacio entre los graficos
plt.show()

# Con 3 se empiezan a montar las graficas
```