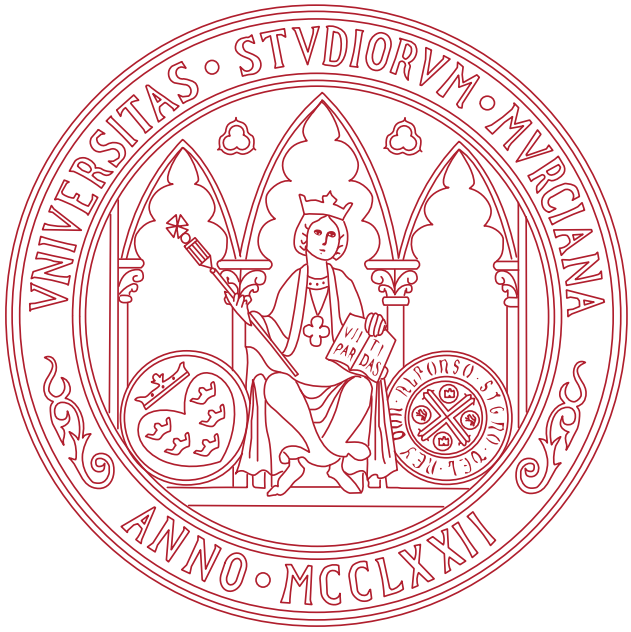

Universidad de Murcia

Trabajo Final de Grado

Explicando Tree SHAP: una herramienta para explicar un explicador

Autor:
José Antonio Nicolás Navarro
joseantonio.nicolasn@um.es

Tutores:
José Manuel Juárez Herrero
jmjuarez@um.es
Manuel Campos Martínez
manuelcampos@um.es

15 de junio de 2021

Índice general

Índice de figuras

Resumen

El uso en entornos reales de la tecnología basada en aprendizaje automático (ML) para la toma de decisiones en áreas críticas como la atención médica está generando preocupación en ciertos sectores de la sociedad debido a su uso como 'cajas negras', de las que no se emite ninguna explicación sobre las decisiones que son tomadas. Esta preocupación se refleja en el derecho a la explicación y una mayor transparencia respaldada por mecanismos regulatorios como el RGPD y la legislación propuesta para regular la inteligencia artificial en la Unión Europea en abril de 2021.

Dentro del campo de la Inteligencia Artificial Explicable (XAI), se desarrollan métodos automatizados para lograr explicar predicciones de modelos de aprendizaje automático adquiridos. A día de hoy, los únicos métodos de explicación que son capaces de producir explicaciones, que puedan ser respaldadas legalmente, son aquellos que se asientan sobre formalismos y fundamentos matemáticos sólidos. Las explicaciones basadas en valores de Shapley de Teoría de Juegos son uno de los escasos métodos de los que se dispone, aunque siguen en desarrollo.

Si bien es cierto que estos métodos XAI formales logran producir explicaciones tremendamente útiles, el proceso seguido para su obtención es tan complejo que, paradójicamente, también son utilizados como "cajas negras". Este trabajo surge de este problema, donde se propone contemplar la idea de "un explicador del explicador" que sea útil tanto a un científico de datos que decida utilizar una herramienta de apredizaje automático, como a un usuario final (como podría ser un médico) para poder tener una idea más intuitiva de cómo se han construido las explicaciones.

En particular, este proyecto tiene como objetivo el desarrollo de una herramienta para explicar el método Tree SHAP, un método específico para la explicación de modelos de árboles de decisión. Dentro del trabajo, se compara con otros métodos formales, como son los valores Shapley, y se muestra el procedimiento interno de su funcionamiento, tanto de manera formal, con la formulación matemática de sus cálculos, como de manera visual, construyendo un grafo de coaliciones interactivo. Esta utilidad se ha probado en un caso de uso del ámbito médico para un problema de resistencia antibiótica hospitalaria.

Extended abstract

In this work is proposed and addressed the idea of contemplating an "explainer of the explainer" Tree SHAP [36], a specific algorithm within the SHAP method, in order to obtain, in addition to an explanation of which characteristics are most important for a prediction, the calculations that were carried out to obtain the 'importance values' of each one of them, thus providing transparency and reliability to the method. This idea emerges from the fact that explainability techniques are used as 'black boxes' to be able to interpret *machine-learning* (ML) models, despite the fact that their very existence is based on avoiding this opacity. The impact of this type of models, used blindly and carelessly, is unacceptable in many cases, as is the case in the field of health care, due to the magnitude of the consequences that a decision can have on a patient's diagnosis.

Thus, the main objective of this work is to provide a tool that facilitates the interpretation of explanations of predictions made by decision tree models obtained by ML, by users who are not experts in formal mathematical techniques. Specifically, this tool will focus on explaining the technique Tree SHAP, as it is currently the only explainer with a proven formal mathematical basis, and also explaining the mathematical origin of the values computed to validate the technique and give it reliability, because explainers based on strongly formal foundations are not universally transferable to the human user and these explanations can be unintuitive and generate little confidence.

The tool has two target users: data scientists, to facilitate the corroboration of whether the interpretation tool used (Tree SHAP) is the most appropriate for the problem, and the end medical users, to help explain how the Tree SHAP explainer works, favoring the transparency of the calculation of the explanation values.

As for proposed sub-objectives, we considered the following: study and implementation of TreeExplainer's Tree SHAP algorithm, a white-box and black-box analysis to compare the development of the previous sub-objective and the version developed by Scott M. Lundberg, an empirical study between Shapley values and SHAP values from Tree SHAP, the development of an interactive tool to understand the process of calculating Tree SHAP values and finally a use case within the medical field with antibiotic resistance in Intensive Care Units.

The developed tool, written in *Python*, consists of two separate modules; the implementation of the Tree SHAP algorithm, fragmented into two classes to separate the functions used in the algorithm from the algorithm itself, and on the other hand, the module with those classes of the developed utility that have to do with the construction,

management and interaction of the coalition graph on which it is based, as well as with the calculation of Shapley values and its mathematical trace. These modules are not related, nor should they be, in order to operate independently, although the values they produce can be put in common if desired.

It is therefore not a fully integrated application with cross-platform support (although it has been tested to work on Windows and Mac), but just another library for *Python*, such as SHAP [35], which allows to get the source of the calculations performed by S. Lundberg's original implementation and browse through them.

The first algorithm implemented in the conduction of this work is Tree SHAP. Lundberg, in his paper [36], states that this algorithm performs the equivalent work of computing Shapley values by applying the conditional expectation as a characteristic function *val* of the following equation recursively:

$$\phi_i(val) = \frac{1}{|N|!} \sum_R [val(P_i^R \cup \{i\}) - val(P_i^R)] \tag{1}$$

This equation satisfies three fundamental properties [36] in Cooperative Game Theory that guarantee a fair and equitable distribution of importances for a given set of features with which a model has been trained.

To implement it, I had to clarify many previous concepts and follow the pseudocode in the article that proposes it [36], determining the appropriate data structures for each variable suggested. This is an algorithm that does not lend itself to being easily understood, although it does not present that much of a problem when it comes to implementing it, since the most complex data types that are part of it are lists and the rest of the work is to determine what each variable is used for and what form it takes when any of the functions receive it as an input parameter.

Once the Tree SHAP implementation was finished, it was necessary to test it to be sure that the values of my implementation matched those of Lundberg's implementation.

For that, I developed a script that took different datasets with more than enough instances of the SHAP library, which are usually the typical ones to illustrate examples within ML (Iris, Boston, Adult...). These datasets cover regression and classification problems (the computation is different for one type of problem and another). In addition to these datasets, I have taken into account the decision tree based ML models supported by *scikit-learn*; Decision Tree, Random Forest and Gradient Boosting. With these datasets and trained models, my code generates all possible combinations and runs both versions of Tree SHAP.

To understand the fundamentals of my original contribution, 'Explainer of Tree SHAP' it is necessary to introduce certain ideas that motivated this explainer of the explanations that Tree SHAP generates. The values that Tree SHAP computes come from Game Theory. Lundberg, in his paper [36], states that this algorithm performs the equivalent work of 'computing Shapley values by applying conditional expectation' to the characteristic function *val* of the equation 1.

Originally, classical Shapley values operate by means of the marginal expectation, which gets rid of the limitation produced by the tree structure in models of this type. Conditional expectation does take the tree structure into account, which may seem a limiting factor a priori, although the only difference that occurs in the Shapley values using conditional expectation is the fact that features that have no impact on a prediction can obtain a value other than 0. This fact does not affect the explanation globally. Therefore, these Shapley values, with conditional expectation as a characteristic function, are accurate except in the situation described.

And why is the marginal expectation not taken into account? The calculation of Shapley values with this function is fundamentally theoretical and its implementation would be complex since it depends on the randomness of the data and its selection for the training of many models, which test how the aggregation of a characteristic affects sets with different combinations of them to calculate an average according to the predictions obtained, which is known as marginal contribution. In fact, the other methods that calculate these values are simply approximations since they do not take into account all the coalitions that can be formed to make their computation less burdensome.

For this reason, and because calculating Shapley values by following the equation to the letter is computationally expensive, such implementations are not popular and of no interest outside of research and pure academia, so they do not seem to exist.

These data serve to illustrate the mental process I followed to make the decision to implement, in an original way since there does not seem to be any published code available for use, the calculation of Shapley values, using conditional expectation as a characteristic function.

The situation was as follows: I wanted to somehow explain the explanations produced by Tree SHAP. I carried out the implementation of this algorithm in order to, in principle, modify it and use it for this purpose. Finally, we have the basis on which the calculation is based. With this on the table, it was concluded that using Tree SHAP to explain itself is impossible because it takes into account 2^M feature sets, M being the number of features in a data set, whose marginal contributions are computed simultaneously and in a recursive fashion. This fact together with the inherent complexity of the algorithm makes the trace, therefore, practically untraceable, thus needing a different approach.

This approach is based on the implementation and use of the algorithm that calculates the conditional expectation of a S set of features, which is mentioned and shown in the same paper [36], which would have to be applied on the calculation of Shapley values and which I had to implement as well. If Tree SHAP indeed was based on the fundamentals described in section 4.2.1 .... The values produced should be the same and the trace of their calculation would have to be more manageable and useful for querying. For this I have relied heavily on the (non-scientific) article "SHAP Part 3: Tree SHAP" [63] and used it as inspiration.

The possible orderings can be represented as an acyclic graph, as shown in figure 1, where each node represents a coalition of features and each axis the newly added feature with respect to the previous set. This arrangement helps to visualize what is

happening in the equation 1; the descent through the graph represents the addition of new features to the coalition. The difference between the value computed by the conditional expectation with the set at a parent node and at its child node is the marginal contribution of the feature on the edge joining the two nodes, at that computational step. The sum of all marginal contributions for each feature, among the total number of ordinations, is its exact Shapley value.

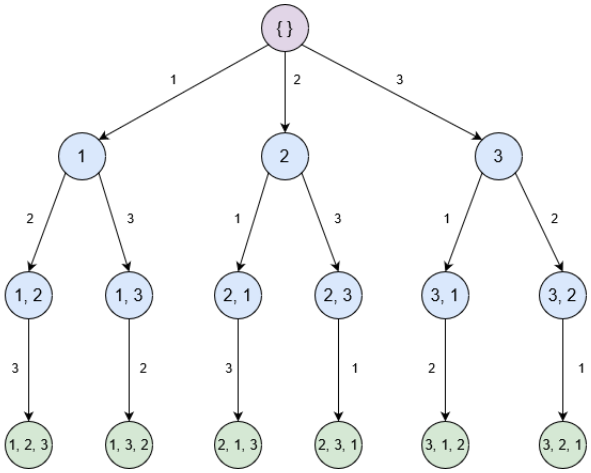

Figure 1: Coalitions graph of an artificial dataset.

Moving on to the developed tool, the trace of the calculations are displayed in one of two ways depending on the complexity of the input model. Models that are simple enough based on their number of features can choose to be visualized through an interactive coalition graph, as shown in figure 2, in addition to getting a complete trace of the calculations in text form, which is output in a file separately.

If the model is more complex, it is not worthwhile to visualize the calculations graphically because they start to become too many, as the number of nodes generated grows by a factorial increase according to the number of features.

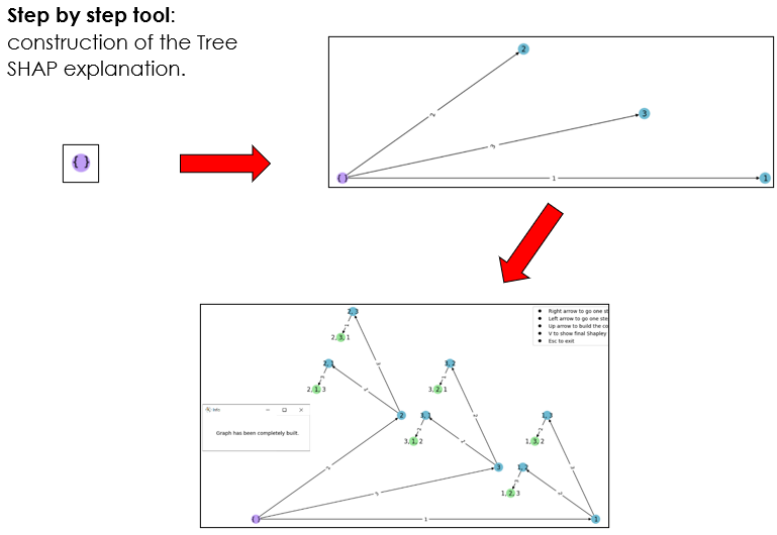

Figure 2: Step by step construction of the coalition graph.

The graph that generates the utility is interactive, as can be seen in figure 3, meaning that it is possible to go forward and backward in the construction of the graph as much as desired, as well as to click inside the nodes, which represent each possible ordering given by the Shapley values equation, to consult the status of the calculations of the marginal contributions that are part of the calculation of the final Shapley values. It is also possible to construct the entire graph quickly.

Another feature of this graph is that, in order to make it easier to follow its construction despite its layout on the screen, the first node and the leaf nodes have different colors than the internal nodes.

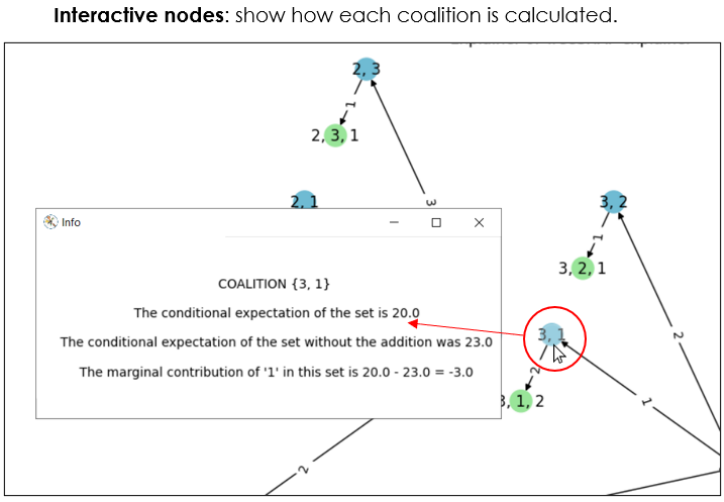

Figure 3: Interactive nodes of the coalition graph.

In addition to this study and implementation of the tool described, I carried out a real-world experimentation using MIMIC-III, a healthcare database that holds information on more than 40,000 patients in the Intensive Care Unit, to show how Tree SHAP works to use my explainer for Tree SHAP as a useful healthcare tool. From this database, through the SITSUS project, a query and subsequent filtering of demographic data, samples and susceptibility study has been provided under specialized supervision.

I believe that the objectives have been achieved, since the tool developed is useful for the target users for whom it was designed. First of all, in the case of data scientists, it can be used to facilitate the verification of whether the explanatory method used (in this case Tree SHAP) is the most appropriate for the problem to be solved. And secondly, for end users, such as healthcare professionals, it serves as a support for decision making in the clinical setting, since it explains how the explainer works, favoring the transparency of the calculation of the importance values of the characteristics, thus validating and giving reliability to the discoveries that are made.

Despite the usefulness of the developed tool, it is still a 'black box' for end users in the sense that solid data science and machine learning, and even mathematical knowledge is required for the source of significance values produced from a prediction to become truly reliable and assist in the medical decisions that are made every day.

The problem lies in the fact that the ingenious Tree SHAP technique is based on strongly formal foundations, and this type of model is not easily transferable to humans

in a universal way, so as long as a new model, just as powerful and effective, does not enter the scene in this paradigm, whose foundations are not based on excessively complex mathematical structures, it is inevitable to use these explainability techniques as 'black boxes', with which one 'must believe' that the value of importance assigned to each characteristic within a data set is reliable, despite knowing the origin of its calculation or having it available.

In this sense, there is still a long way to go in the field of explainability, since we have fast and apparently reliable methods, but they are still not entirely intuitive if they are very complex, and can provoke rejection and mistrust if they are not. It is a difficult balance to achieve and one that will certainly be difficult to achieve in the near future.

CAPÍTULO 1

Introducción

El constante desarrollo y demanda de nuevas tecnologías, junto con el nivel de digitalización que está alcanzando la sociedad en los últimos años, está permitiendo registrar y almacenar cantidades de datos masivas provenientes de dispositivos móviles, bases de datos de diferentes ámbitos y del uso de aplicaciones a gran escala, entre otras fuentes de información. Uno de los entornos donde también se ha producido este proceso transformador es el ámbito de la salud, con la llegada de la digitalización de todos aquellos datos relativos al cuidado de pacientes y a la gestión clínica, como las pruebas de imagen, historiales clínicos, etc.

Disponer de esta gran cantidad de datos que se está produciendo permite aplicar técnicas de inteligencia artificial que se conocían y fueron desarrolladas hace décadas, pero que no podían ser utilizadas con su mayor potencial, puesto que la sociedad aún no había sufrido el proceso de digitalización necesario para ello. Ahora que se tienen los datos disponibles para su explotación dentro del ámbito de la salud, se requiere de especialistas con un alto nivel de conocimiento en técnicas relacionadas con el tratamiento de estos datos e inteligencia artificial.

Tanto es así que han surgido nuevos perfiles profesionales, como el de científico de datos, para el que no existe una descripción perentoria sobre la labor que desempeña, pero que está especializado en el análisis de datos y el uso de la estadística y el aprendizaje automático para la búsqueda, análisis e interpretación de patrones subyacentes en los mismos. Además, extrae y convierte información de datos (estructurados y no estructurados) en conocimiento útil para diversos campos. La diferencia fundamental que guarda con el analista de datos es que el perfil de científico de datos posee una visión más amplia y global del problema que se quiere solucionar y realiza un análisis teniendo en cuenta distintas fuentes de información.

A día de hoy, el uso del aprendizaje automático para realizar predicciones a partir de un conjunto de datos y asistir en la toma de decisiones dentro de diferentes campos de la ciencia, sobre todo en el ámbito de la salud, está tomando cada vez más importancia debido al potencial y a los beneficios que viene reportando en los últimos años [21] [2]

[39] [69]. Durante este tiempo, también se ha producido la eclosión del Deep Learning, fruto del desarrollo de nuevas arquitecturas de redes neuronales, debido en su mayor parte a la mejora en la capacidad de cómputo que albergan las nuevas arquitecturas hardware dedicadas, pensadas para este tipo de problemas.

De esta importancia surge la demanda social de que se pueda ser capaz de explicar por qué una inteligencia artificial ha realizado una predicción o tomado una decisión concreta y no cualquier otra. Establecer su origen en determinadas situaciones puede llegar a ser crítico para poder ofrecer ciertas garantías, que generen confianza en las áreas de aplicación donde estos modelos son utilizados, siendo más incisivos en el ámbito de la salud, donde estas decisiones tienen más impacto por su naturaleza intrínseca al campo médico.

Nacen de esta forma distintas técnicas para tratar de cubrir esta nueva necesidad, como la utilización de modelos que sean interpretables de forma inmediata, como es el caso de los árboles de decisión, o diferentes técnicas de inteligencia artificial explicable (XAI), como IME [61], LIME [52], MAPLE [49] y más recientemente SHAP [34], entre otras, como respuesta a esta demanda social. Estas técnicas pueden ser 'model-specific' o 'model-agnostic'. La diferencia entre ambas es que las del primer tipo generan una explicación dependiente del modelo al que se quiere dar explicabilidad y las del segundo tipo generan una explicación independiente del modelo que se esté utilizando, siendo mucho más flexible la técnica de explicabilidad en este último caso. Además, esta explicación puede realizarse sobre una sola predicción (explicación local del modelo) o sobre el modelo al completo (explicación global del modelo).

Esta creciente demanda también requiere de mecanismos legales que amparen el derecho a la explicación y afronten los problemas provenientes de la importancia que están cobrando los algoritmos, contemplados por la UE en el Reglamento General de Protección de Datos (RGPD). Los sesgos racistas o sexistas que podría tener una IA, así como aplicaciones de IA de alto riesgo que vulneren derechos o pongan vidas en peligro, deben ser consideradas como inaceptables y ya existe una propuesta de ley en la UE [1] para regular estrictamente su uso de forma que vaya acorde con los valores éticos y morales europeos. En lugares como EEUU, las compañías de seguros, por ejemplo, están obligadas a poder explicar las decisiones que sus algoritmos toman con las tarifas y cobertura de sus clientes [31].

Paradójicamente, las técnicas de explicabilidad son usadas como 'cajas negras' para poder interpretar los modelos de *machine-learning* (ML), a pesar de que su misma existencia se basa en evitar esta opacidad. Esto viene dado por el hecho de que los modelos de explicabilidad con una sólida componente matemática son difíciles de entender para los usuarios finales, que podrían ser médicos en este caso, puesto que exigen un conocimiento formal previo. Un ejemplo de este tipo de técnicas es el método SHAP, cuyas explicaciones se cimentan sobre Teoría de Juegos, con los valores de Shapley. El impacto de este tipo de modelos, utilizados ciegamente de manera despreocupada, es inaceptable en muchos casos, como ocurre dentro del ámbito de la salud por la magnitud de las consecuencias que una decisión puede provocar en el diagnóstico de un paciente.

Es por ello que en este trabajo se propone y se aborda la idea de contemplar un

"explicador del explicador" TreeSHAP, un algoritmo concreto dentro del método SHAP, para poder obtener, además de una explicación sobre qué características son más importantes para una predicción, los cálculos que fueron llevados a cabo para la obtención de los 'valores de importancia' de cada una de ellas, dotando así de transparencia y fiabilidad al método.

CAPÍTULO 2

Estado del arte

2.1. Inteligencia Artificial Explicable (XAI)

2.1.1. Historia

El problema de la inteligencia artificial explicable no es algo que haya aparecido recientemente, sino que lleva explorándose e investigándose desde los años 70, aunque en los últimos años ha tomado un mayor interés por los avances que se han producido en ML, y por el gran potencial que se tiene a día de hoy en cuanto a aplicaciones dentro del mundo real gracias a la gran disponibilidad de datos que existe.

Entre los años 1970 y 1990, se exploraron sistemas de razonamiento simbólico, como MYCIN [20], GUIDON [18], SOPHIE [7] y PROTOS [3][4] que podían representar, razonar y explicar su razonamiento con el fin de diagnosticar para o aplicarlo dentro de técnicas de aprendizaje automático.

Entre los años ochenta y principios de los noventa, se desarrollaron sistemas de mantenimiento de la verdad (TMS) que permitían generar explicaciones a partir de las trazas de razonamiento producidas por las operaciones de reglas o inferencias lógicas [54].

Durante la década de 1990, los investigadores intentaron desarrollar explicaciones dinámicas para conseguir darle a estas tecnologías más fiabilidad en la práctica [17]. En la década de 2010, aumentó la demanda de IA transparente debido a los sesgos raciales y de más tipos producidos por el uso de IA para asistir en la toma de decisiones sobre sentencias penales [55].

Las técnicas más modernas y complejas de IA, como el aprendizaje profundo y los algoritmos genéticos, son inherentemente opacas [44] y esto ha propiciado un desarrollo de muchos nuevos métodos para intentar dotar a estos modelos de explicabilidad e interpretabilidad [38][52][34][9][59][64]. Se ha conseguido entrenar modelos basados en redes neuronales para que proporcionen explicaciones lingüísticas de su comportamien-

to, interpretables por el ser humano [28]. Además de estos modelos, se ha trabajado con árboles de decisión, 'ensembles' de árboles y redes bayesianas, que son menos opacos a la inspección [22][68].

Actualmente, existen múltiples investigaciones en curso y continúan asentándose las bases de lo que será el nuevo paradigma dentro del campo del aprendizaje automático y la interpretabilidad en el futuro.

2.1.2. Explicabilidad en ML: métodos interpretables vs. cajas negras

Se dice que un modelo es un modelo caja negra si bien la función que el modelo calcula es demasiado complicada para que cualquier humano la comprenda, o bien, si ocurre que aun siendo el modelo muy sencillo, sus detalles quedan ocultos, no pudiendo ser inspeccionados para su revisión [13]. Es por esto que en este tipo de modelos se hace necesario el uso de un segundo modelo auxiliar cuyo propósito es explicar el modelo principal.

Por otro lado, un modelo interpretable [13] es un modelo utilizado para realizar predicciones, igual que un modelo caja negra, pero la diferencia es que puede ser inspeccionado e interpretado directamente por expertos humanos, sin la necesidad de construir un segundo modelo que proporcione explicabilidad al primero.

2.1.3. Métodos de explicabilidad: agnósticos y específicos del modelo

Los métodos de explicabilidad pueden ser divididos en dos grupos; los métodos agnósticos del modelo y los métodos específicos del modelo.

Los métodos agnósticos del modelo hacen referencia a aquellos métodos que son capaces de producir una explicación independientemente del modelo que se esté utilizando, de forma que queda separada la 'capa' de explicaciones de la 'capa' de modelos de ML, estableciéndose la primera un nivel de abstracción por encima. En cambio, los métodos específicos del modelo, como su propio nombre indica, son modelos dependientes del modelo que se esté usando.

Separar las explicaciones de los modelos de ML tiene algunas ventajas [52]. La evidente ventaja principal de este tipo de métodos respecto a los específicos del modelo es su flexibilidad, ya que proporcionan a los desarrolladores de ML libertad para utilizar cualquier modelo de ML que crean conveniente, pues son aplicables a cualquier modelo.

Es habitual cuando se está trabajando con ML para la resolución de una tarea, que se evalúen muchos tipos de modelos, y a la hora de compararlos en términos de interpretabilidad, es más sencillo trabajar con explicaciones agnósticas al modelo, porque se puede utilizar el mismo método para todos ellos como se ha comentado.

Una alternativa a los métodos de explicabilidad agnóstica del modelo es tener en cuenta únicamente modelos que ya sean interpretables, aunque esta opción tiene el in-

conveniente de que se pierde capacidad predictiva al limitar la posibilidad de elegir otros modelos de ML [41].

Los aspectos deseables de un sistema de explicación agnóstica de modelos son [52]:

- **Flexibilidad del modelo**: que el método de interpretación pueda funcionar con cualquier modelo de ML, como *Random Forest* o redes neuronales.
- **Flexibilidad de la explicación**: que no exista una limitación en cuanto a la forma de la explicación, puesto que en algunos casos puede ser útil una fórmula lineal, en otros, un gráfico con los valores de importancia de las características, etc.
- **Flexibilidad de representación**: que el método de explicación sea capaz de utilizar una representación de las características diferente a la del modelo que se explica.

2.1.4. Explicaciones locales vs. globales

Las técnicas de interpretabilidad agnósticas del modelo permiten interpretar los modelos de ML de manera global o local.

La interpretabilidad global proporciona explicaciones sobre el comportamiento general del modelo para todas las instancias y explica "qué variables desempeñan un papel importante en la construcción del modelo describiendo el impacto de cada característica en las predicciones que realiza de forma global" [26].

La interpretabilidad local, sin embargo, proporciona explicaciones para una predicción concreta producida por el modelo, y sus técnicas asumen que las predicciones de los modelos de ML sobre una instancia en particular "pueden ser aproximadas por un modelo interpretable 'caja blanca' " [26]. Este modelo local no funciona bien globalmente, aunque consigue aproximar el comportamiento del modelo entrenado dentro una pequeña región local alrededor de la instancia que se esté evaluando. Como se verá más adelante, LIME y el cálculo de los valores de Shapley son ejemplos de técnicas de explicación local.

2.1.5. Importancia en el ámbito de la medicina

Los modelos de árboles son bien conocidos en el ámbito de la salud, puesto que es una forma común y extendida de representar el conocimiento para la toma de decisiones en el diagnóstico y tratamiento médico por tratarse de modelos interpretables.

Explicar las predicciones producidas por modelos basados en árboles se ha vuelto una necesidad dentro de las aplicaciones médicas, donde son más importantes los patrones que un modelo es capaz de descubrir que el rendimiento que consiga un modelo entrenado para realizar una predicción [58][37]. A pesar de todo, la aplicación de la explicabilidad en este campo no es todavía tan efectiva como podría ser debido a que las explicaciones producidas por las técnicas implicadas no son del todo comprensibles ni

poseen fuertes bases matemáticas (a excepción de SHAP), y aun cumpliéndose lo anterior, sigue siendo complicado lograr entender el mecanismo interno que estos modelos de explicabilidad utilizan para determinar cómo y por qué se ha obtenido un resultado. [30]

2.2. Árboles y aprendizaje

A continuación, serán explicados los modelos de ML basados en árboles en los que este trabajo pone énfasis.

2.2.1. Decision Tree

Un árbol de decisión [29] es un conjunto de condiciones organizadas jerárquicamente, representadas por nodos, de forma que siguiendo estas condiciones establecidas desde el nodo raíz del árbol hasta alguno de sus nodos hoja, se puede determinar la decisión final que se ha tomado sobre un conjunto de datos de entrada. Estos árboles de decisión tienen multitud de aplicaciones en la representación de métodos de distinta índole, como médicos, legales, comerciales, matemáticos, lógicos entre otros.

Existen diferentes algoritmos para el aprendizaje de este tipo de estructuras a partir de datos, basándose su entrenamiento en diversos métodos, que funcionan mediante el particionado de la muestra en el caso de un problema de clasificación, determinando de entre varias clases a qué clase pertenece una instancia, o teniendo en cuenta el hecho de que los árboles de decisión se pueden expresar como un conjunto de reglas, aprendiendo por tanto mediante la creación de las mismas.

También hay algoritmos para su uso en problemas de regresión, agrupamiento o estimación de probabilidades [29]. Estos algoritmos se incluyen dentro del conjunto de algoritmos de aprendizaje supervisado.

En cuanto a cómo se construyen a partir de los datos de entrada, de manera general, la raíz de este árbol representa toda la muestra, a partir de la cual se va dividiendo el subespacio de datos en dos o más nodos tras cada división. Llegado un punto durante su construcción, el árbol dejará de dividirse en más nodos y los llamados nodos hoja, los nodos que no se dividen en otros y que se encuentran al final del árbol, se corresponden con la clasificación del problema.

La siguiente fórmula [41] describe la relación entre la variable de salida y y las características x:

$$\hat{y} = \hat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\} \tag{2.1}$$

Cada instancia cae exactamente en un nodo de hoja. $I_{\{x \in R_m\}}$ es la función de identidad que devuelve 1 si x está en el subconjunto R_m y 0 en caso contrario. Si una instancia

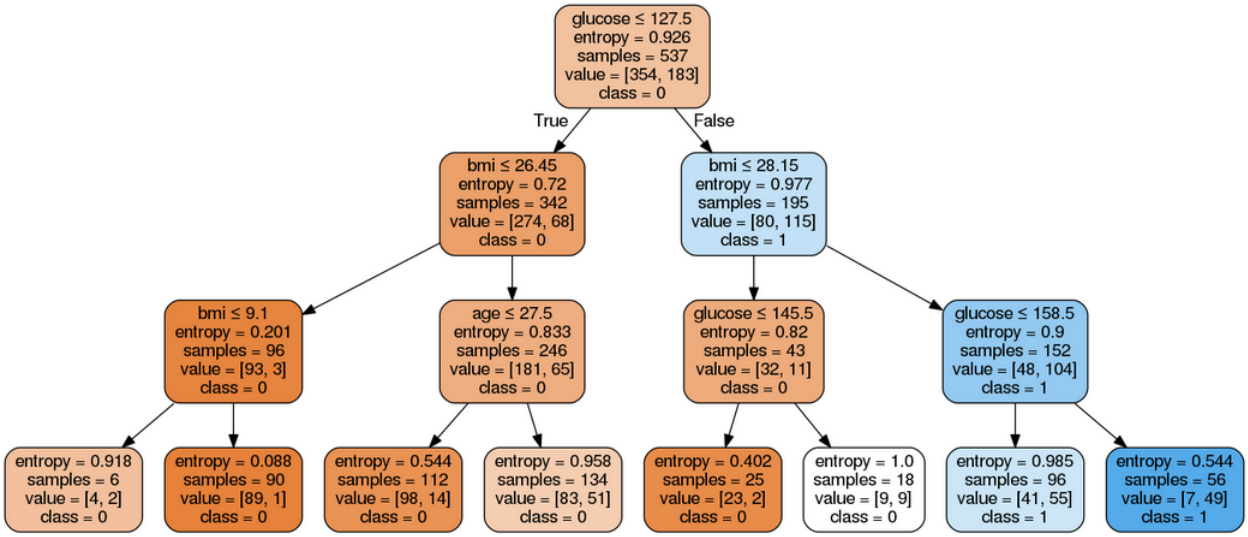

Figura 2.1: *Decision Tree* para el cjto. 'diabetes' [45]

cae en un nodo de hoja R_l, el resultado previsto es $\hat{y} = c_l$, donde c_l es la media de todas las instancias de entrenamiento en el nodo de hoja R_l.

La idea es que para construir el árbol, se toma una característica y se determina qué punto de corte minimiza la varianza de y para un problema de regresión o el índice de Gini o la entropía para problemas de clasificación. La varianza indica la dispersión de los valores de y en un nodo según su valor medio. El índice de Gini o la entropía, entre otras, son medidas que indican la pureza de un nodo. Si todas las clases son igual de frecuentes, el nodo no es nada puro y si sólo está presente una clase, su pureza es máxima [41]. Dado un atributo A, la disminución en impureza promedio (a maximizar) al usar ese atributo en la disgregación viene dada por:

$$i(p_c) - \sum_{i_1}^{m} p_i \times i(p(c|a_i)) \tag{2.2}$$

siendo

- $i(p_c)$ la impureza del nodo padre.
- $i(p(c|a_i))$ la impureza en el correspondiente nodo hijo generado a partir de un test para el valor a_i de A.
- $\sum_{i_1}^{m} p_i \times i(p(c|a_i))$ la impureza generada en los hijos.

Los algoritmos que se aplican en la construcción de los árboles seleccionan la característica para dividirlos que genera la mejor partición en términos de varianza o índice de Gini (u otras medidas) y añaden esta división al árbol en forma de nuevos nodos. La búsqueda continúa siguiendo el mismo proceso y se realizan divisiones de forma recursiva en los nuevos nodos hasta que se alcanza un criterio de parada, que depende de la implementación de cada algoritmo. Estos criterios pueden ser, por ejemplo, que un número mínimo de instancias tienen que estar en un nodo antes de la división, o que un número mínimo de instancias tienen que estar en un nodo terminal.

Como se ha mencionado, existen diferentes algoritmos para la construcción del árbol, y se diferencian fundamentalmente en la forma del árbol (como el número de divisiones por nodo), el criterio de división, el criterio de parada y cómo estiman los modelos simples dentro de los nodos hoja [41]. CART [6] es el algoritmo más popular dentro de los árboles de decisión y fue además uno de los primeros algoritmos de aprendizaje de este tipo, que puede actuar tanto de clasificador como de árbol de regresión.

2.2.2. Random Forest

Random Forest [15] es un algoritmo de aprendizaje supervisado, que utiliza el método conocido en ML como *bagging*, que se basa en combinar modelos de aprendizaje para conseguir mejorar los resultados globales que se obtendrían utilizando esos modelos de forma individual. Estos modelos, en el caso de *Random Forest*, son los árboles de decisión. La idea intuitiva es que este algoritmo entrena un número de árboles determinado de manera separada y la salida que produce en conjunto es la clase que más se ha predecido de entre todas las clases posibles de la variable objetivo en el caso de los problemas de clasificación, y la media de los valores en el caso de los problemas de regresión.

El algoritmo [16] utiliza el concepto de *aleatoridad de características*, es decir, genera un subconjunto aleatorio de características de los datos de entrada para asegurar así una correlación baja entre los árboles de decisión, y son entrenados individualmente. Recordemos del apartado anterior que un único árbol de decisión tendría en cuenta todas las características en el momento de generar una partición de los datos en cada nodo.

Utilizar varios árboles de decisión para producir una predicción tiene la ventaja con respecto a los árboles de decisión individuales de que previene el sobreentrenamiento, ya que cuando se tiene un número suficiente de estos árboles, el modelo no se va a ajustar excesivamente a los datos del conjunto de entrenamiento, puesto que el valor promedio de las predicciones de los árboles (que no están correlacionados como se ha mencionado) reduce la varianza global y el error de predicción del modelo [16].

Por contra, presenta el problema de que las predicciones que generan son menos interpretables que las que produce un único árbol [16], como es lógico. También es un modelo que requiere de más recursos para ser entrenado, tanto en términos de memoria como de tiempo de ejecución.

2.2.3. Gradient Boosting Trees

Gradient Boosting Trees [8] es otro algoritmo de aprendizaje supervisado basado en árboles que se caracteriza por tener los siguientes elementos; una función de pérdida, que depende del tipo de problema a resolver y que debe ser diferenciable, un 'aprendiz débil', que no es más que un modelo con un rendimiento, como poco, mejor que otro que dependa totalmente del azar y además, este aprendiz debe ser aditivo, de forma que se puedan ir sumando aprendices para conseguir minimizar la función de pérdida.

El 'aprendiz débil' que se utiliza en *Gradient Boosting Trees* es el árbol de decisión de regresión, cuyas salidas son aditivas, permitiendo así agregar las salidas de modelos posteriores para mejorar el error de las predicciones. Para mejorar este error se utiliza lo que se conoce en ML como 'descenso de gradiente', en el que los árboles que se van añadiendo al modelo reducen la pérdida residual, siguiendo este descenso de gradiente de error [8].

La idea es que la salida de los árboles que se van añadiendo a la secuencia de árboles existentes en el modelo previamente tratan de corregir o mejorar la salida final del modelo en su conjunto. El entrenamiento para cuando la función de pérdida devuelve un valor aceptable para el problema que se quiera resolver o cuando deja de mejorar con el conjunto de datos aunque se vayan añadiendo nuevos árboles.

Al tratarse de un algoritmo voraz, el sobreentrenamiento es uno de los problemas que puede presentar y es por ello que existen mejoras para tratar de minimizar su impacto [12] [23], que limitan el número de árboles, la profundidad del árbol, la mejora mínima de la función de pérdida entre otras restricciones, o tratan de asignar un peso a la salida de cada árbol y eliminar la correlación entre ellos mediante procesos estocásticos.

2.3. Explicabilidad en árboles

¿Por qué solamente se tienen en cuenta en este trabajo modelos basados en árboles? La respuesta es que, aunque los modelos de Deep Learning (DL) "son más apropiados en campos como el reconocimiento de imágenes, del habla y el procesamiento del lenguaje natural, los modelos basados en árboles los superan de forma sistemática cuando tratan con conjuntos de datos tabulares, donde las características no tienen una estructura temporal o espacial" [36] [12] como es habitual en el campo médico (campo en el que el trabajo está orientado), al margen de problemas que traten con fotogramas de cualquier tipo.

Lundberg y su equipo de investigación, en su artículo donde se presenta Tree SHAP [36], se examinan tres conjuntos de datos médicos tabulares, y el modelo *Gradient Boosting Trees* supera tanto a DL como a la regresión lineal en los tres casos que se proponen. Además, se trata de un tipo de modelo computacionalmente eficiente, fácil de usar y que arroja gran precisión en sus predicciones.

En cuanto a interpretabilidad, para calcular las medidas globales de importancia de las características en los modelos de *ensembles* de árboles, se podría pensar en reportar los nodos sobre los que se ha hecho una partición de los datos, pero esta idea no es práctica ni manejable cuando se dispone de conjuntos de muchos árboles o de árboles con muchas particiones sobre distintas características. Es por esto que, aunque los algoritmos de árboles parezcan interpretables en un principio debido a la estructura que poseen, necesitan de un modelo externo que produzca explicaciones sobre las predicciones que realizan.

En un estudio anterior realizado por el mismo grupo de investigación [34], se unificaron diferentes métodos de atribución de importancia aditiva de características [32]

[60] [33] [61] [14] en una única clase de métodos, y los resultados de Teoría de Juegos cooperativos muestra, que para estos métodos, solamente existe un enfoque de explicación óptimo, los llamados valores de Shapley, puesto que cumplen ciertas propiedades deseables en IA explicable [56] [57].

La dificultad surge a la hora de computar estos valores, que es un problema *NP-hard* y su tiempo de ejecución crece de manera exponencial en función del número de características de entrada que reciba el modelo. IME [61] o QII [14] son métodos que realizan este cálculo de manera aproximada para paliar el problema del tiempo de ejecución excesivo que requiere su cómputo. Lo mismo ocurre con LIME u otros, técnicas que son más rápidas que realizar el cálculo exacto de estos valores (hasta la aparición de Tree SHAP), a cambio de ser menos precisas y consistentes.

Este trabajo se centra en Tree SHAP de SHAP [36], ya que para modelos basados en árboles de decisión, es capaz de computar los valores de Shapley de forma exacta en tiempo polinomial de bajo orden, dejando de pertenecer el problema a la clase *NP-hard*, por lo que mejora cualquier otro método desarrollado hasta el momento, asentándose sobre sólidas bases matemáticas y garantías (por las propiedades que cumple) que el resto no posee y que dotan de fiabilidad y confianza a la técnica.

2.3.1. Modelos de explicación agnósticos

2.3.1.1. IME

IME [61] es uno de los primeros métodos para explicar las predicciones de un modelo para una instancia que utiliza el enfoque de asignar a cada característica un número real como indicador de la contribución de esa característica a una predicción, proporcional a la influencia del valor. Se basa en la idea de observar cómo cambia la predicción para cada subconjunto del *conjunto potencia* que forman las características.

Sin entrar en la formalización del método, el cómputo de valores que este método propone conlleva una complejidad de tiempo exponencial y es una aproximación muy parecida al cálculo de los valores de Shapley de Teoría de Juegos que será presentado más adelante.

Este método fue uno de los primeros intentos de crear una técnica que lograra dotar de explicabilidad a las predicciones producidas por modelos de ML, y aunque la técnica es muy poco eficiente y prácticamente inservible fuera del ámbito académico e investigador, consiguió resultados antes no obtenidos y marcó en cierta manera el camino a seguir en el futuro dentro de la investigación en este campo.

2.3.1.2. LIME

El método LIME [52] trata a los modelos como una caja negra para comprobar los cambios que se producen en las predicciones que genera cuando se perturban los datos de entrada. Cuanto más grandes sean los cambios en la salida producidos por la

alteración del valor de una característica, más importante es una característica para la predicción.

Para conseguir esto se entrena un nuevo modelo interpretable (como por ejemplo, un árbol de decisión) con los nuevos datos y sus predicciones respectivas realizadas con el primer modelo, ponderado en función de la "proximidad de las instancias muestreadas a la instancia de interés" [41].

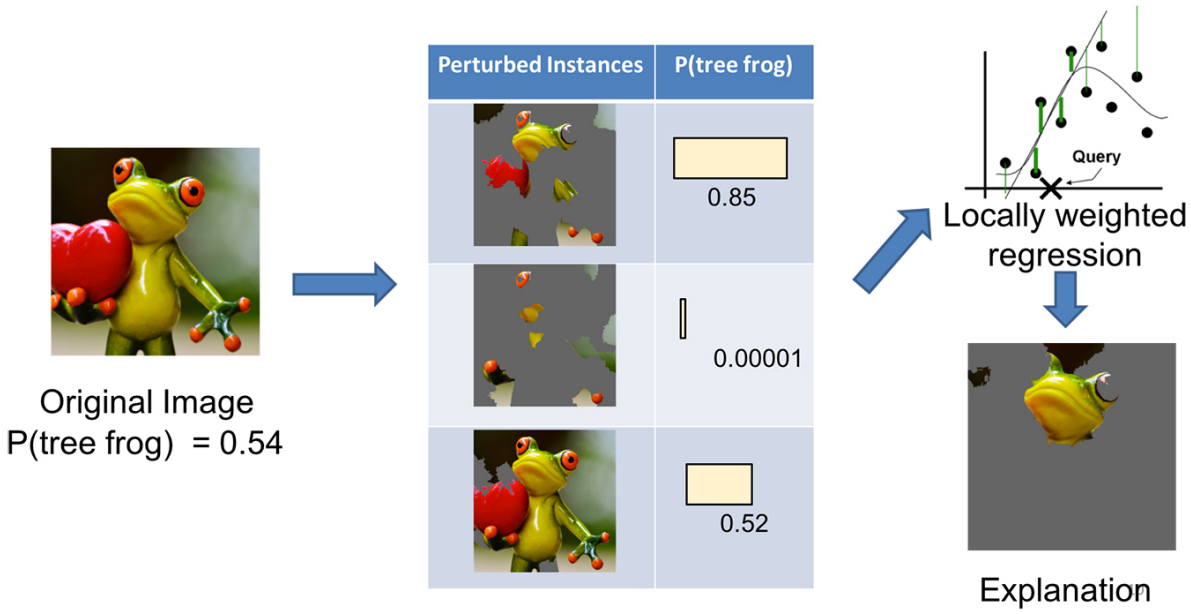

Figura 2.2: Explicando una predicción con LIME [53][51]
Pixabay

Este nuevo modelo resultante es una aproximación local de las predicciones, por lo que sería necesario entrenar más modelos para aproximar todas las predicciones de manera global.

El tipo de modelo que se ha descrito puede ser expresado matemáticamente de la siguiente manera [41]:

$$explanation(x) = arg\ min_{g \in G}\ L(f, g, \pi_x) + \Omega(g) \tag{2.3}$$

Que indica que "el modelo para la instancia x es el modelo g que minimiza la pérdida L, que podría ser el error cuadrático medio o cualquier otra medida de error, que mide lo cerca que está la explicación de la predicción del modelo original f, mientras que la complejidad del modelo $\Omega(g)$ se mantiene baja. G es la familia de posibles explicaciones. La medida de proximidad π_x define cómo de grande es el vecindario alrededor de la instancia x que se considera para la explicación" [41].

2.3.1.3. MAPLE

MAPLE [49] surge como un método para mejorar y hacer frente al popular LIME, "combinando la técnica SILO [5], que propone la idea de crear modelos lineales locales

utilizando *Random Forest* como método de selección supervisada de vecindarios alrededor de una instancia y DStump como método de selección de características".

La diferencia con respecto a LIME radica en el hecho de que el comportamiento de las explicaciones locales en torno a los patrones globales es diferente porque MAPLE utiliza un método de selección de vecindario supervisado (SILO), al contrario que LIME, siendo MAPLE más preciso en las explicaciones que produce según experimentos realizados [48].

Esto quiere decir, de forma sencilla, que la selección de instancias próximas a la instancia de interés para aproximar las predicciones construyendo modelos auxiliares, se realiza de tal manera que MAPLE tiene más en cuenta el patrón global que describe el conjunto de datos aproximando las predicciones, por tanto, de forma más precisa, consiguiendo explicaciones más significativas.

2.3.2. Teoría de Juegos

2.3.2.1. Juegos cooperativos

En Teoría de Juegos, se conoce como juego cooperativo aquel juego en el que dos o más jugadores cooperan entre ellos para lograr un mismo objetivo, ganando o perdiendo en conjunto como un equipo en lugar de competir entre ellos [65].

De manera formal, se representan mediante funciones características y vienen dados por el par $\Gamma = (N, \nu)$ donde N es un conjunto finito no vacío de jugadores y ν es una función característica tal que $\nu : 2^N \rightarrow \mathbb{R}$ y que asigna a cada coalición de jugadores dentro de N un valor real que indica la ganancia colectiva que puede obtener ese conjunto si se agruparan sus jugadores [10].

2.3.2.2. Valor de Shapley

El valor de Shapley surge de un método de distribución de beneficios en la Teoría de Juegos cooperativos. "Para cada juego cooperativo se asigna un único reparto (entre los jugadores) del beneficio total generado por la coalición de todos los jugadores" [27].

Supongamos una coalición de jugadores que cooperan, y cada uno obtiene una ganancia por participar en esa cooperación. La idea intuitiva es que el valor de Shapley ofrece una respuesta a la pregunta de cuál es la importancia que tiene cada jugador a la hora de cooperar, y qué recompensa justa puede esperar, sabiendo que algunos jugadores pueden contribuir más a la coalición que otros [27].

Matemáticamente, el valor de Shapley se define como una función característica *val* de jugadores en S. Este valor, asociado a una característica, es su contribución a la ganancia total, ponderada y sumada con todas las posibles combinaciones de los valores de las características que se contemplan. En otras palabras, es la contribución marginal media de un valor de una característica en todas las posibles coaliciones [41]:

$$\phi_i(val) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_i\}} \frac{|S|!(p-|S|-1)!}{p!}(val(S \cup \{x_i\}) - val(S)) \tag{2.4}$$

donde S es un subconjunto de características utilizadas en el modelo, x es el vector de valores de características de la instancia que se quiere explicar y p el número de características. $val_x(S)$ es la predicción para los valores de las características en el conjunto S que son marginales sobre las características que no están incluidas en el conjunto S:

$$val_x(S) = \int \hat{f}(x_1, \ldots, x_p) d\mathbb{P}_{x \notin S} - E_X(\hat{f}(X)) \tag{2.5}$$

Existe una fórmula alternativa equivalente para el valor de Shapley, que viene dada por la siguiente expresión y es a la que hace referencia el artículo de Lundberg [36]:

$$\phi_i(val) = \frac{1}{|N|!} \sum_R [val(P_i^R \cup \{i\}) - val(P_i^R)] \tag{2.6}$$

Donde $|N|$ es en este caso el número de características del modelo, R es el conjunto de todas las posibles ordenaciones de las características y P_i^R es el conjunto de jugadores en N que van antes de i en el orden R.

Estos conceptos se relacionan con las predicciones en ML y la explicabilidad si se entiende que el "juego" es el problema de predecir una instancia en un conjunto de datos, la "ganancia" es el valor predecido real de la instancia menos la media de los valores de las predicciones del resto de instancias y los "jugadores", que son las características de la propia instancia que cooperan para obtener la "ganancia" mencionada [41].

2.3.2.3. SHAP (SHapley Additive exPlanations)

SHAP (SHapley Additive exPlanations), creado por Lundberg y Lee [34] es un método que utiliza los valores Shapley de Teoría de Juegos para explicar una predicción de una instancia cualquiera, producida por un modelo de ML, calculando la contribución de cada característica de los datos de entrada a esta predicción.

El avance que aporta este método con respecto a técnicas anteriores es que la explicación producida por los valores de Shapley se representa como un método de atribución de características aditivo, como un modelo lineal, conectando así LIME con los valores de Shapley.

Estos valores SHAP son aproximaciones de los valores de Shapley, puesto que su cálculo exacto es computacionalmente costoso. De entre las técnicas propuestas [34] para abordar el cálculo de esta contribución que aporta cada característica, destaca Kernel SHAP en primer lugar, como método agnóstico del modelo.

Kernel SHAP consiste, sin entrar en su formalización, en muestrear todas coaliciones posibles en función de las características de entrada, obtener una predicción para cada

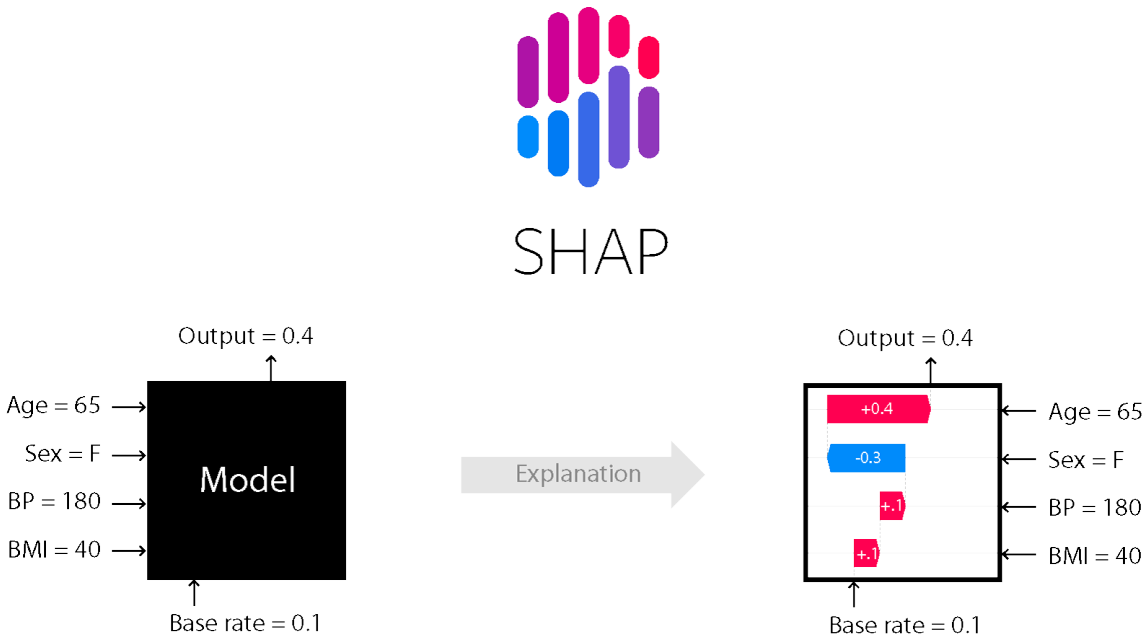

Figura 2.3: Explicación de modelos "caja negra" con SHAP [35]

coalición aplicando el modelo entrenado para calcular un peso y ajustar un modelo lineal en función de estos pesos para, finalmente, devolver los valores SHAP calculados [42].

La gran diferencia que guarda con respecto a LIME es la ponderación de las instancias en el modelo de regresión [42]; mientras que en LIME se ponderan las instancias en función de su proximidad con la instancia original, SHAP las pondera según el peso que cada coalición obtendría en la estimación que realiza de su valor de Shapley.

Aunque Kernel SHAP mejora la eficiencia del cálculo de los valores de Shapley, independientemente del modelo, si se restringe la atención hacia modelos concretos, se pueden desarrollar métodos de aproximación más rápidos y específicos del modelo, como Linear SHAP, para modelos lineales, Low-Order SHAP, que mejora la complejidad para regresiones lineales, Max SHAP, Deep SHAP, para modelos de DL [34] y más recientemente, Tree SHAP para modelos formados por árboles de decisión.

2.3.2.4. Tree SHAP

Tree SHAP [36] es un nuevo método de explicabilidad local para modelos basados en árboles que consigue calcular explicaciones locales óptimas para modelos formados por árboles a través de los valores de Shapley, siendo la única forma de medir la importancia de cada característica de un modelo manteniendo varias propiedades que vienen de la Teoría de los Juegos cooperativos [34]. Se trata, por tanto, de una variante de SHAP. La genialidad detrás de este método es que es capaz de computar de forma exacta los valores de Shapley en tiempo polinomial como se ha comentando en apartados anteriores, por lo que no es un método que proponga otra aproximación diferente de estos valores.

Además, da un paso más allá en campo de la explicabilidad en ML, porque los métodos que existían antes de Tree SHAP que producían explicaciones locales que asignaban

un valor a cada característica de entrada no podían representar los efectos de la interacción que podía existir entre características de forma directa, pero Tree SHAP permite medir estos efectos de interacción local mediante una generalización de los valores de Shapley [25].

También presenta una manera de explicar la estructura global del modelo obteniendo una representación más precisa que los métodos ya existentes valiéndose de muchas explicaciones locales y conservando al mismo tiempo la fidelidad local al modelo original.

A pesar de todos los beneficios que trae la técnica, el algoritmo Tree SHAP puede producir atribuciones a las características poco intuitivas ya que cambia la función característica de la ecuación de los valores de Shapley clásicos (la esperanza marginal) por la esperanza condicional, lo que provoca que las características que no tienen importancia en una predicción podrían obtener un valor distinto de cero [43].

CAPÍTULO 3

Objetivos y metodologías

3.1. Objetivos

3.1.1. Objetivo principal

El objetivo principal de este trabajo es proporcionar una herramienta que facilite la interpretación de explicaciones sobre predicciones realizadas por modelos de árboles de decisión obtenidos mediante ML, por parte de usuarios no expertos en técnicas matemáticas formales.

En concreto, esta herramienta se centrará en explicar la técnica Tree SHAP[36], por ser a día de hoy el único explicador con una base formal matemática contrastada, y explicando además el origen matemático de los valores que computa para validar la técnica y poder darle fiabilidad.

Esta última propuesta surge del problema de que los explicadores basados en fundamentos fuertemente formales no son trasladables al usuario humano de forma universal y estas explicaciones pueden ser poco intuitivas y generar poca confianza.

La herramienta tiene dos usuarios objetivo:

- Científicos de datos, para facilitar la corroboración de si la herramienta de interpretación utilizada (Tree SHAP) es la más adecuada para el problema.
- Usuario final (médico en este caso), para ayudar a explicar cómo funciona el explicador, favoreciendo la transparencia y compresión del cálculo de los valores de importancia.

En cuanto a subobjetivos propuestos, se proponen los siguientes:

- Estudio e implementación del algoritmo Tree SHAP de TreeExplainer.
- Análisis comparativo entre implementaciones: análisis de caja blanca y caja negra

para comparar el desarrollo del subobjetivo anterior y la verisón desarrollada por Scott M. Lundberg [36].

- Estudio empírico entre valores de Shapley y valores SHAP de Tree SHAP.
- Desarrollo de una herramienta interactiva que permita entender el proceso de cálculo de los valores Tree SHAP.
- Caso de uso dentro del ámbito médico con resistencias antibióticas en Unidad de Cuidados Intensivos.

Queda fuera del alcance de este trabajo desarrollar un nuevo método de explicabilidad local, construir un nuevo modelo que explique Tree SHAP que a su vez explique un árbol de decisión, realizar mejoras significativas sobre los algoritmos propuestos ya existentes o proponer otros nuevos, puesto que estos (y parte de los del trabajo desarrollado) son temas en la frontera de investigación que se están desarrollando durante los años 2019-2021.

3.2. Metodología y herramientas

3.2.1. Lenguaje y entorno de desarrollo

El lenguaje utilizado para llevar a cabo las implementaciones ha sido *Python*, puesto que dispone de un gran abanico de librerías útiles que ofrecen facilidades para el desarrollo de aprendizaje automático y para realizar operaciones sobre conjuntos de datos, además de que es el lenguaje utilizado para la implementación de los métodos de explicación en estudio.

En cuanto al entorno de desarrollo, se ha usado *Visual Studio Code* por su interfaz y sus plugins para *Python* que asisten en la corrección sintáctica del código.

Además, se ha hecho uso de la herramienta *Jupyter* como medio para ilustrar los diferentes ejemplos y casos de uso. La utilización de *Jupyter* facilitará el acceso y uso de la herramienta desarrollada, puesto que es la herramienta utilizada en los artículos científicos para ilustrar y publicar sus resultados, ya que facilita la difusión y reproducibilidad de los experimentos en otros sistemas.

3.2.2. Metodología seguida

La metodología seguida se basa en la lectura y el estudio de la técnica principal sobre la que va este trabajo, junto con los algoritmos que la rodean, para poder realizar su implementación. Hay que resaltar que este trabajo tiene una gran componente teórica y matemática, por lo que gran parte del tiempo empleado en este trabajo fue para pensar y extraer conclusiones sobre las bases en las que se asienta el método.

Es por ello que para entender bien los fundamentos de la técnica, fue necesaria la consulta de diferentes fuentes para profundizar en temas como Teoría de Juegos, valores de Shapley y explicabilidad en general dentro del ML.

Este estudio inspiró la idea para dar una explicación a la salida que producía el explicador mediante los valores de Shapley clásicos con la esperanza condicionada como función característica.

Una vez entendida y aplicada la base formal y matemática para generar una explicación del explicador, el siguiente paso en la metodología seguida se ha basado en realizar una parte de desarrollo, que permitiera la visualización del cálculo llevado a cabo de los valores de Shapley en forma de grafo de coaliciones interactivo.

Finalmente, para la detección y análisis de fallos en el diseño del método, se ha aplicado la técnica sobre una base de datos médica con clases desbalanceadas y con datos correlacionados entre ellos, propiedades que pueden llegar a ser conflictivas con este método.

3.2.3. Selección de librerías y herramientas

Para el manejo general de matrices se ha escogido el paquete *numpy*[1] por su eficiencia tanto en memoria como en tiempo de ejecución de las operaciones matriciales que computa. Existen alternativas, como *TensorFlow*[2] o *PyTorch*[3] pero añaden complejidad adicional al tratamiento de los datos y tiempo de carga de los mismos en GPU. Estas opciones se recomiendan solamente si se necesita una escalabilidad y un rendimiento extremo, como ocurre cuando se entrenan redes neuronales, donde hace falta ejecutar operaciones matriciales muy costosas varias veces. No es el caso del trabajo realizado, ya que se ha trabajado con modelos basados en árboles de decisión como se ha mencionado anteriormente. El resto de elecciones que pueden tenerse en cuenta seriamente implican el uso de otro lenguaje de programación, como *R*, *MATLAB* o *Julia*.

Paralelamente, se ha utilizado el módulo *pandas*[4] para manejar y analizar fácilmente los conjuntos de datos con los que se ha trabajado y se han usado en los algoritmos de aprendizaje automático. La ventaja que tiene esta librería con respecto a otras que ofrecen la misma funcionalidad, como *Dask*[5], *PySpark*[6], *Modin*[7] y *Vaex*[8] es que los tutoriales, la documentación y la comunidad que existe alrededor es mucho más grande por lo que su curva de dificultad no muy pronunciada y se facilita su utilización enormemente. Esto junto a su alto rendimiento en conjuntos de datos no excesivamente grandes motivó su uso.

[1] numpy - `https://numpy.org/`
[2] TensorFlow - `https://www.tensorflow.org/`
[3] PyTorch - `https://pytorch.org/`
[4] pandas - `https://pandas.pydata.org/`
[5] Dask - `https://dask.org/`
[6] PySpark - `https://spark.apache.org/docs/latest/api/python/`
[7] Modin - `https://modin.readthedocs.io/en/latest/`
[8] Vaex - `https://vaex.io/`

Siguiendo con ML, *scikit-learn*[9] es la librería por excelencia en *Python* y dispone de todos los modelos de ML basados en árboles con los que se ha trabajado, Decision Tree, Random Forest y Gradient Boosting Trees, listos para usar con una sencilla configuración. Opciones alternativas dentro de las librerías de ML, como *Keras*[10], ofrecen una interfaz a más bajo nivel para poder alterar la implementación de los algoritmos y hacer que los entrenamientos sean más eficientes, sobre todo de *Deep Learning*. Este trabajo no se centra en *Deep Learning* y el ajuste fino de hiperparámetros de los modelos no es algo prioritario, es por eso que se ha optado por *scikit-learn*.

En cuanto a las librerías de explicabilidad, para el método IME no existe ninguna implementación publicada. Para LIME existe la librería *lime*[11], que soporta la explicación de predicciones individuales para clasificadores de texto o clasificadores que actúan sobre matrices *numpy* de datos numéricos o categóricos, además de imágenes. En lo referente a SHAP, que es el método en el que se centra el trabajo, se tiene el paquete *shap*[12], que puede explicar el resultado de cualquier modelo de ML, tiene implementado un algoritmo muy rápido y exacto para los modelos de *ensembles* de árboles, además de soportar modelos para el lenguaje natural y DL. Estas implementaciones se siguen desarrollando y son únicas en el sentido de que no tienen un competidor directo, así que son una referencia cuando se quieren aplicar estos métodos de explicabilidad.

Implementaciones de los valores Shapley en *Python* no existen realmente (aunque en *R* sí [67]) porque su cálculo es muy costoso y hay métodos que son buenas aproximaciones que se han desarrollado para realizar este cálculo de forma mucho más eficiente. Es por este motivo que he desarrollado una implementación propia para poder utilizarla.

Para la parte de desarrollo, se ha hecho uso de la librería *Networkx*[13], la herramienta más potente y útil para el manejo y visualización de grafos en *Python*. Existen alternativas pero gozan de bastante menos popularidad y escasa documentación como *neo4j*[14] o *RedisGraph*[15]. El resto de herramientas de este tipo está más orientado a un uso dentro del Big Data y trabajan con bases de datos de gran tamaño. Para la parte visual de estos grafos fue necesaria la inclusión de *matplotlib*[16] y *pylab* (de *matplotlib*), que soportan visualizaciones en 2D muy personalizables, fáciles de implementar, y lo más importante, son perfectamente compatibles con *Networkx*.

El resto de módulos incluidos dentro del código son nativos del lenguaje y de uso general. Pueden ser consultados en la librería estándar de *Python* [66].

Finalmente, para la creación de figuras y diagramas ilustrativos sobre los procesos seguidos en el diseño y resolución del proyecto que sirvan como apoyo a mis explicaciones he utilizado la herramienta *diagrams*[17].

[9] scikit-learn - `https://scikit-learn.org/stable/`
[10] Keras - `https://keras.io/`
[11] lime - `https://github.com/marcotcr/lime`
[12] shap - `https://shap.readthedocs.io/en/latest/index.html`
[13] Networkx - `https://networkx.org/`
[14] neo4j - `https://neo4j.com/`
[15] RedisGraph - `https://oss.redislabs.com/redisgraph/`
[16] matplotlib - `https://matplotlib.org/`
[17] diagrams - `https://app.diagrams.net/`

CAPÍTULO 4

Diseño y resolución

4.1. Selección de técnicas

De entre las técnicas seleccionadas para desarrollar el trabajo, destaca en primer lugar SHAP. En concreto, el algoritmo Tree SHAP, por poseer ciertas características y propiedades que ya han sido explicadas anteriormente y que favorecen su uso en el ámbito médico y en la asistencia en la toma de decisiones clínicas haciendo uso de ML. La utilización de este algoritmo se justifica también si se tiene en cuenta que se quería desarrollar un explicador que produjera explicaciones sobre la salida que genera esta misma técnica, por lo que resultaba más que conveniente tener una implementación propia para poder comparar y jugar con ella alterándola a mi antojo con el fin de obtener ideas para este explicador del explicador.

Esta investigación y estudio de la técnica me llevó a la raíz del algoritmo; los valores de Shapley y el cálculo de la esperanza matemática condicionada sobre árboles de decisión, que son las dos técnicas restantes sobre las que se asienta este trabajo. La justificación de por qué usar estas técnicas y no otras, o por qué no se utilizó el propio algoritmo Tree SHAP para el explicador del explicador viene más adelante.

Cabe destacar también que para la aplicación de la herramienta desarrollada a un caso del mundo real, se ha trabajado con un modelo que presentaba varias dificultades, en particular, estaba compuesto de características tanto numéricas como no numéricas, y me parece interesante mencionar que para solventar este problema en particular he hecho uso de la técnica 'One-Hot Encoding' por ser la única forma posible de realizar divisiones arbitrarias sobre un nodo de un árbol de decisión que no depende del ordenamiento de estos valores no numéricos, aunque añade complejidad al modelo [46] y por lo tanto eleva el coste computacional, pero para el ejemplo sirve y el impacto en el rendimiento es imperceptible.

4.2. Diseño e implementación

El diseño de la arquitectura para el desarrollo de la herramienta propuesta es el que se puede ver en la figura 4.1.

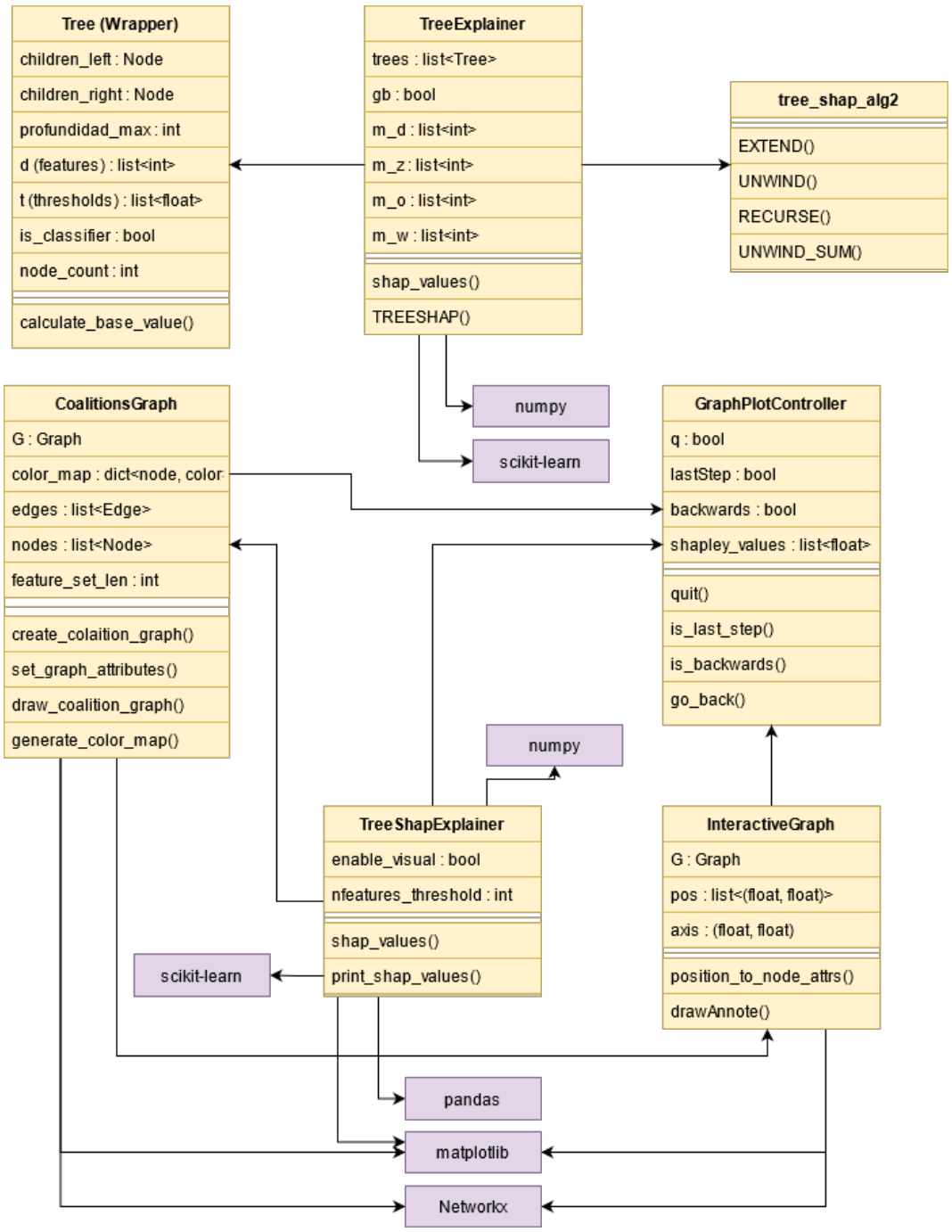

Figura 4.1: Diagrama de clases de la herramienta desarrollada.

Consta de dos módulos separados; la implementación del algoritmo Tree SHAP, fragmentado en dos clases para separar las funciones utilizadas en el algoritmo del propio algoritmo, y por otro lado, el módulo con aquellas clases de la utilidad desarrollada que tienen que ver con la construcción, manejo e interacción del grafo de coaliciones en

el que se basa, además de con el cálculo de valores de Shapley y su traza matemática. Estos módulos no están relacionados ni deben estarlo para poder operar por su cuenta de forma independiente, aunque se pueden poner en común los valores que producen si así se quiere.

No se trata por tanto de una aplicación completamente integrada con soporte para diferentes plataformas (aunque se ha comprobado su funcionamiento en Windows y en Mac), sino de una librería más para *Python*, como puede ser la de SHAP [35], que permite obtener el origen de los cálculos realizados por la implementación original de S. Lundberg y navegar a través de ellos.

4.2.1. Algoritmo Tree SHAP

El primer algoritmo implementado en la realización del trabajo es Tree SHAP. Lundberg, en su artículo [36], establece que este algoritmo realiza el trabajo equivalente a calcular los valores de Shapley aplicando la esperanza condicionada como función característica *val* de la siguiente ecuación que ya ha aparecido previamente, de forma recursiva:

$$\phi_i(val) = \frac{1}{|N|!} \sum_R [val(P_i^R \cup \{i\}) - val(P_i^R)] \tag{4.1}$$

La interpretación que recibe esta expresión para poder entenderla es que los valores que se obtienen de ella se computan atribuyendo el cambio producido en cada cálculo, que provoca la aplicación de la esperanza condicionada de un subconjunto de características (ordenación) en el modelo, a la característica que fue introducida en el paso anterior, y finalmente se realiza un promedio de todos los cambios producidos sobre todas las ordenaciones posibles de las características del conjunto de datos de entrada.

Esta ecuación es la que se tiene en cuenta en este algoritmo ya que satisface tres propiedades fundamentales [36] en Teoría de Juegos cooperativos que garantizan un reparto justo y equitativo de importancias para un conjunto de características determinado con el que haya sido entrenado un modelo:

1. **Localidad y aditividad.**

$$f(x) = \phi_0(f) + \sum_{i=1}^{M} \phi_i(f, x) \tag{4.2}$$

   Esta propiedad indica que la suma de todas las importancias $\phi_i(f, x)$ calculadas coinciden con la salida que produce el modelo $f(x)$. Se trata de una propiedad importante para los agregados de modelos de árboles como *Random Forest*, que permite calcular un promedio de entre todos los valores Shapley que produce cada árbol.

2. **Consistencia y monotonía.** Dados dos modelos f y f', si se da que:

$$f'_x(S) - f'_x(S \setminus i) \geq f_x(S) - f_x(S \setminus i) \tag{4.3}$$

entonces para cada subconjunto de características $S \in F$ (siendo F el conjunto de todas las características que tenga el conjunto de datos de entrada), se tiene que:

$$\phi_i(f', x) \geq \phi_i(f, x) \tag{4.4}$$

Esto quiere decir que si un modelo es alterado de forma que la contribución marginal (la diferencia entre el cálculo de f y f') de una característica aumenta o se mantiene, el valor de Shapley final que se obtiene también aumenta o se mantiene.

3. **Falta (Missingness).** Si se tiene que:

$$f_x(S \cup i) = f_x(S) \tag{4.5}$$

entonces para cada subconjunto de características $S \in F$ (siendo F el conjunto de todas las características que tenga el conjunto de datos de entrada), se tiene que:

$$\phi_i(f, x) = 0 \tag{4.6}$$

De esta forma, se asegura que las características que no tengan ningún impacto sobre una predicción no obtengan un valor de importancia distinto de 0.

Para entender de forma sencilla el algoritmo, la idea es que lleva la cuenta de cada posible tamaño de subconjunto que entra en la ecuación durante la recursión además de un recuento de todos los subconjuntos, que aumentan de tamaño con el método $EXTEND$ a medida que se desciende en el árbol, según una fracción dada de unos y ceros. Estas fracciones indican la proporción de caminos, hasta el momento, en los que la característica actual está en el subconjunto de características que se evalúa en cada llamada recursiva.

Este proceso puede invertirse mediante su método conmutativo $UNWIND$, que se utiliza para deshacer los incrementos de tamaño anteriores en el caso de que ya se haya dividido el árbol por la característica actual, y al final de todo el cálculo para deshacer todos los incrementos producidos mientras se descendía en el árbol y así calcular los pesos de cada característica.

$EXTEND$ lleva también el peso aplicado a esos subconjuntos por la ecuación anterior, que es algo necesario porque el peso aplicado a un subconjunto en la ecuación es diferente cuando se incluye la característica i y se necesita aplicar el método $UNWIND$ nuevamente a cada característica por separado para calcular los valores SHAP de cada una.

Para implementarlo, tuve que aclarar muchos conceptos previos y seguir el pseudocódigo del artículo que lo propone [36], determinando las estructuras de datos adecuadas para cada variable que se sugiere. Se trata de un algoritmo que no se presta a ser comprendido fácilmente, aunque no presenta tanto problema a la hora de implementarlo, ya que los tipos de datos más complejos que forman parte del mismo son las listas y el resto del trabajo es determinar para qué se usa cada variable y qué forma tiene cuando alguna de las funciones la recibe como parámetro de entrada.

4.2.2. Batería de pruebas de comparación de valores

Una vez terminada la implementación de Tree SHAP, había que probarla para estar seguro de que los valores de mi implementación coincidían con los de la de Lundberg.

Para eso, desarrollé un script que tomaba diferentes conjuntos de datos con instancias más que suficientes, propios de la librería SHAP, que suelen ser los típicos para ilustrar ejemplos dentro del ML (Iris, Boston, Adult...). Estos conjuntos de datos cubren los problemas de regresión y clasificación (el cálculo es diferente para un tipo de problema y otro). Además de estos conjuntos de datos, he tenido en cuenta los modelos de ML basados en árboles de decisión que soporta *scikit-learn*; *Decision Tree*, *Random Forest* y *Gradient Boosting Trees*.

Con estos conjuntos de datos y modelos, mi código genera todas las posibles combinaciones y ejecuta ambas versiones de Tree SHAP. Inicialmente, no coincidían del todo, pero fijándome me di cuenta de que se trataba de un error en el redondeo de decimales por el uso de diferentes tipos de datos en los valores númericos, que fue resuelto añadiendo un error ϵ permitido de 0.1 para determinar si dos valores son iguales. Esta diferencia no es atribuible a otro parámetro desconocido puesto que en ese caso los errores serían mucho más evidentes y nada constantes a lo largo de la salida, como ocurre en este caso. El valor de ϵ es arbitrario (dentro de tratarse de un valor de error y por tanto pequeño) y se fue ajustando en función de los resultados obtenidos.

Estas pruebas son suficientes para comprobar la validez de mi implementación ya que comparan decenas de miles de valores de forma satisfactoria sin detectar diferencias, por lo que se puede asegurar que, como poco, para una muestra considerablemente grande de datos, se comporta como es esperado. Cabe mencionar que la implementación original está escrita en *C++* a pesar de distribuirse como una librería de *Python* y está muy optimizada por lo que es bastante más rápida que la que he desarrollado.

4.2.3. Explicador del explicador

Para entender los fundamentos de mi aporte original, 'Tree SHAP Explainer', es necesario introducir ciertas ideas que motivaron este explicador de las explicaciones que Tree SHAP genera. Como se ha mencionado en puntos anteriores, los valores que Tree SHAP computa proceden de Teoría de Juegos. Recordemos que Lundberg, en su artículo [36], establece que este algoritmo realiza el trabajo equivalente a 'calcular los valores de Shapley aplicando la esperanza condicionada' a la función característica *val* de la ecuación 4.1.

Originalmente, los valores de Shapley clásicos operan mediante la esperanza marginal, que se deshace de la limitación que produce la estructura de árbol en modelos de este tipo. La esperanza condicionada sí que tiene en cuenta la estructura de árbol, lo cual puede parecer un factor limitante a priori, aunque la única diferencia que se produce en los valores de Shapley utilizando la esperanza condicionada es el hecho de que características que no tienen impacto en una predicción pueden obtener un valor

distinto de 0. Por tanto, estos valores de Shapley, con la esperanza condicionada como función característica, son exactos excepto en la situación descrita.

¿Y por qué no se tiene en cuenta la esperanza marginal? El cálculo de valores de Shapley con esta función es fundamentalmente teórico y su implementación sería compleja ya que depende de la aleatoriedad de los datos y de su selección para el entrenamiento de muchos modelos, que comprueben cómo afecta la agregación de una característica a conjuntos con diferentes combinaciones de las mismas para calcular una media en función de las predicciones que se obtienen, lo que se conoce como contribución marginal. De hecho, el resto de métodos que realizan un cálculo de estos valores son simplemente (buenas) aproximaciones ya que no tienen en cuenta todas las coaliciones que se pueden formar para hacer menos pesado su cómputo.

Por este motivo y porque el cálculo de valores de Shapley siguiendo la ecuación al pie de la letra es computacionalmente costoso, este tipo de implementaciones no tienen popularidad ni interés fuera de la investigación y de lo puramente académico, por lo que no parecen existir.

Estos datos sirven para ilustrar el proceso mental que seguí para tomar la decisión de implementar, de forma original puesto que no parece existir código publicado y disponible para su uso, el cálculo de los valores de Shapley, utilizando la esperanza condicionada como función característica.

La situación era la siguiente: se quería explicar de alguna manera las explicaciones que producía Tree SHAP. Se disponía de la implementación de este algoritmo para, en principio, modificarlo y usarlo para este fin. Por último, se tienen las bases sobre las que se asienta el cálculo. Con esto sobre la mesa, se concluyó que utilizar Tree SHAP para explicarlo a sí mismo es imposible porque tiene en cuenta 2^M conjuntos de características, siendo M el número de características de un conjunto de datos, cuyas contribuciones marginales son calculadas simultáneamente y de forma recursiva. Este hecho junto con la complejidad inherente del algoritmo hace que la traza, por tanto, fuera prácticamente irrastreable y aunque no lo fuera, la explicación no sería nada legible, por lo que necesitaba un enfoque diferente.

Este enfoque se basa en la implementación y el uso del algoritmo que calcula la esperanza condicionada de un conjunto S de características, que se menciona y se muestra en el mismo artículo [36], que tendría que ser aplicada sobre el cálculo de los valores de Shapley y que tenía que implementar también. Si Tree SHAP de verdad se basaba en los fundamentos descritos en el apartado 4.2.1... Los valores producidos deberían ser los mismos y la traza de su cálculo tendría que ser más manejable y útil para su consulta. Para ello me he apoyado muchísimo en el artículo (no científico) “SHAP Part 3: Tree SHAP” [63]

Dentro del mismo, se crea un conjunto de datos artificial como el que se muestra en la figura 4.2, que contiene 10 instancias con 3 características cada una que representan un valor numérico aleatorio, siendo la columna ‘t’ la variable objetivo de este problema de regresión.

El proceso de aprendizaje que sigue un árbol de decisión para este conjunto da como

	1	2	3	t
0	206	108	114	10
1	206	380	390	10
2	206	179	340	10
3	206	153	380	10
4	206	166	243	10
5	194	326	328	20
6	6	299	158	50
7	6	299	237	50
8	6	301	193	30
9	6	301	186	30

Figura 4.2: Conjunto de datos artificial.

resultado el modelo mostrado en la figura 4.3. Los hiperparámetros utilizados son los propuestos en el artículo para poder validar mis resultados fácilmente.

Las coaliciones posibles dado el conjunto de datos, que tiene tres características, *1*, *2* y *3*, son todas las posibles ordenaciones que pueden hacerse con las mismas (que son tantas como el factorial del número de características, en este caso 6), es decir, se tienen en consideración los siguientes conjuntos: $\{1,2,3\}$, $\{1,3,2\}$, $\{2,1,3\}$, $\{2,3,1\}$, $\{3,1,2\}$ y $\{3,2,1\}$.

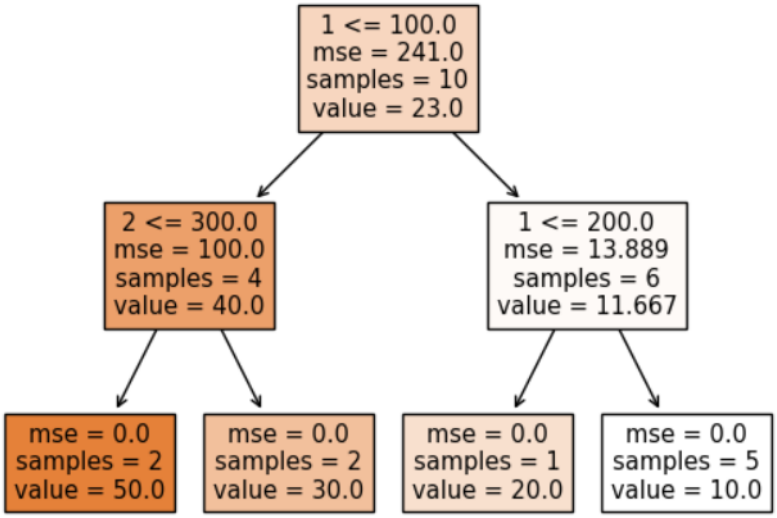

Figura 4.3: Modelo resultante tras entrenar un árbol con el cjto. de datos artificial.

Estos conjuntos con todas las ordenaciones posibles, pueden ser representados mediante un grafo de coaliciones acíclico. Un ejemplo de este grafo, inspirado en el mostrado en el artículo (no científico nuevamente) "SHAP Values Explained Exactly How You Wished Someone Explained to You" [40], puede verse en la figura 4.4, donde cada nodo representa una coalición de características y cada eje la característica añadida nueva con respecto al conjunto anterior.

Esta disposición ayuda a visualizar qué está ocurriendo en la ecuación 4.1; el descenso en el grafo representa la adición de nuevas características a la coalición. La diferencia

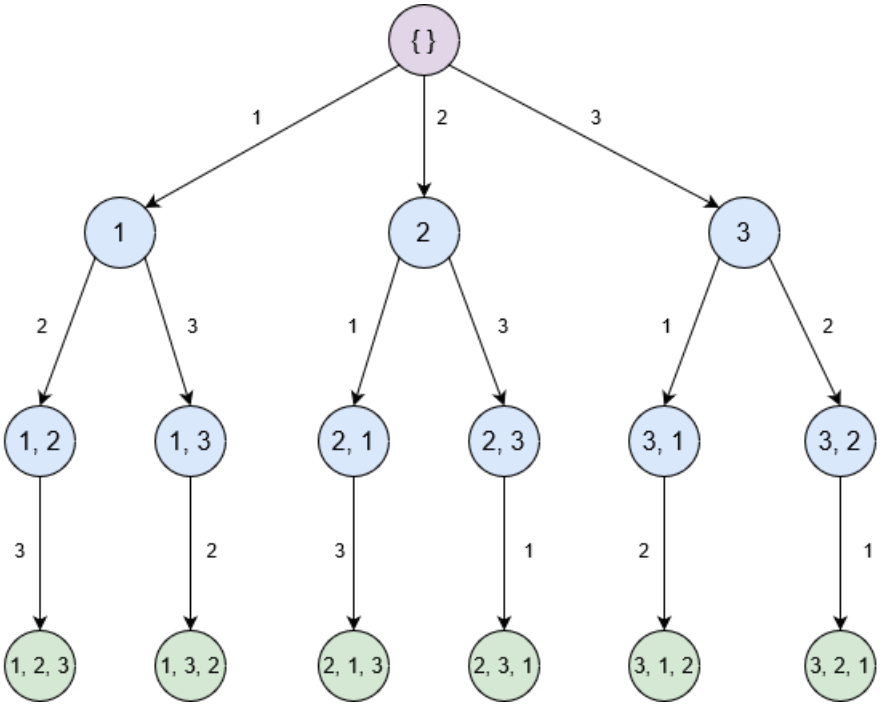

Figura 4.4: Grafo de las coaliciones del cjto. de datos artificial.

entre el valor calculado por la esperanza condicionada con el conjunto en un nodo superior y en su nodo hijo es la contribución marginal de la característica en la arista que une ambos nodos, en ese paso de cálculo. La suma de todas las contribuciones marginales para cada característica, entre el número total de ordenaciones, es su valor de Shapley exacto.

Por ejemplo, consideremos la ordenación $\{1, 2, 3\}$ y una instancia de prueba tal que $[1 = 150, 2 = 75, 3 = 200]$. Inicialmente, el conjunto de partida está vacío. La esperanza condicionada de este conjunto es lo que se conoce como 'valor base', que es la predicción media para el conjunto de entrenamiento, que no es más que la suma del número de instancias que caen sobre cada nodo hoja multiplicado por su valor, dividido entre el total de instancias. Según la figura 4.3, este valor es $(50 \times 2 + 30 \times 2 + 20 \times 1 + 10 \times 5)/10 = 23$.

Para el resto de casos, esta esperanza condicionada, intuitivamente, se calcula siguiendo los nodos del árbol de la figura 4.3 según las condiciones con respecto a la característica añadida. Si la característica está en el nodo, se desciende por la división que haga la condición verdadera con el valor de la característica de la instancia que se defina. Si se encuentra en todos los nodos del camino, este valor es el mismo que el que tenga el nodo hoja. Si no se encuentra en un nodo, se desciende en ambos hijos, sumando los valores finales de ambas ramas, que irán ponderados en función del número de características que caen en cada hijo. Durante el descenso se aplica la misma lógica, según esté la característica del nodo en la coalición o no. Si la característica agregada no aparece en ninguna división del árbol, su valor es 0.

Siguiendo con el ejemplo de cálculo y teniendo lo anterior en mente, se agrega la característica 1 al conjunto (recordemos que tiene un valor de 150). La esperanza condi-

cionada para el conjunto $\{1\}$ tiene un valor de 20. Por tanto, una primera contribución marginal para la característica 1 es $20 - 23 = -3$ según la ecuación 4.1.

A continuación se añade 2 a la coalición, que ahora es $\{1,2\}$. Su esperanza condicionada es 20 nuevamente, pues no afecta a la predicción su inclusión. La primera contribución marginal de 2 es $20 - 20 = 0$.

Finalmente, se agrega 3, completando la ordenación $\{1,2,3\}$ y, como 2, no afecta al valor final de la predicción, así que su primera contribución marginal es de 0. Este proceso se repite con cada ordenación, guardando las contribuciones marginales de cada característica para el cálculo de su valor de Shapley.

Pasando a la herramienta desarrollada, la traza de los cálculos se muestran de una o dos maneras en función de la complejidad del modelo de entrada. Los modelos con un número de características inferior a 5 pueden optar a visualizarse a través de un grafo de coaliciones interactivo, además de obtener un seguimiento completo de los cálculos en forma de texto, que se emite en un fichero de forma separada. Si el modelo es más complejo, no merece la pena visualizar gráficamente los cálculos porque comienzan a ser demasiados, ya que el número de nodos generados crece de forma factorial según el número de características, pero los cálculos pueden ser consultados en forma de texto de igual manera.

El grafo que genera la utilidad, mostrado en la figura 4.5, es interactivo, lo cual quiere decir que es posible avanzar y retroceder en la construcción del grafo tanto como se desee.

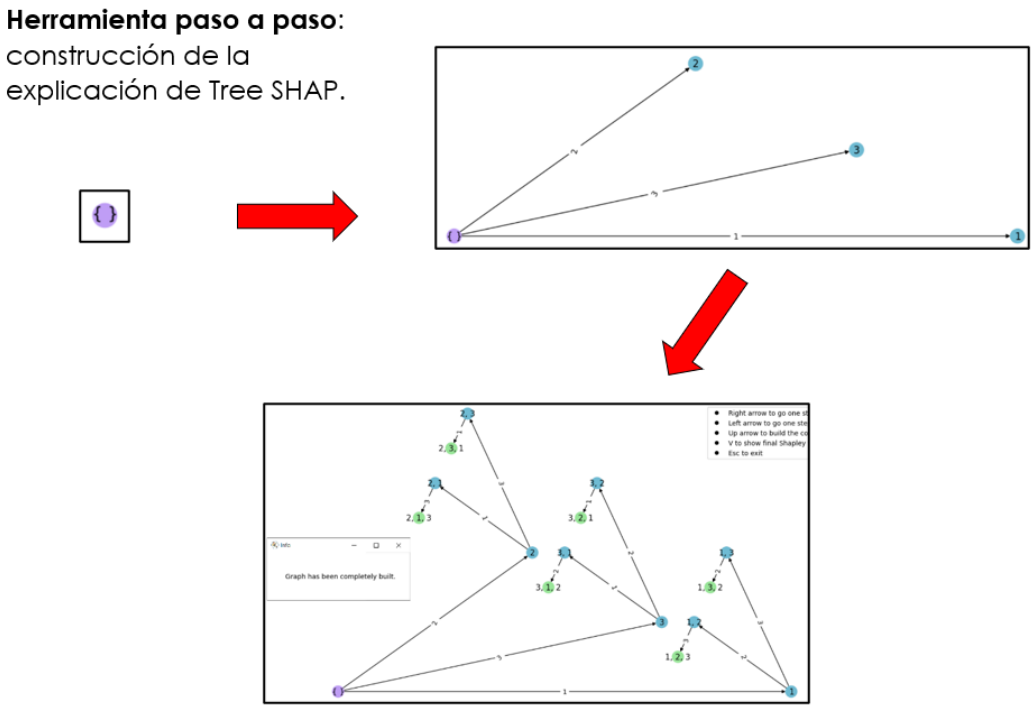

Figura 4.5: Construcción paso a paso del grafo de coaliciones.

También es posible hacer click dentro de los nodos, que representan cada ordenación posible dada por la ecuación 4.1, para consultar el estado de los cálculos de las contribu-

ciones marginales que forman parte del cálculo de los valores Shapley finales, así como construir el grafo al completo rápidamente.

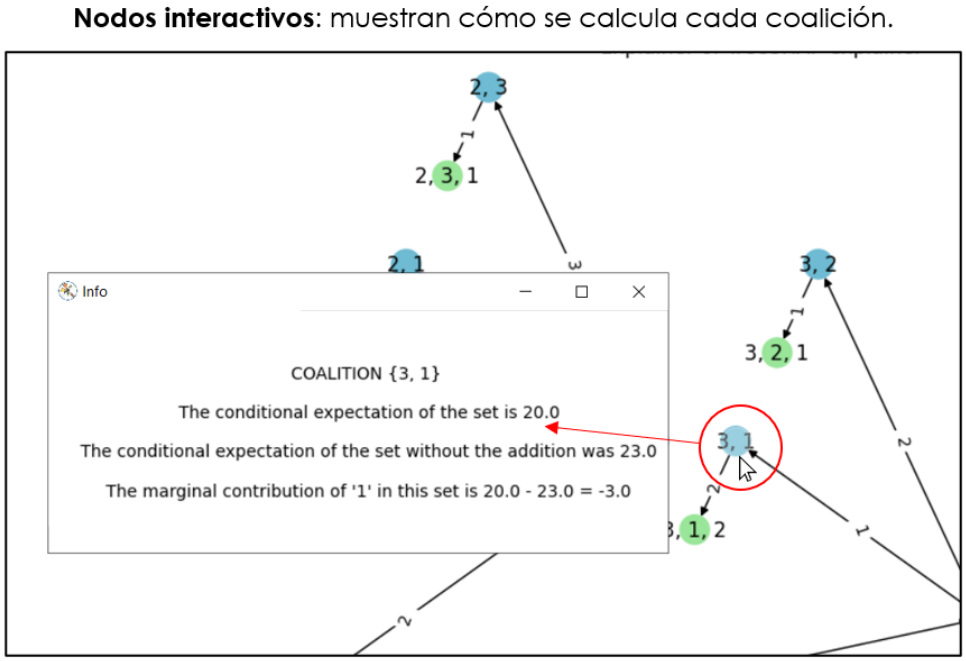

Figura 4.6: Nodos interactivos del grafo de coaliciones.

Otra característica de este grafo que quiero recalcar, es que para facilitar el seguimiento de su construcción a pesar de su disposición en pantalla, el primer nodo y los nodos hoja poseen colores distintos a los nodos internos.

Como ha sido mencionado anteriormente, se trata de un grafo acíclico, con forma de árbol. Este grafo es diferente al que se muestra en el artículo en el que me inspiro, porque en él se está construyendo lo que se conoce en matemáticas como *power set* o conjunto potencia, que vendría asociado a la ecuación 2.4 y no a la ecuación 4.1 que tiene en cuenta los conjuntos de ordenaciones de las características, no este conjunto potencia.

El seguimiento de los cálculos en forma de texto tiene la forma que muestra la figura 4.7. Se puede ver que el detalle matemático es mayor, mostrando las 'instancias' que va tomando la ecuación utilizada junto con las agregaciones y el cálculo final que obtiene los valores de Shapley. Como información adicional, se muestra el nombre del conjunto de datos con el que se está trabajando.

Es interesante tener en cuenta que esta implementación puede servir para realizar una auditoría a la técnica Tree SHAP, comprobando los valores que produce con los míos.

4.2.4. Caso de uso en resistencia antibiótica

En la actualidad, la administración de antibióticos es una práctica intensiva en la atención sanitaria. Uno de los mayores esfuerzos que se están llevando a cabo es la

Definiciones formales: muestran cómo se calcula cada contribución marginal...

```
Dataset name: test

========= Calculating base value for the model...

Base value Φ_0 = [23.] and corresponds to f(S) when S is empty

Let f(x) be the conditional expectation function of the model's output

>>> Considering ordering #1 --> ['1', '2', '3']
Φ_1_1 = f(S U 1) - f(S) = 20.0 - 23.0 = -3.0
S = ['1']
Φ_2_1 = f(S U 2) - f(S) = 20.0 - 20.0 = 0.0
S = ['1', '2']
Φ_3_1 = f(S U 3) - f(S) = 20.0 - 20.0 = 0.0
S = ['1', '2', '3']

>>> Considering ordering #2 --> ['1', '3', '2']
Φ_1_2 = f(S U 1) - f(S) = 20.0 - 23.0 = -3.0
S = ['1']
Φ_3_2 = f(S U 3) - f(S) = 20.0 - 20.0 = 0.0
S = ['1', '3']
Φ_2_2 = f(S U 2) - f(S) = 20.0 - 20.0 = 0.0
S = ['1', '3', '2']
```

... Y el cálculo final de los valores de Shapley.

```
========= Summing up all feature values...

M! = 6

Φ_1 = (-3.0 + -3.0 + -7.0 + -7.0 + -3.0 + -7.0) / 6 = -5.0
Φ_2 = (0.0 + 0.0 + 4.0 + 4.0 + 0.0 + 4.0) / 6 = 2.0
Φ_3 = (0.0 + 0.0 + 0.0 + 0.0 + 0.0 + 0.0) / 6 = 0.0
```

Figura 4.7: Origen matemático de los cálculos.

definición de políticas que permitan mantener la eficacia de los antibióticos, paliando la generación de resistencias debidas a un consumo innecesario o un uso inadecuado de los mismos.

Una cuestión fundamental es el estudio de patrones y comportamiento de perfiles de pacientes en la respuesta de la resistencia a los antibióticos. Este caso de uso se centra en el análisis de una base de datos (proporcionada a través del proyecto nacional SISTUS) para tal fin.

Los datos utilizados corresponden a MIMIC-III, una base de datos *open-access* sanitaria que recopila información de más de 40.000 pacientes en Unidad de Cuidados Intensivos.

De esta base de datos, a través del proyecto SITSUS, se ha facilitado bajo supervisión especializada una cosulta y posterior filtrado de datos demográficos, muestras y estudio de susceptibilidad (resistencia o no a los antibióticos).

Los datos contienen cultivos de muestras, fundamentalmente de sangre, para analizar la presencia de bacterias y la respuesta de los antibióticos sobre las mismas. Es decir, cada fila es un cultivo. El proceso que se va a seguir para este caso de uso viene ilustrado en la figura 4.8.

En la siguiente tabla, se muestra la lista de características que componen el conjunto de datos junto con una descripción de lo que sus valores representan:

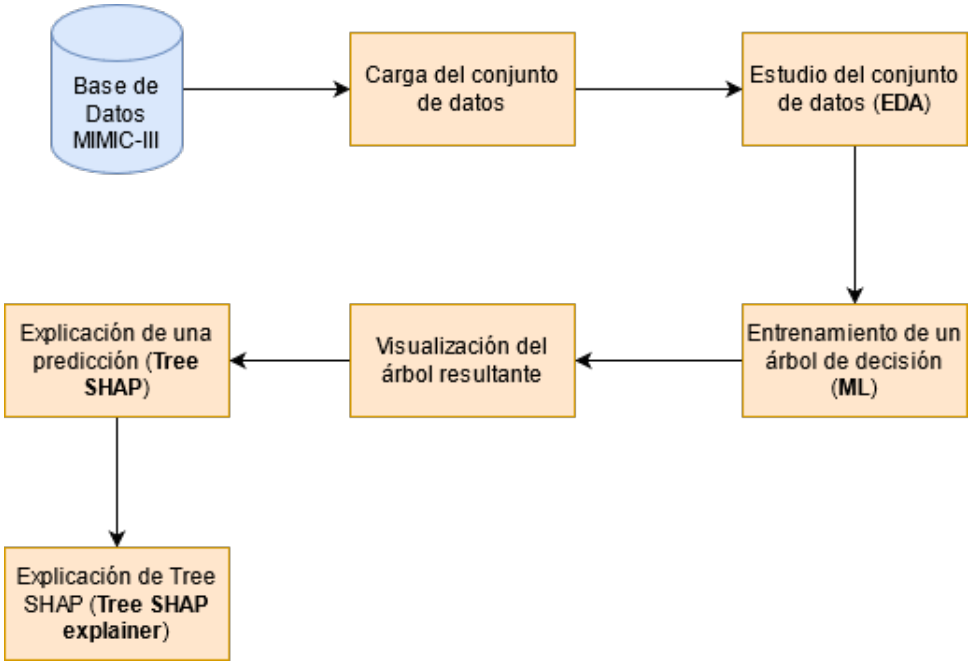

Figura 4.8: Diagrama ilustrativo de la experimentación.

Característica	**Tipo**	**Descripción**
patient_gender	M/F	Género del paciente
patient_age	Entero	Edad del paciente
exitus	Y/N	Fallecimiento del paciente
admission_type	Cadena	Servicio hospitalario por el que se realizó el ingreso
admission_location	Cadena	Lugar físico donde se realizó el ingreso
discharge_location	Cadena	Lugar donde se realizó el alta (en casa, por fallecimiento, durante servicio de rehabilitación, etc.)
culture_specimen_type_description	Cadena	Tipo de cultivo (cultivo de sangre, bilis, piel, etc.)
culture_microorganism_name	Cadena	Nombre de la bacteria
culture_susceptibility	R/S	Sensibilidad del cultivo al antibiótico
service_when_culture	Cadena	El servicio que pidió el cultivo, si es que se conoce
icu_when_culture	Cadena	Tipo de UCI cuando se hizo el cultivo
culture_mic_<=1,<=2,<=4,<=8, <=16, <=32, <=128	Y/N	Indica si el valor del Conjunto Mínimo Inhibitorio (MIC) cumple la condición

days_since_last_admission	Entero	Días que pasaron desde el ingreso anterior. Sirve para saber si el paciente tuvo un ingreso hace poco.
days_between_admission_and_first _ICU	Entero	Cuánto pasó desde que el paciente ingresó hasta que entró en UCI
days_between_last_vanc_treatment\ _and_culture	Entero	Días transcurridos desde que se proporcionó el último tratamiento de vancomicina anterior y el cultivo
duration_of_last_vancomycin\ _treatment__AA	Entero	Días transcurridos desde que se proporcionó el último tratamiento de vancomicina anterior y la realización del cultivo que se estudia ahora
number_of_last_vancomycin _treatments__AA	Entero	Número de tratamientos de vancomincina en todos los ingresos (acumulativo)
duration_in_natural_d_last _vanc_treatm_last_180d	Entero	Cuánto duraron los tratamientos de vancomincina anteriores en los últimos 6 meses anteriores al ingreso actual
n_of_last_vanc_treatments_in\ _the_last_180d	Entero	Número de tratamientos de vancomincina anteriores en los últimos 6 meses en cualquier ingreso
culture_month	Entero	Mes del año en el que el cultivo fue hecho

La variable objetivo del conjunto de datos se llama 'culture susceptibility', que es un atributo categórico que indica si el cultivo es susceptible de ser resistente (R) o sensible (S) a antibióticos. Nos encontramos, por ende, ante un problema de clasificación.

El modelo de aprendizaje que se ha escogido para la experimentación es *Decision Tree* por diferentes motivos. El primero es que este trabajo no trata sobre ML o sobre la creación de un modelo que logre una precisión espectacular sobre la clasificación, y por tanto es preferible trabajar con modelos sencillos, basados en árboles de decisión por la naturaleza de Tree SHAP. Otro motivo es que no se necesitan modelos complejos para ilustrar la técnica e intentar detectar deficiencias, ya que estas, teóricamente, estarían provocadas por los propios datos de entrada. Además, la herramienta desarrollada soporta únicamente *Decision Tree* porque la salida que produce la traza del cálculo de los valores de Shapley tiene ya un volumen considerable para un solo árbol al igual que el coste computacional de los cálculos.

Un primer análisis de los datos mostraba que existen características que no son numéricas. Esto es un problema para el entrenamiento de un modelo clasificador, así que apliqué la técnica conocida como 'One-Hot encoding' [47], que se basa en convertir cada valor categórico en una nueva columna y asignar un valor 1 o 0 (Verdadero/Falso) a la columna. La ventaja de esta técnica con respecto a otras es que no se pondera un valor

indebidamente. Si esto ocurriera, los valores numéricos pueden ser malinterpretados por el algoritmo.

Es importante mencionar que realicé otra experimentación que comentaré más adelante con este conjunto de datos, eliminando las columnas categóricas para no complicar el modelo y poder aplicar mi explicador en un tiempo razonable, puesto que considerar el factorial de las casi 90 características resultantes de aplicar 'One-Hot encoding' es inviable y su ejecución tardaría muchísimo.

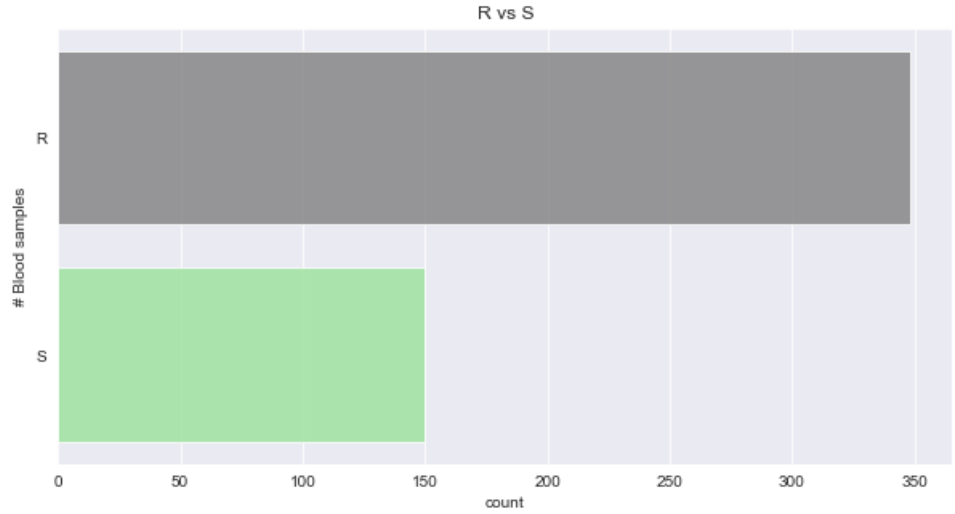

Figura 4.9: Desbalanceo del atributo clase entre los valores de Resistencia (R) y Susceptibilidad (S).

Una vez resuelto este problema, me dispuse a realizar un análisis exploratorio de los datos más a fondo, y aunque la base de datos era una vista minable, resultaba haber duplicados que eliminé. Para esta experimentación era importante incidir sobre el desequilibrio entre el número de instancias de cada clase, que como puede observarse en la figura 4.9, era muy pronunciado.

Esto es algo a tener en cuenta, pues imaginemos un modelo con dos clases con una precisión del 90 % para el que el 90 % de las instancias son de una de esas dos. Estaríamos ante un modelo poco fiable y no solamente eso, sino que además, las características que un explicador determinaría como más o menos importantes podrían llevar a equivocación y provocar una toma de decisiones errónea en el mundo real. Precisamente se quiere estudiar este impacto por lo que viene bien este desequilibro.

Otro problema que pueden presentar los datos, es la correlación entre las distintas variables, que puede verse en la figura 4.10.

Los tonos de verde representan una correlación positiva, mientras que los tonos marrones representan una correlación negativa. Como se puede observar, existe una fuerte correlación positiva entre las características que indican cuánto duraron los tratamientos de vancomincina anteriores a cualquier franja de tiempo, y el número de tratamientos de vancomincina anteriores a cualquier franja de tiempo también. Cuantos más tratamientos se hayan tenido, más parecen durar.

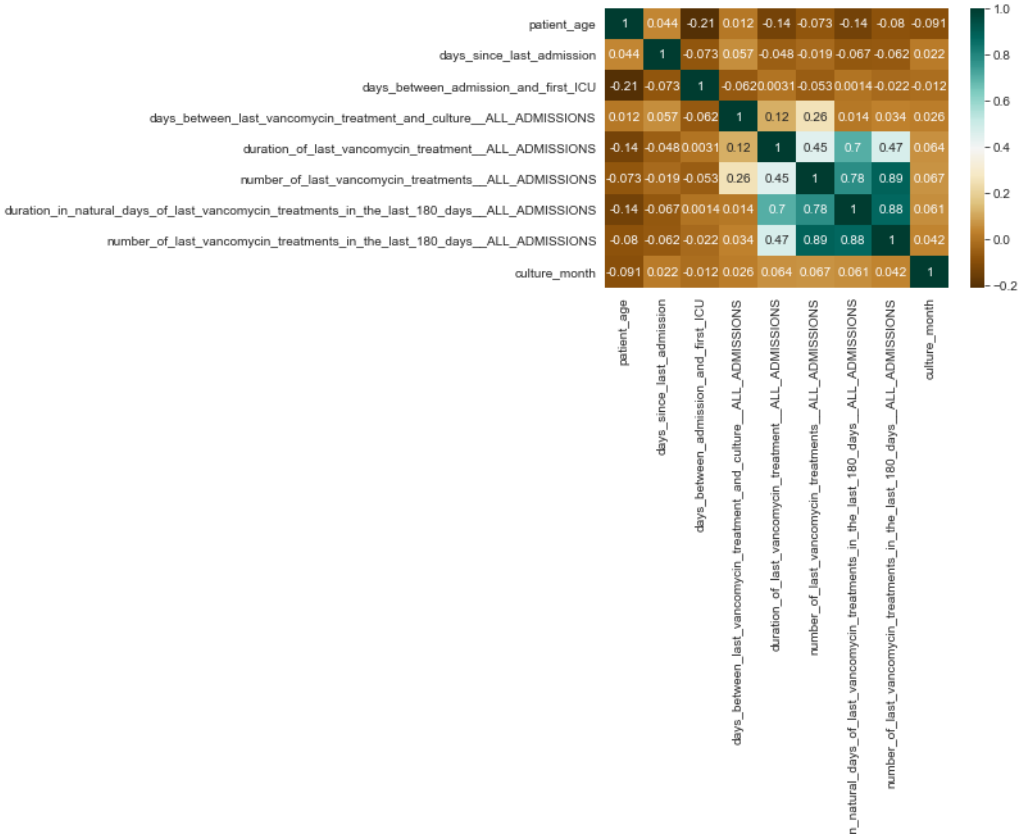

Figura 4.10: Correlación entre las características de MIMIC-III.

El resto de características están poco o nada correlacionadas, excepto la que indica cuánto pasó desde que el paciente ingresó hasta que entró en UCI ('days_between_admission_and_first_ICU') y la edad del paciente, con una ligera correlación negativa, indicando que cuanta más edad tenía el paciente, antes ingresaba en UCI.

En principio, para lo que se centra este trabajo, no hay constancia de que la correlación de variables afecte de manera alguna al cálculo de valores de Shapley y su fiabilidad. Es cierto que si se estuviera tratando de resolver un problema con un conjunto de datos complejo, lo conveniente sería aplicar una reducción de dimensionalidad de características que afecten poco a la variabilidad de los datos, como RFE o PCA. Como no es el caso, se tuvo en cuenta esta condición para el análisis, pero no se aplicó ninguno de estos métodos.

También comprobé si el conjunto tenía datos atípicos, estudiando el recuento, la media, la desviación estándar, los valores mínimos y máximos y los cuantiles de los datos. Las observaciones de este análisis observaciones sugerían que había valores extremos en ciertos casos, como en los días entre la admisión y el ingreso en UCI, por lo que era probable que la existencia de estos valores atípicos provocaran algunas explicaciones erróneas o con poco sentido, así que se tuvo en cuenta de cara a comprobar el funcionamiento de Tree SHAP (aunque no para el entrenamiento del modelo, ya que el trabajo no va sobre ML). Estos valores se pueden ver reflejados en la figura 4.11.

Con todos estos datos en mente, el siguiente paso es realizar el entrenamiento del modelo, con los hiperparámetros por defecto que se determinen. Insisto en que no es el

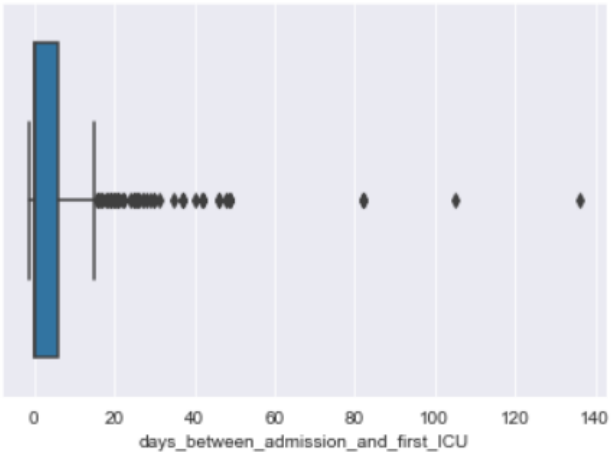

Figura 4.11: Diagrama de cajas para la característica que indica el número de días entre la admisión y el ingreso en UCI.

objetivo de la experimentación el ajuste de hiperparámetros. La salida que produce Tree SHAP es independiente de lo bueno que sea el modelo, en el sentido de que no requiere que clasifique perfectamente para establecer valores de importancia a las características. El modelo resultante puede verse en la figura 4.12.

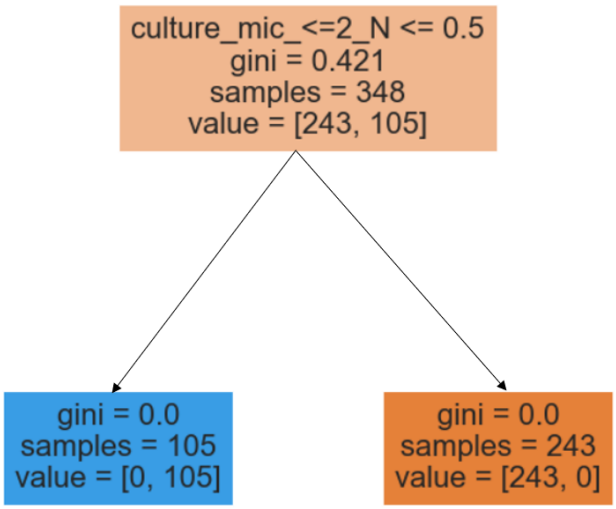

Figura 4.12: Árbol de decisión resultante del entrenamiento de MIMIC-III.

Una vez entrenado el modelo, he aplicado el explicador Tree SHAP sobre una instancia cualquiera del conjunto de 'test', además de mi implementación propia de Tree SHAP junto con la aplicación del explicador de este explicador, para generar la traza de los cálculos de valores SHAP, y se obtuvieron los resultados de la figura 4.13 y 4.14.

La figura 4.13, denominada diagrama de fuerzas, muestra una explicación por cada instancia del conjunto de 'test'. A pesar de que Tree SHAP se centra en explicaciones locales, ofrece una manera novedosa que explicar conjuntos completos rotando los diagramas producido por una sola explicación 90 grados y poniéndolos todos en fila.

Como era de esperar por la forma del árbol resultante, la única característica que influye en la clasificación es 'culture_mic_<= 2'. Esta característica indica el valor 'mic'

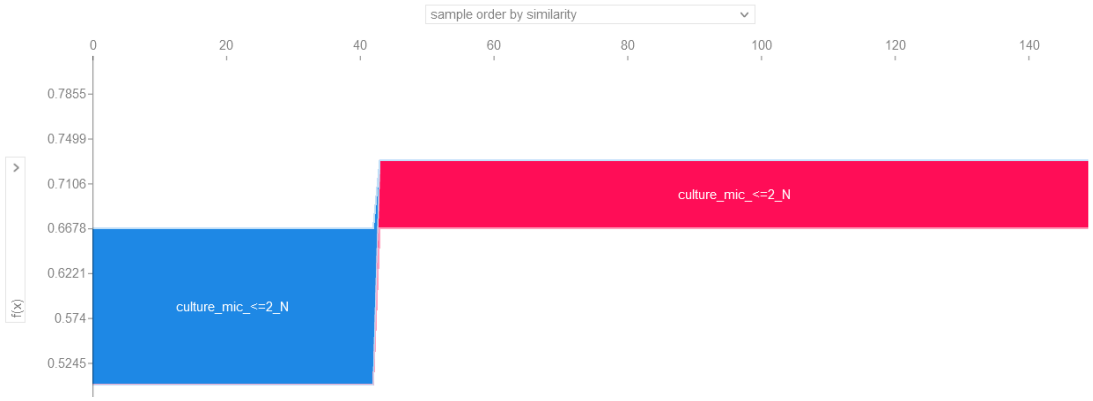

Figura 4.13: Diagrama de fuerzas para múltiples instancias de 'test' del conjunto MIMIC-III.

del cultivo, que representa la concentración más baja de un fármaco para impedir el crecimiento de una bacteria o un bacilo. Es por ello que este valor es el que tiene más impacto para clasificar y el explicador Tree SHAP muestra mediante sus valores de SHAP este hecho. La forma de interpretar estos diagramas se comenta de forma más detallada en el apartado de Anexos.

En la figura 4.14 puede observarse el mismo diagrama para una sola instancia, indicando que la única característica que tiene importancia para clasificar es 'culture_mic_<= 2'.

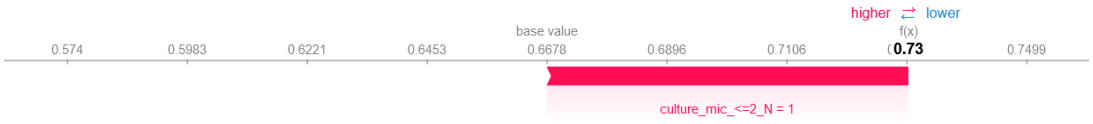

Figura 4.14: Diagrama de fuerzas para una instancia de 'test' del conjunto MIMIC-III.

Para este conjunto de datos, el desequilibro entre clases objetivo, la correlación entre ciertas características o la inclusión de valores atípicos no ha afectado a las explicaciones. Esto es debido, en parte, a que este conjunto de datos no está preparado realmente para ser usado en aprendizaje automático y la sencillez del modelo que se puede construir con el mismo facilita el trabajo al explicador Tree SHAP.

En la experimentación con el conjunto MIMIC-III simplificado, eliminando las características no numéricas, se aplicó mi explicador de Tree SHAP. Al considerarse 9 características tras la modificación del conjunto, se consideraban $9! = 362880$ ordenaciones, y dentro de cada una se realizan 9 cálculos de contribuciones marginales. En total, $9!\ x\ 9 = 3265920$ cálculos. No solamente pueden consultarse todos los pasos de la ecuación 4.1 para validar los valores SHAP, sino que tras tal cantidad de cálculos, los valores SHAP coinciden con los obtenidos con el paquete *shap* como muestra la figura 4.15.

Es por esto que aparte de mostrar la traza matemática de los cálculos, se le está dando fiabilidad a la técnica en este experimento, ya que esta prueba actúa como auditor al método Tree SHAP. La visualización por grafo de la representación de las coaliciones no

```
============================== SHAP values from explainer ==============================
[[0.21304, 0.12107, -0.01001, 0.01445, 0.00492, 3e-05, -0.00993, 0.00399, -0.03298],
 [-0.21304, -0.12107, 0.01001, -0.01445, -0.00492, -3e-05, 0.00993, -0.00399, 0.03298]]

============================== SHAP values from shap module ============================
[array([[ 2.13040050e-01,  1.21067602e-01, -1.00063957e-02,
          1.44525721e-02,  4.92268715e-03,  2.97793032e-05,
         -9.93034474e-03,  3.99098503e-03, -3.29847245e-02]]),
 array([[-2.13040050e-01, -1.21067602e-01,  1.00063957e-02,
         -1.44525721e-02, -4.92268715e-03, -2.97793032e-05,
          9.93034474e-03, -3.99098503e-03,  3.29847245e-02]])]
```

Figura 4.15: Salida del explicador de Tree SHAP sobre MIMIC-III simplificado.

tiene sentido en este caso, pues habría un número de nodos demasiado grande para ser legible (pensemos que hay $9! = 362880$ nodos hoja sin contar nodos internos).

Cabe decir que esta aplicación se ha realizado sobre otro conjunto de datos a parte del artificial y el médico. Estas experimentaciones están disponibles para su consulta en el apartado de Anexos de forma separada ya que no aportan nueva información sobre problemas que pueda presentar la técnica, más allá de su aplicación y validación en diferentes situaciones.

4.2.5. Conclusiones de la experimentación realizada

A la luz de los resultados obtenidos en la experimentación anterior y de las incluidas en Anexos, se puede concluir que:

- Tree SHAP está basado en los fundamentos matemáticos que clama tener y esta herramienta puede utilizarse como auditoría a la técnica para validar sus resultados.
- Con un modelo relativamente complejo, está bien disponer de la herramienta desarrollada para transmitir fiabilidad y confianza a la hora de tomar decisiones, pero el número de contribuciones marginales es excesivo y la consulta del cálculo de todas ellas es inviable.
- Con modelos más sencillos, es posible entender cómo funciona Tree SHAP internamente de forma sencilla, evitando al máximo los formalismos e ilustrando las matemáticas subyacentes a través de ejemplos en forma de grafo, dotando a usuarios no expertos una visión intuitiva sobre el origen de los valores SHAP. También se facilita el entendimiento para usuarios expertos que quieran adentrarse en el campo de XAI y Teoría de Juegos de manera más ágil.
- Aunque los datos presenten deficiencias que puedan afectar al entrenamiento de los modelos, Tree SHAP parece funcionar independientemente de estos datos. El problema se encontraría en el lado del entrenamiento de modelos, que es un problema que nada tiene que ver con el explicador.

CAPÍTULO 5

Conclusiones y vías futuras

En este trabajo se ha desarrollado una herramienta cuyo objetivo era facilitar la interpretación de explicaciones sobre predicciones realizadas por modelos de árboles de decisión obtenidos mediante ML, por parte de usuarios no expertos en técnicas matemáticas formales.

Creemos que se han alcanzado los objetivos planteados, ya que la herramienta desarrollada tiene utilidad para los usuarios objetivo para los que estaba orientada:

- En el caso de los científicos de datos, puede utilizarse para facilitar la comprobación de si el método de explicabilidad utilizado (en este caso Tree SHAP) es la más adecuada para el problema que se quiera resolver.
- En caso de los usuarios finales, como médicos, les sirve como apoyo a la toma de decisiones en el ámbito clínico puesto que logra explicar cómo funciona el explicador, favoreciendo la transparencia del cálculo de los valores de importancia de las características, consiguiendo validar y dar fiabilidad a descubrimientos que se produzcan.

En cuanto a dificultades encontradas, he de decir que el propio tema que trata este trabajo me resultó difícil de abordar al empezar a documentarme, y la poca información que existe alrededor es un factor limitante que hizo que la curva de aprendizaje fuera muy pronunciada. El hecho de pensar una forma de lograr explicar una explicación producida por el método Tree SHAP posee esa componente creativa que me llevó tiempo tomar por la naturaleza matemática del algoritmo. Hablo, por tanto, de dificultades que tienen que ver más con la comprensión y el conocimiento, que con problemas estructurales y organizativos de la implementación llevada a cabo.

A pesar de la utilidad de la herramienta desarrollada, se sigue tratando de una 'caja negra' para usuarios finales en el sentido de que se requieren conocimientos sólidos propios de ciencia de datos y aprendizaje automático, e incluso mátemáticos, para que el origen de los valores de importancia producidos a partir de una predicción lleguen a ser realmente fiables y asistan en las decisiones médicas que se toman cada día.

El problema reside en el hecho de que la ingeniosa técnica Tree SHAP está basada en fundamentos fuertemente formales, y este tipo de modelos no son fácilmente trasladables al ser humano de manera universal, por lo que mientras no entre en escena en este paradigma un nuevo modelo, igual de potente y eficaz, cuyas bases no se cimenten sobre estructuras matemáticas excesivamente complejas, es inevitable utilizar estas técnicas de explicabilidad como 'cajas negras', con las que se 'debe creer' que el valor de importancia asignado a cada característica dentro de un conjunto de datos es fiable, a pesar de conocer el origen de su cálculo o disponer del mismo.

En este sentido, existe un camino por recorrer todavía en el campo de la explicabilidad, puesto que se tienen métodos rápidos y aparentemente fiables, pero siguen sin ser del todo intuitivos si son muy complejos, y pueden provocar rechazo y desconfianza si no lo son. Es un equilibrio difícil de alcanzar y que, de seguro, será complicado conseguir en un futuro cercano y dará que hablar en los años venideros.

Bibliografía

[1] "Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts", 2021. URL: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=COM:2021:206:FIN.

[2] U Rajendra Acharya, Oliver Faust, Vinitha Sree Subbhuraam, Dhanjoo Ghista, Sumeet Dua, Paul Joseph, Ahamed Thajudin, Nittiagandhi Janarthanan, and Toshiyo Tamura. An integrated diabetic index using heart rate variability signal features for diagnosis of diabetes. *Computer methods in biomechanics and biomedical engineering*, 16, 10 2011. doi:10.1080/10255842.2011.616945.

[3] E. Ray Bareiss, Bruce W. Porter, and Craig C. Wier. Protos: An exemplar-based learning apprentice. In Pat Langley, editor, *Proceedings of the Fourth International Workshop on MACHINE LEARNING*, pages 12–23. Morgan Kaufmann, 1987. URL: https://www.sciencedirect.com/science/article/pii/B9780934613415500052, doi:https://doi.org/10.1016/B978-0-934613-41-5.50005-2.

[4] R. Bareiss and B. Chandrasekaran. *Exemplar-based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning*. Personality and Psychopathology. Academic Press, 1989. URL: https://books.google.es/books?id=8p9QAAAAMAAJ.

[5] Adam Bloniarz, Ameet Talwalkar, Bin Yu, and Christopher Wu. Supervised neighborhoods for distributed nonparametric regression. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1450–1459, Cadiz, Spain, 09–11 May 2016. PMLR. URL: http://proceedings.mlr.press/v51/bloniarz16.html.

[6] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984. URL: https://books.google.es/books?id=JwQx-WOmSyQC.

[7] J. Brown. Pedagogical, natural language, and knowledge engineering techniques in sophie-i, ii and iii. 1982.

[8] Jason Brownlee. "A gentle introduction to the gradient boosting algorithm for machine learning", 2016. URL: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

[9] Brandon Carter, Jonas Mueller, Siddhartha Jain, and David K. Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. *CoRR*, abs/1810.03805, 2018. URL: http://arxiv.org/abs/1810.03805, arXiv:1810.03805.

[10] G. Chalkiadakis, E. Elkind, and M.J. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, 2011. URL: https://books.google.es/books?id=bN9aC0uabBAC.

[11] Nagesh Singh Chauhan. "Decision tree algorithm, explained", 2020. URL: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL: http://arxiv.org/abs/1603.02754, arXiv:1603.02754.

[13] Adrian Colyer. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead", 2019. URL: https://www.kdnuggets.com/2019/11/stop-explaining-black-box-models.html.

[14] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. pages 598–617, 05 2016. doi:10.1109/SP.2016.42.

[15] Niklas Donges. "A complete guide to the random forest algorithm", 2020. URL: https://builtin.com/data-science/random-forest-algorithm.

[16] IBM Cloud Education. "Random forest", 2020. URL: https://www.ibm.com/cloud/learn/random-forest.

[17] Lilian Edwards and Michael Veale. Slave to the algorithm? why a right to explanationn is probably not the remedy you are looking for. *SSRN Electronic Journal*, 16, 12 2017. doi:10.2139/ssrn.2972855.

[18] R. Edwards. Knowledge based tutoring (author: William j. clancy). *McGill Journal of Education / Revue des sciences de léducation de McGill*, 23(002), Apr. 1988. URL: https://mje.mcgill.ca/article/view/7817.

[19] Weng S. F., Reps J., Kai J., Garibaldi J. M., and Qureshi N. "Can machine-learning improve cardiovascular risk prediction using routine clinical data?". *PLoS One*, 2017. URL: https://doi.org/10.1371/journal.pone.0174944.

[20] Lawrence Fagan, Edward Shortliffe, and Bruce Buchanan. Computer-based medical decision making: from mycin to vm. 3, 01 1984.

[21] Oliver Faust and Eddie Ng. Computer aided diagnosis for cardiovascular diseases based on ecg signals: A survey. *Journal of Mechanics in Medicine and Biology*, 16:1640001, 02 2016. doi:10.1142/S0219519416400017.

[22] K. Frankish and W.M. Ramsey. *The Cambridge Handbook of Artificial Intelligence*. Cambridge University Press, 2014. URL: https://books.google.es/books?id=ECCmAwAAQBAJ.

[23] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 10 2001. doi:10.2307/2699986.

[24] Jerome Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 02 2002. doi:10.1016/S0167-9473(01)00065-2.

[25] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55:72–99, 02 2006. doi:10.1016/j.geb.2005.03.002.

[26] Funda Günes. "4 model-agnostic interpretability techniques for complex models", 2020. URL: https://blogs.sas.com/content/subconsciousmusings/2020/05/07/model-agnostic-interpretability/.

[27] Sergiu Hart. *Shapley Value*, pages 210–216. Palgrave Macmillan UK, London, 1989. doi:10.1007/978-1-349-20181-5_25.

[28] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. *CoRR*, abs/1603.08507, 2016. URL: http://arxiv.org/abs/1603.08507, arXiv:1603.08507.

[29] Jose Hernandez-Orallo. Introducción a la minería de datos. 01 2004.

[30] Andreas Holzinger, Chris Biemann, Constantinos S. Pattichis, and Douglas B. Kell. "What do we need to build explainable AI systems for the medical domain?". *CoRR*, abs/1712.09923, 2017. URL: http://arxiv.org/abs/1712.09923, arXiv:1712.09923.

[31] Jeremy Kahn. "Artificial intelligence has some explaining to do", 2018. URL: https://www.bloomberg.com/news/articles/2018-12-12/artificial-intelligence-has-some-explaining-to-do.

[32] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10:e0130140, 07 2015. doi:10.1371/journal.pone.0130140.

[33] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17:319 – 330, 10 2001. doi:10.1002/asmb.446.

[34] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017. URL: http://arxiv.org/abs/1705.07874, arXiv:1705.07874.

[35] Scott M. Lundberg. "Shap: A game theoretic approach to explain the output of any machine learning model.", 2018. URL: https://github.com/slundberg/shap.

[36] Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. "Explainable AI for trees: From local explanations to global understanding". *CoRR*, abs/1905.04610, 2019. URL: http://arxiv.org/abs/1905.04610, arXiv:1905.04610.

[37] Scott M. Lundberg, Bala Nair, Monica S. Vavilala, Mayumi Horibe, Michael J. Eisses, Trevor Adams, David E. Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, and Su-In Lee. "Explainable machine learning predictions to help anesthesiologists prevent hypoxemia during surgery". *bioRxiv*, 2017. URL: https://www.biorxiv.org/content/early/2017/10/21/206540, arXiv:https://www.biorxiv.org/content/early/2017/10/21/206540.full.pdf, doi:10.1101/206540.

[38] David Martens and Foster Provost. Explaining data-driven document classifications. *MIS Quarterly*, 38:73–100, 03 2014. doi:10.25300/MISQ/2014/38.1.04.

[39] Roshan Martis, Chandan Chakraborty, and Ajoy Ray. *Wavelet-based Machine Learning Techniques for ECG Signal Analysis*, volume 56, pages 25–45. 01 2014. doi:10.1007/978-3-642-40017-9_2.

[40] Samuele Mazzanti. "SHAP Values Explained Exactly How You Wished Someone Explained to You", 2020. URL: https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30.

[41] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[42] Christoph Molnar. "Kernel SHAP", 2019. URL: https://christophm.github.io/interpretable-ml-book/shap.html#kernelshap.

[43] Christoph Molnar. "SHAP disadvantages.", 2019. URL: https://christophm.github.io/interpretable-ml-book/shap.html#disadvantages-14.

[44] Siddhartha Mukherjee. "A.I. Versus M.D", 2017. URL: https://www.newyorker.com/magazine/2017/04/03/ai-versus-md.

[45] Avinash Navlani. "Decision Tree Classification in Python", 2018. URL: https://www.datacamp.com/community/tutorials/decision-tree-classification-python.

[46] James Owers. "Passing categorical data to sklearn decision tree", 2016. URL: https://stackoverflow.com/questions/38108832/passing-categorical-data-to-sklearn-decision-tree.

[47] Manish Pathak. "Handling Categorical Data in Python", 2020. URL: https://www.datacamp.com/community/tutorials/categorical-data.

[48] Gregory Plumb. "MAPLE: Towards interpretable tree ensembles", 2019. URL: https://blog.ml.cmu.edu/2019/07/13/towards-interpretable-tree-ensembles/.

[49] Gregory Plumb, Denali Molitor, and Ameet Talwalkar. "Supervised local modeling for interpretability". *CoRR*, abs/1807.02910, 2018. URL: http://arxiv.org/abs/1807.02910, arXiv:1807.02910.

[50] Joshua Poduska. "SHAP and LIME python libraries: Part 1 great explainers, with pros and cons to both", 2018. URL: https://blog.dominodatalab.com/shap-lime-python-libraries-part-1-great-explainers-pros-cons/.

[51] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Local Interpretable Model-Agnostic Explanations (LIME): An Introduction", 2016. URL: https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/.

[52] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should I trust you?": Explaining the predictions of any classifier". *CoRR*, abs/1602.04938, 2016. URL: http://arxiv.org/abs/1602.04938, arXiv:1602.04938.

[53] Marco Tulio Correia Ribeiro. "Lime: Explaining the predictions of any machine learning classifier.', 2016. URL: https://github.com/marcotcr/lime.

[54] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002. URL: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0137903952.

[55] Ian Sample. "Computer says no: why making AIs fair, accountable and transparent is crucial", 2017. URL: https://www.theguardian.com/science/2017/nov/05/computer-says-no-why-making-ais-fair-accountable-and-transparent-is-crucial.

[56] L.S. Shapley. *A Value for N-person Games*. Defense Technical Information Center. URL: https://books.google.es/books?id=QzW6tgAACAAJ.

[57] L.S. Shapley, A.E. Roth, and Cambridge University Press. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988. URL: https://books.google.es/books?id=JK7MKu2A9cIC.

[58] Edward H. Shortliffe and Martin J. Sepúlveda. "Clinical decision support in the era of artificial intelligence". *Jama 320.21*, pages 2199–2200, 2018. URL: https://www.deepdyve.com/lp/american-medical-association/clinical-decision-support-in-the-era-of-artificial-intelligence-xfPeJrApKo.

[59] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017. URL: http://arxiv.org/abs/1704.02685, arXiv:1704.02685.

[60] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016. URL: http://arxiv.org/abs/1605.01713, arXiv:1605.01713.

[61] Erik Strumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, 12 2013. doi:10.1007/s10115-013-0679-x.

[62] Erik Strumbelj, Igor Kononenko, and Marko Robnik-Sikonja. Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, 68:886–904, 10 2009. doi:10.1016/j.datak.2009.01.004.

[63] Rakesh Sukumar. "SHAP Part 3: Tree SHAP", 2020. URL: https://medium.com/analytics-vidhya/shap-part-3-tree-shap-3af9bcd7cd9b.

[64] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365, 2017. URL: http://arxiv.org/abs/1703.01365, arXiv:1703.01365.

[65] Alan D. Taylor and William S. Zwicker. *Simple Games: Desirability Relations, Trading, Pseudoweightings*. Princeton University Press, 1999. URL: http://www.jstor.org/stable/j.ctv18zhdsm.

[66] Python Team. "The Python Standard Library", 2001. URL: https://docs.python.org/3/library/.

[67] R Team. "Shapley values R implementation", 2020. URL: https://rdrr.io/cran/iml/src/R/Shapley.R.

[68] Thibaut Vidal, Toni Pacheco, and Maximilian Schiffer. Born-again tree ensembles. *CoRR*, abs/2003.11132, 2020. URL: https://arxiv.org/abs/2003.11132, arXiv:2003.11132.

[69] Peng Xia, J. Hu, and Y. Peng. Emgbased estimation of limb movement using deep learning with recurrent convolutional neural networks. *Artificial Organs*, 42:E67E77, 2018.

APÉNDICE A

Experimentación con un conjunto de datos artificial

En este anexo presento la aplicación de la herramienta desarrollada, 'Tree SHAP Explainer', sobre un conjunto de datos artificial [63], que utilicé para ir comprobando mis resultados durante el desarrollo de la utilidad.

Nos encontramos ante un problema de regresión, para el que se entrena un árbol de decisión regresor con los hiperparámetros sugeridos por el creador del conjunto de datos, para poder obtener así los mismos valores y comprobar que el cálculo es correcto.

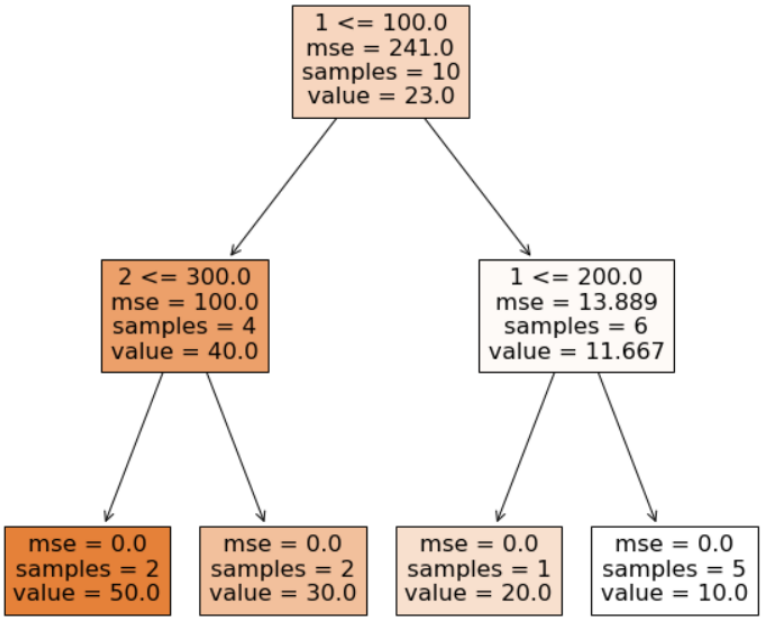

Figura A.1: Árbol de decisión resultante del entrenamiento con el conjunto de datos artificial.

Al tratarse de un conjunto de datos pequeño con pocas características, no tiene sentido realizar un análisis de los datos previo. Así que el primer paso es entrenar un árbol de decisión con el conjunto de entrada para poder ejecutar el explicador Tree SHAP. El modelo generado se puede ver en la figura A.1.

A continuación, aplico mi explicador de Tree SHAP al conjunto y al modelo. De esta aplicación se construye un fichero de texto con la traza del cálculo realizado por Tree SHAP, que puede ser consultado para esclarecer de dónde provienen los valores SHAP calculados, observando todas las contribuciones marginales de cada característica a la predicción. Además, el grafo resultante tras aplicar mi herramienta es el de la figura A.2.

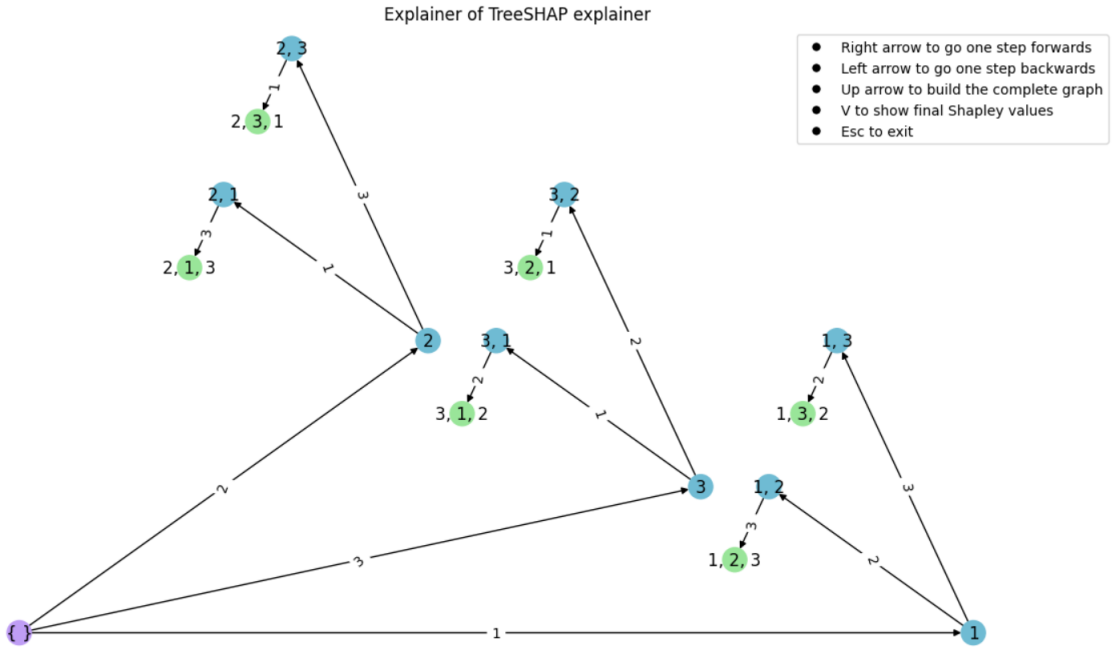

Figura A.2: Grafo de coaliciones para el conjunto de datos artificial.

Los valores calculados por Tree SHAP para la instancia de 'test' se pueden visualizar mediante los *plots* que ofrece la librería *shap*, como el 'force_plot'. La forma de interpretarlo es entender que las características azules en la derecha hacen que la predicción final disminuya su valor, y las de la izquierda lo aumentan. Cuanto más grande sea su contribución marginal, más grande es su barra y más impacto tienen en el cálculo, siendo por tanto más 'importantes' o determinantes en el valor último de la predicción.

Puede observarse además, que los valores de Shapley obtenidos, $[-5, 2, 0]$, es lo que se suma o se resta al 'valor base' comentado en el trabajo, ya que estos valores representan el impacto de la inclusión de las características de una instancia sobre el valor de la predicción media.

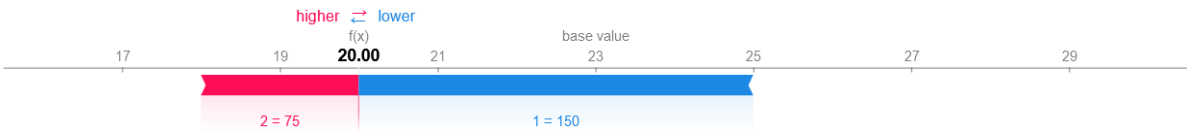

Figura A.3: Diagrama de fuerzas sobre una instancia de 'test' del conjunto artificial.

No es casualidad que estos valores que empujan el valor base acaben dando como resultado el valor predecido por el modelo (20), puesto que este comportamiento es una característica de SHAP, que viene de los fundamentos en los que se basa.

En este caso, se puede extraer del diagrama creado que la característica 1 tiene un mayor impacto que la 2 (la característica 1 reduce el valor de la predicción media en 5 y la característica 2 lo suma en 2). La característica 3 no aporta nada a la predicción y por eso no se muestra en el diagrama.

APÉNDICE B

Experimentación con el conjunto de datos Iris

Para esta experimentación, voy a utilizar el archiconocido conjunto de datos Iris. Este conjunto está preparado para resolver un problema de clasificación de flores en función de 4 características que tienen que ver con la longitud y anchura de sus pétalos y sépalos. De esta manera se muestra el uso de Tree SHAP con un modelo de clasificación, a diferencia del conjunto de datos artificial del apartado anterior.

Nuevamente, como este conjunto está preparado para hacer pruebas, no es necesario un análisis exploratorio de los datos. Así que el primer paso es entrenar un árbol de decisión con el conjunto de entrada para poder ejecutar el explicador Tree SHAP. Los hiperparámetros son los que se establezcan por defecto. Recordemos que este trabajo no va sobre ML ni sobre ajuste de hiperparámetros. El modelo generado para el cálculo de los valores SHAP tiene la forma de la figura B.1.

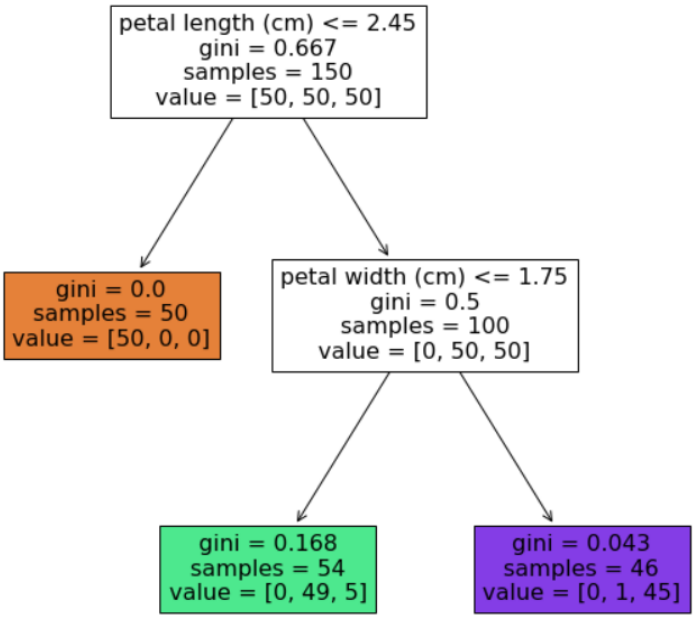

Figura B.1: Árbol de decisión resultante del entrenamiento con el conjunto de datos Iris.

Con el árbol puede observarse rápidamente que la instancia de 'test' se clasificaría como una flor 'setosa'. Una vez entrenado el modelo, el siguiente paso es calcular los valores SHAP de la instancia de 'test' con el explicador Tree SHAP y el modelo entrenado.

A diferencia de con un problema de regresión, a pesar de tratarse de una única instancia, se obtienen 3 listas de valores SHAP, una por cada clase, indicando para cada de ellas la importancia de cada característica para determinar la pertenencia o la no pertenencia a una clase. Estos valores son $[0, 0, 0.667, 0]$, $[0, 0, -0.469, 0.136]$ y $[0, 0, -0.198, -0.136]$.

A continuación, aplico mi explicador de Tree SHAP al conjunto y al modelo. De esta aplicación se construye un fichero de texto con la traza del cálculo realizado por Tree SHAP, disponible para su consulta en caso de desconfianza con los valores generados. Además, el grafo de coaliciones y la consulta de una de las contribuciones marginales se muestra en la figura B.2.

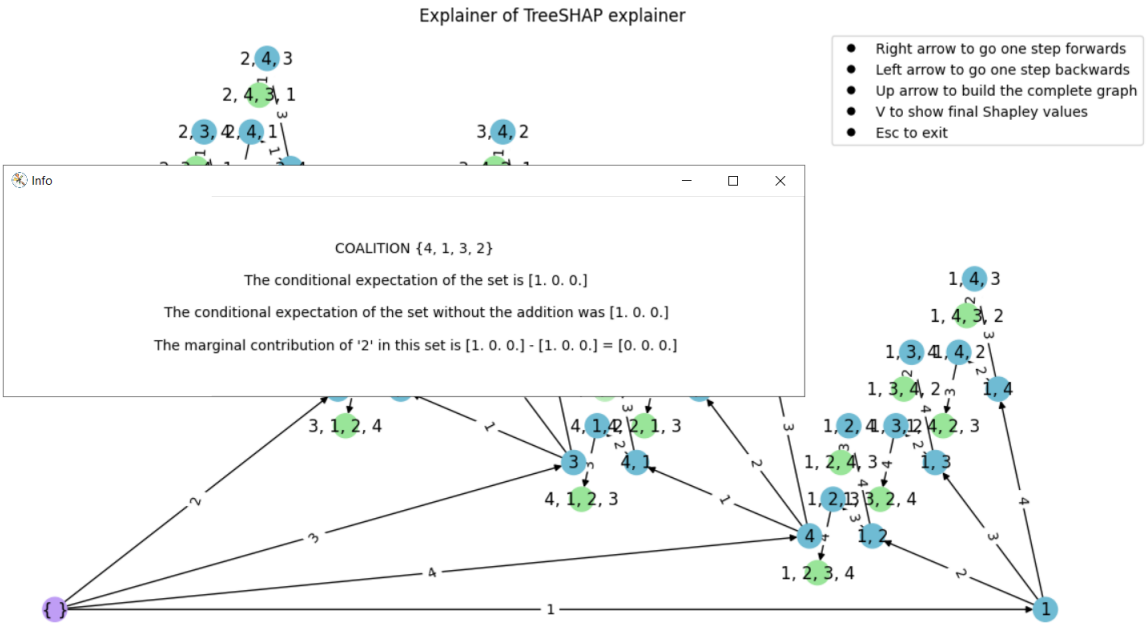

Figura B.2: Grafo de coaliciones para el conjunto de datos Iris.

Los valores calculados por Tree SHAP para la instancia de 'test' se pueden visualizar mediante los *plots* que ofrece la librería *shap*, como el 'force_plot'.

Puede observarse que los valores SHAP obtenidos para la especie 'versicolor', por ejemplo, son $[0, 0, -0.469, 0.136]$, que es lo que se suma o se resta al 'valor base' comentado en el trabajo, ya que estos valores representan el impacto de la inclusión de las características de una instancia sobre el valor de la predicción media.

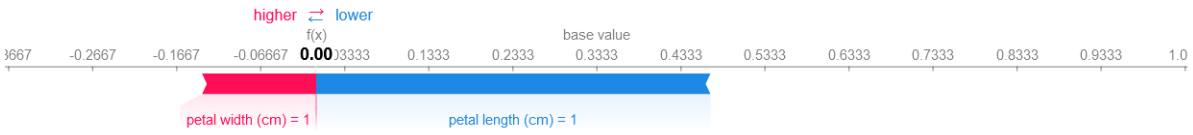

Figura B.3: Diagrama de fuerzas sobre una instancia de 'test' del conjunto Iris.

En el diagrama de la figura B.3 puede verse que, para la instancia de 'test' definida, con todas las características con valor de 1, a la hora de clasificar la flor como 'versicolor', tienen impacto dos características, que contribuyen a empujar la salida del modelo desde el valor base (la salida media del modelo sobre el conjunto de datos de entrenamiento que pasamos) hacia 0; la anchura y la longitud del pétalo.

Los valores SHAP de estas características se contrarrestan de tal manera que coinciden con la salida del modelo para esta clasificación, que es de 0 (es decir, que la flor no es 'versicolor'). La interpretación de estos datos es que "para determinar que la flor no se puede clasificar como 'versicolor', tienen importancia la anchura y longitud de sus pétalos, siendo más determinante en la decisión la longitud que la anchura".

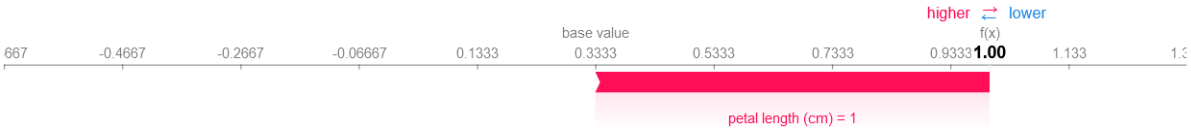

Figura B.4: Diagrama de fuerzas sobre una instancia de 'test' del conjunto Iris.

Si se toma la salida de valores SHAP de 'setosa' generados por la instancia de 'test', se obtiene el siguiente diagrama de la figura B.4, que indica que para determinar que la flor de la instancia de 'test' es de clase 'setosa', tiene impacto únicamente la longitud del pétalo.