Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área: 5

Análisis de modelos de Deep Learning mediante SHAP values.

Autor: Germán Yepes Calero

Tutor: Roger Pros Rius

Profesor: Albert Solé Ribalta

Barcelona, 20 de diciembre de 2020

Créditos/Copyright

Una página con la especificación de créditos/copyright para el proyecto (ya sea aplicación por un lado y documentación por el otro, o unificadamente), así como la del uso de marcas, productos o servicios de terceros (incluidos códigos fuente). Si una persona diferente al autor colaboró en el proyecto, tiene que quedar explicitada su identidad y qué hizo.

A continuación se ejemplifica el caso más habitual, aunque se puede modificar por cualquier otra alternativa:

FICHA DEL TRABAJO FINAL

Título del trabajo:	Análisis de modelos de Deep Learning mediante SHAP values.
Nombre del autor:	Germán Yepes Calero
Nombre del colaborador/a docente:	Roger Pros Rius
Nombre del PRA:	Albert Solé Ribalta
Fecha de entrega (mm/aaaa):	MM/AAAA
Titulación o programa:	Máster en Ciencia de Datos
Área del Trabajo Final:	AI Model selection and explainability
Idioma del trabajo:	Español
Palabras clave	Deep Learning, Explainability, SHAP values

Dedicatoria/Cita

Breves palabras de dedicatoria y/o una cita.

Agradecimientos

Si se considera oportuno, mencionar a las personas, empresas o instituciones que hayan contribuido en la realización de este proyecto.

Abstract

Deep Learning is increasingly applied in decision making and analytical problems. As a consequence, there is a growing demand for the ability to explain model prediction. One explanation technique is through the use of Shapley values (SHAP: SHapley Additive exPlanations) as a measure of aggregation to determine the influence of individual characteristics on the prediction of the collective model. SHAP values have very advantageous properties in the analysis of this type of model, since they use notions of game theory to measure the influence of the different features on the final result of the model. Throughout this work, we will develop and program a tool that allows us to interpret Deep Learning models using SHAP values, giving us a better interpretation of the factors on which the model is based when making one decision or another. All the design and implementation of the work will be carried out in a generalized and parameterized way, so that the tool can be used in multiple Deep Learning problems.

Keywords: Machine Learning, Deep Learning, Explainability, Artificial Intelligence, Data Science.

Resumen

El Deep Learning se aplica cada vez más en la toma de decisiones y problemas analíticos. Como consecuencia, existe una demanda creciente por la capacidad de explicar la predicción de modelos. Una técnica de explicación es mediante la utilización de los valores de Shapley (SHAP: SHapley Additive exPlanations) como medida de agregación para determinar la influencia de las características individuales en la predicción del modelo colectivo. Los SHAP values presentan propiedades muy ventajosas en el análisis de este tipo de modelos, pues usan nociones de teoría de juegos para medir la influencia de las diferentes características en el resultado final del modelo. A lo largo de este trabajo, desarrollaremos y programaremos una herramienta que nos permita interpretar modelos de Deep Learning usando las SHAP values, dándonos una mejor interpretación de los factores en los que se basa el modelo a la hora de tomar una decisión u otra. Todo el diseño e implementación del trabajo se realizará de manera generalista y parametrizada, de modo que la herramienta pueda ser en empleada en múltiples problemas de Deep Learning.

Palabras clave: Machine Learning, Deep Learning, Explainability, Artificial Intelligence, Data Science.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Descripción general del problema

El Deep Learning se aplica cada vez más en la toma de decisiones y problemas analíticos. Como consecuencia, existe una demanda creciente por la capacidad de explicar la predicción de modelos. Una técnica de explicación es mediante la utilización de los valores de Shapley (SHAP: SHapley Additive exPlanations) como medida de agregación para determinar la influencia de las características individuales en la predicción del modelo colectivo. Los SHAP values presentan propiedades muy ventajosas a la hora de analizar este tipo de modelos. Propuesta: Programación de una herramienta que, de manera generalizada y parametrizada, permita interpretar modelos usando estas SHAP values, dando una mejor intuición de cómo el modelo ha llegado a la solución.

1.2. Motivación personal

Mi principal motivación para escoger esta propuesta es la posibilidad de crear una herramienta generalizada que permita analizar los factores más relevantes en un problema analítico, problema cada vez más frecuente en el Machine Learning. Además, el concepto de "Caja negra", propio del Deep Learning, es algo que encuentro muy interesante y creo que su análisis (Explainability) es fundamental a la hora de obtener soluciones robustas e interpretables. Por último, dentro de los ámbitos de la ciencia de datos, siento gran predilección por el Deep Learning y aspiro a que sea un campo muy presente en futuros proyectos o empleos. En este aspecto, disponer de una herramienta capaz de analizar los factores más relevantes demuestra ser muy útil al conferir una mayor velocidad y completitud en el análisis de futuros modelos.

1.3. Definición de los objetivos

El objetivo principal es la programación de una herramienta de soporte al análisis de los factores más relevantes en un problema analítico. La técnica empleada se basará en la interpretación de modelos usando SHAP values, permitiendo desglosar el modelo para mostrar el impacto de cada característica. Los objetivos secundarios tendrán que ver con todo lo que rodea a la herramienta:

- Teoría: Establecer unas bases teóricas que ayuden a comprender cómo funcionan las SHAP values.
- Programación: Elaborar el código con buenas prácticas de programación y de forma generalista y parametrizada, de modo que pueda ser utilizado en casos analíticos de diversa índole.
- Uso práctico: Se comprobarán y mostrarán las funcionalidades implementadas en la herramienta mediante su aplicación en datasets o casos analíticos.
- Visualización: La interpretación de los casos prácticos se realizará mediante visualizaciones que permitan comprender de manera clara e intuitiva los resultados arrojados por la herramienta.
- Otras técnicas: En caso de que la herramienta quede incompleta o sea positivo aportar más información, se buscará implementar otras técnicas (Análisis de causalidad mediante redes bayesianas, métodos de cálculo de intervalos de confianza...). Además de posibles funcionalidades adicionales relevantes.

1.4. Descripción de la metodología

La metodología 'Agile' será la empleada en el desarrollo del proyecto a la hora de establecer un orden en la elaboración del trabajo. La metodología Agile se especializa en el sector del software y de las nuevas tecnologías, por lo que se adaptará en gran medida a nuestras necesidades. Los principios en los que se basa este modelo de gestión nos permitirán una mejor organización y capacidad de adaptación frente a circunstancias cambiantes. Además, establece plazos de entregas por semanas con los que nos será posible medir el progreso y llevar a cabo un desarrollo sostenible. También facilita implementar nuevos requisitos o mejoras en el proyecto, siempre buscando la excelencia técnica y un buen diseño, justo lo que buscamos en este trabajo.

La metodología Agile introduce el concepto "Scrum", basado en entregas parciales y regulares del producto final, algo positivo en proyectos con requisitos cambiantes. La metodología

Scrum se desarrolla en iteraciones denominadas “sprints”. La planificación de las iteraciones desarrolla la siguiente serie de fases:

- Selección de requisitos que se quieren alcanzar.
- Planificación de la iteración: elaboración de la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados.
- Ejecución de la iteración (Sprint).
- Inspección y adaptación.

Al final de cada sprint se lleva a cabo una labor de inspección y revisión del trabajo realizado, en la que se recibe ‘feedback’ sobre las nuevas implementaciones y el estado actual del proyecto. En este caso, el encargado de realizar el feedback será el tutor del trabajo: Roger Pros Rius.

1.5. Planificación

Las fases de planificación del proyecto irán ligadas a las fechas de entrega establecidas por la UOC. Cada una de las etapas del trabajo tendrán que cumplir con, al menos, las expectativas y requisitos solicitados en cada prueba de evaluación continua (PEC).

Las fases tendrán duraciones diferentes según el volumen de trabajo estimado para completar las tareas incluidas en cada una de ellas. Tendremos 5 fases en total y serán las siguientes:

FASE 1 (11 días). Definición y planificación:

- Determinar la temática del trabajo.
- Justificar el interés y/o relevancia del trabajo.
- Especificar qué se quiere conseguir al finalizar el TFM.
- Definir los objetivos principales y secundarios.
- Establecer una planificación temporal del proyecto.
- Explicar la motivación personal del estudiante.

FASE 2 (21 días). Estado del arte o análisis de mercado del proyecto:

- Justificar con evidencias científicas el trabajo o línea de investigación escogida.
- Buscar bibliografía adecuada para el trabajo o investigación a desarrollar.
- Refinar los objetivos parciales definidos en la actividad anterior.
- Identificar la metodología y técnicas de generación de datos a utilizar.
- Redactar documentos siguiendo los formalismos científicos establecidos.
- Organizar la información de manera coherente.

FASE 3 (63 días). Diseño e implementación del trabajo:

- Realizar las tareas propias para el desarrollo del proyecto y/o investigación.
- Seguir la planificación y la metodología definidos para la investigación.

FASE 4 (13 días). Redacción de la memoria:

- Documentar y justificar el desarrollo y el resultado del trabajo.
- Redactar el documento científico que integrará la memoria del trabajo siguiendo los formalismos establecidos.

FASE 5 (7 días). Presentación y defensa del proyecto:

- Realizar una presentación oral del trabajo escrito sintetizado.

Capítulo 2

Análisis del ámbito y estado del arte

2.1. Estado actual del Deep Learning y XAI

El Deep Learning fundamenta la base de nuestro trabajo por lo que, pese a no ser el objetivo central del proyecto, será conveniente describir su funcionamiento básico y el problema de explicabilidad al que se ve sometido actualmente. El DL es una forma de aprendizaje automático que permite a los modelos computacionales aprender de la experiencia y comprender el mundo en términos de una jerarquía de conceptos. Mediante esta jerarquía el modelo es capaz de aprender conceptos complicados construyéndolos a partir de otros más simples.

En nuestro caso nos centraremos principalmente en las redes neuronales multicapa (NN). Las redes neuronales son modelos computacionales que alcanzan un rendimiento de vanguardia en una amplia gama de aplicaciones. Desde un punto de vista técnico, las NN se componen de capas sucesivas de 'neuronas' o nodos, elementos computacionales simples lineales o no lineales, que conectan las características de entrada (input) a la variable objetivo (target). Cada nodo de una capa intermedia recopila y agrega las salidas de la capa anterior y luego produce una salida propia, pasando el valor agregado a través de una función (llamada función de activación). A su vez, estos valores se pasan a la siguiente capa y este proceso continúa hasta que se alcanza la capa de salida (output). Podemos ver un ejemplo de red neuronal en la siguiente figura, Fig. 2.1.

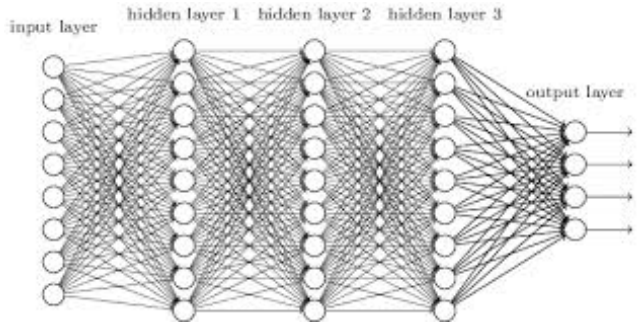

Figura 2.1: Ejemplo de Red Neuronal Multicapa NN. [7]

El principal problema que presentan este tipo de modelos radica en su arquitectura y régimen de aprendizaje altamente complejos, lo que dificulta en gran medida su transparencia, al menos cuando vamos más allá de los modelos simples, como el perceptrón de una sola capa. Esto ha llevado al desarrollo de métodos de Explainable Artificial Intelligence (XAI, Inteligencia Artificial Explicable) específicos para las NN. La mayoría de estos métodos entran dentro de la categoría de simplificación de modelos o relevancia de características.

El creciente despliegue de sistemas de inteligencia artificial basados en modelos DL en labores de alto riesgo, ha llevado ligado consigo un aumento en las demandas a estos sistemas para proporcionar explicaciones sobre sus predicciones. Las explicaciones pueden servir a los usuarios para obtener información sobre el proceso de toma de decisiones del sistema, que es un componente clave para fomentar la confianza y la seguridad en la IA.

Sin embargo, la mayoría de técnicas de Deep Learning responsables de gran parte de los avances en IA, no son fácilmente explicables incluso por expertos en la materia. Su compleja topología dificulta su interpretación, ya que no está claro cómo interactúan las variables entre sí o qué tipo de características de alto nivel podría haber elegido la red para tomar sus decisiones. Además, la comprensión teórica/matemática de sus propiedades no ha sido suficientemente desarrollada, convirtiéndolos en los denominados modelos virtuales de caja negra.

Esto ha llevado a una creciente labor de investigación centrada en la interpretabilidad o explicabilidad de las técnicas de aprendizaje automático.

2.2. Técnicas de explicabilidad de modelos Deep Learning

La explicabilidad del Deep Learning es un campo de investigación en rápido crecimiento en el que han surgido una gran variedad de métodos. Los diferentes algoritmos disponibles pueden atender mejor a unas necesidades que a otras, que pueden buscar distintos tipos de explicaciones

(p. ej.: basadas en características, basadas en instancias, basadas en lenguaje...). Por tanto, es importante comprender las diversas formas de explicaciones disponibles y las preguntas que cada una puede abordar.

En este apartado presentaremos las técnicas de aplicación general más habituales a la hora de abordar el problema de explicabilidad del Deep Learning [7]. Las técnicas serán flexibles, sin dependencia en la arquitectura intrínseca del modelo, por lo que fundamentarán sus operaciones en relacionar la entrada de un modelo con sus salidas. Los tipos de explicación más frecuentes son: simplificación de modelos, relevancia de características y visualizaciones.

Simplificación de modelos

La técnica más popular en esta categoría son las Local Interpretable Model-Agnostic Explanations (LIME, Explicaciones locales interpretables de modelos agnósticos) [17]. LIME aproxima localmente un modelo opaco, en el área circundante de la predicción que estamos interesados en explicar, construyendo un modelo lineal o un árbol de decisión en torno a las predicciones del modelo opaco, utilizando el modelo resultante como sustituto para explicar el más complejo. Además, este enfoque requiere una transformación de los datos de entrada en una 'representación interpretable', de modo que las características resultantes sean comprensibles, independientemente de las características utilizadas por el modelo.

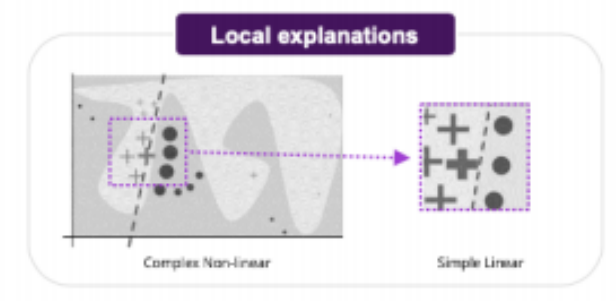

Figura 2.2: Ejemplo gráfico de explicación por simplificación de modelos.. [7]

Una técnica similar, denominada anclajes [18], también busca aproximar localmente el modelo, pero esta vez empleando reglas 'if-then' fáciles de entender, que anclan las decisiones del modelo. Las reglas tienen como objetivo capturar las características esenciales, omitiendo el resto, por lo que resulta en explicaciones menos completas que con LIME.

Otro enfoque de simplificación se introduce en [13], donde los autores exploran una forma de aprender reglas en Conjunctive Normal Form (CNF, Forma Normal Conjuntiva) o Disjunctive Normal Form (DNF, Forma Normal Disyuntiva). Suponiendo que todas las variables son binarias, el algoritmo construye un modelo de clasificación que intenta explicar las decisiones del modelo complejo utilizando sólo dichas reglas proposicionales. Estos enfoques tienen el beneficio

adicional de generar un conjunto de reglas simbólicas que se pueden explicar por defecto, así como también pueden ser utilizadas como modelo predictivo.

En [15] se introduce otra perspectiva de simplificación. En este trabajo, el objetivo es aproximar un modelo opaco usando un árbol de decisión, con la novedad de que, primero, el conjunto de datos de entrenamiento es dividido en instancias similares. Siguiendo este procedimiento, cada vez que se inspecciona un nuevo elemento, el árbol responsable de explicar instancias similares será utilizado, lo que resultará en un mejor rendimiento local.

Los autores de [6] introducen un proceso de extracción del modelo por aproximación del modelo complejo a uno transparente. El enfoque propuesto utiliza las predicciones de un modelo de caja negra para construir un árbol de decisiones (greedy), con el fin de obtener algunas ideas sobre el original a partir de inspeccionar el modelo sustituto. La simplificación se aborda desde una perspectiva diferente en [21], donde se presenta un enfoque para 'destilar' y auditar modelos de caja negra. En conjunto, el enfoque proporciona una forma de inspeccionar si un conjunto de variables es suficiente para recrear el modelo original, o si se requiere información adicional para lograr la misma precisión.

Recientemente, ha habido un desarrollo considerable en las llamadas counterfactual explanations [22]. Aquí, el objetivo es crear instancias lo más cercanas posible a la instancia que deseamos explicar, pero de manera que el modelo clasifique la nueva instancia en una categoría diferente. Al inspeccionar este nuevo punto de datos y compararlo con el original, podemos obtener información sobre lo que el modelo considera como los cambios mínimos en el elemento original para cambiar su decisión. Un ejemplo simple es el caso de un solicitante a quien se le negó su solicitud de préstamo, y la explicación podría decir que si hubiera tenido un contrato permanente con su empleador actual, se habría aprobado el préstamo.

Relevancia de características

Una de las contribuciones más populares aquí, y en XAI en general, es la de SHAP (SHapley Additive exPlanations) [16]. El objetivo en este caso es construir un modelo lineal alrededor de la instancia que se va a explicar y luego interpretar los coeficientes como la importancia de la característica. Esta idea es similar a LIME, de hecho LIME y SHAP están estrechamente relacionados, pero SHAP viene con un conjunto de buenas propiedades teóricas que lo hacen más consistente. Su base matemática tiene sus raíces en la teoría de juegos de coalición, específicamente en los valores de Shapley [1]. Como indicábamos anteriormente, nuestro trabajo girará fundamentalmente alrededor de esta técnica, en la que profundizaremos con más detalle en el siguiente apartado. De forma similar, en [24] los autores proponen medir la importancia de una característica usando su valor de Shapley, pero el objetivo de la función, así como el enfoque de optimización, no es lo mismo que en SHAP.

Figura 2.3: Ejemplo gráfico de explicación por relevancia de características. [7]

Una estrategia diferente es considerada en [2], donde se presenta una amplia variedad de medidas para abordar la cuantificación del grado de influencia de las entradas en las salidas. La Quantitative Input Influence (QII) propuesta mide la cuenta de entradas correlacionadas, que cuantifica la influencia estimando el cambio en el desempeño cuando se usa el conjunto de datos original frente a cuando se usa uno donde la característica de interés es reemplazada por una cantidad aleatoria.

Otro enfoque que se basa en permutaciones de características aleatorias se puede encontrar en [4]. En este trabajo, se introduce una metodología para aleatorizar los valores de una característica, o un grupo de características, en función de la diferencia entre el comportamiento del modelo al hacer predicciones para el conjunto de datos original y cuando hace lo mismo para la versión aleatorizada. Este proceso facilita la identificación de variables importantes o interacciones de variables que el modelo ha recogido.

También se pueden encontrar formas adicionales de evaluar la importancia de una característica, como la de [5]. Los autores introducen una metodología para calcular la importancia de las características, transformando cada característica en un conjunto de datos, por lo que el resultado es un nuevo conjunto de datos donde se ha eliminado la influencia de una determinada característica, lo que significa que el resto de los atributos son ortogonales a él. Al utilizar varios conjuntos de datos modificados, se desarrolla una medida para calcular una puntuación, basada en la diferencia en el rendimiento del modelo en los distintos conjuntos de datos.

Visualizaciones

Algunos enfoques populares de visualizaciones se pueden encontrar en [9], donde se presenta una serie de varios gráficos. En [10] se discuten técnicas adicionales, donde se introducen algunos enfoques nuevos de Sensitivity Analysis (SA).

Figura 2.4: Ejemplo gráfico de explicación por visualización. [7]

En [3] tenemos gráficos del tipo Individual Conditional Expectation (ICE, Expectativa Condicional Individual), los cuales operan en nivel de instancia, describiendo la frontera de decisión del modelo como una función de una sola característica, con el resto de ellos permaneciendo fijos. Otro tipo de gráficos, descritos en [12], son los basados en la Partial Dependence (PD, Dependencia Parcial), en los que se traza la frontera de decisión del modelo como una función de una sola característica, pero esta vez las características restantes se promedian, por lo que esto muestra el efecto promedio.

En definitiva, las visualizaciones proporcionan una forma de utilizar herramientas gráficas para inspeccionar ciertos aspectos de los modelos, como por ejemplo sus fronteras de decisión. En la mayoría de casos son relativamente fáciles de interpretar para todo tipo de audiencias. Sin embargo, al recurrir a las visualizaciones, muchos de los enfoques propuestos hacen suposiciones sobre los datos (como la independencia), que pueden no ser válidos para la aplicación en particular, quizás distorsionando los resultados.

Elección del enfoque

Los enfoques presentados en esta sección son indicativos del rango de diversos ángulos de explicabilidad que pueden ser considerados dentro del campo. Por ejemplo, los enfoques de relevancia de características brindan información midiendo y clasificando cuantitativamente la importancia de una característica, los enfoques de simplificación de modelos construyen modelos relativamente simples como sustitutos de los opacos, mientras que las explicaciones visuales inspeccionan la comprensión interna del problema de un modelo a través de herramientas gráficas. En este punto debemos tener en cuenta que elegir la técnica adecuada para la aplicación depende exactamente del tipo de información que el usuario desea obtener, o tal vez del tipo de explicaciones que se siente más cómodo interpretando.

En nuestro caso nos centraremos en técnicas de interpretabilidad de modelos basadas en las SHAP, pues, además de su especialización en la extracción de características relevantes, también incluye elementos de visualización que permiten una mejor comprensión del modelo.

2.3. SHAP

SHAP (SHapley Additive exPlanations) [16] es un método para explicar predicciones individuales. SHAP se basa en los valores de Shapley [1] (Shapley values). Describamos brevemente la idea detrás del funcionamiento de estos valores de Shapley. Una predicción se puede explicar asumiendo que el valor de cada característica de la instancia es un 'jugador' en un juego en el que la predicción es el 'pago'. Los valores de Shapley, un método de la teoría de juegos de coalición, nos dice cómo distribuir equitativamente el 'pago' entre las características.

El objetivo de SHAP [19] es explicar la predicción de una instancia x calculando la contribución de cada característica a la predicción. El método de explicación SHAP calcula los valores de Shapley a partir de la teoría de juegos de coalición. Los valores de Shapley nos dicen cómo distribuir equitativamente la predicción entre las funciones. Un jugador puede ser tanto un valor de característica individual (p. ej. datos tabulares), como un grupo de valores de características (p. ej. una imagen). La principal innovación que SHAP aporta es que la explicación del valor de Shapley se representa como un método de atribución aditiva de características, un modelo lineal. Esa vista conecta LIME [17] y Shapley Values. SHAP especifica la explicación como:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{2.1}$$

donde g es el modelo de explicación, $z' \in \{0,1\}^M$ es el vector de coalición, M es el tamaño máximo de coalición y $\phi_0 \in \mathbb{R}$ es la atribución de característica para una característica j, los valores de Shapley.

Los valores de Shapley son la única solución que satisface las propiedades de eficiencia, simetría, dummy y aditividad (Apéndice [16]). SHAP también los satisface, ya que calcula los valores de Shapley. Por lo general, no entraremos en muchos más detalles sobre la teoría matemática detrás de SHAP, centrándonos en su aplicación práctica y programación informática, fundamentalmente.

KernelSHAP: un enfoque alternativo de estimación basado en kernel para valores de Shapley inspirado en modelos sustitutos locales.

TreeSHAP: un enfoque de estimación eficiente para modelos basados en árboles.

2.4. G-SHAP

Sin embargo, no es posible responder a todas las preguntas que nos pueden surgir a la hora de interpretar un modelo únicamente explicando cuánto contribuye cada característica al resultado final. Con el objetivo de ampliar el rango de preguntas que pueden ser respondidas

surge G-SHAP (Generalized Shapley Additive Explanations) [8]. G-SHAP consiste en una reinterpretación de SHAP que permite medir la importancia de las características con respecto a alguna función de la salida del modelo. De este modo, son múltiples las funcionalidades y posibilidades añadidas que nos ofrece a la hora de interpretar nuestros modelos frente a SHAP. Las tres más relevantes son:

- Clasificación general: Supongamos que tenemos un modelo de caja negra que diagnostica COVID-19, gripe o resfriado basándose en los síntomas del paciente. G-SHAP nos ayuda a establecer por qué el paciente fue diagnosticado de una enfermedad y no de otra, y cómo difieren los síntomas que el modelo utiliza para distinguir entre parejas de enfermedades (p. ej. diferencia entre COVID-gripe y COVID-resfriado).

- Diferencias entre grupos: Supongamos que tenemos un modelo de caja negra que predice el riesgo de reincidencia de un criminal para determinar si es elegible para la libertad condicional. G-SHAP nos permite conocer por qué ciertos grupos de personas, dependiendo del sexo, raza, religión..., generalmente tienen un mayor riesgo de reincidencia que otros.

- Fallo del modelo: Supongamos que tenemos un modelo de caja negra que predice el producto interior bruto (PIB) basándose en variables macroeconómicas. G-SHAP habilita la posibilidad de comprender por qué el modelo falló en la predicción de un evento específico, como por ejemplo la Gran Recesión de 2008-2009.

En todos los modelos descritos, SHAP únicamente habría sido capaz de responder a por qué el modelo ha llegado a determinada conclusión calculando la importancia de los atributos con respecto al resultado obtenido. G-SHAP nos permite ampliar las fronteras produciendo tres nuevos tipos de explicaciones que las técnicas actuales no consiguen proporcionar limpiamente: explicaciones generales de clasificación, explicaciones de diferencias entre grupos y explicaciones de fallos.

Sin embargo, las técnicas de explicabilidad como G-SHAP pueden tener impactos negativos. Por ejemplo, aumentar la transparencia del modelo podría exacerbar la amenaza de ataques de la competencia. También es posible que ciertos tipos de explicaciones, como explicaciones de fallos del modelo, podrían disminuir la confianza del usuario en algoritmos confiables al llamar la atención sobre errores poco frecuentes.

Capítulo 3

Arquitectura del modelo

3.1. Modelos de evaluación

En nuestras dos primeras labores de clasificación, además de implementar un modelo de aprendizaje profundo, entrenaremos un conjunto de modelos que nos ayudarán a evaluar y comparar el funcionamiento de las SHAP values. Los modelos estarán basados en la combinación secuencial de clasificadores base similares y nos permitirán extraer de forma directa la importancia de los predictores utilizados en la clasificación. De este modo, podremos comparar la relevancia de atributos generada por este tipo de modelos y la generada a partir de la aplicación de las SHAP values a nuestro modelo neuronal.

Para simplificar este proceso, usaremos la librería *sklearn* y utilizaremos el comando `model.fit()` para entrenar los modelos mientras registramos su precisión y mostramos la importancia que asignan a cada atributo del dataset.

3.1.1. Modelo Random Forest

Los modelos **Random Forest** (`RandomForestClassifier`) [14] son aquellos integrados por un conjunto de árboles de decisión individuales, los cuales han sido entrenados usando muestras ligeramente distintas de los datos de entrenamiento (*bagging*). La predicción final se genera a partir de la combinación de las predicciones de todos los árboles individuales que constituyen el modelo.

Mediante este procedimiento nos será posible medir la importancia relativa de cada variable, estimando el error cometido cuando se altera dicha variable, permutando de forma aleatoria sus valores en el conjunto de test. El porcentaje de error se compara al estimado con el conjunto de test sin dicha permutación aleatoria, de forma que es posible medir el impacto de dicha variable, dado que si el error aumenta cuando se permuta una variable, esto quiere decir que la variable

es relevante para el problema que se está resolviendo. De este modo, podremos comparar la relevancia de atributos entre este modelo y la generada por las SHAP values.

3.1.2. Modelo Gradient Boosting

Los modelos **Gradient Boosting** (`GradientBoostingClassifier`) [14], al igual que los modelos Random Forest, están constituidos por árboles de decisión individuales. Pero en este caso, los árboles son entrenados de forma secuencial, de modo que los errores cometidos por árboles anteriores tratan de ser mejorados por cada nuevo árbol generado. La predicción final se genera a partir de la combinación de las predicciones de todos los árboles individuales que constituyen el modelo.

Con el objetivo de poner a prueba la extracción de atributos mediante las SHAP values, emplearemos las medidas de importancia relativa de atributos generadas por los modelos de Gradient Boosting para identificar y comparar de forma rápida y eficiente los predictores más importantes.

3.2. Perceptrón multicapa. MLPClassifier

El **Perceptrón Multicapa** (Multi-layer Perceptron, **MLP**) [20] es un algoritmo de aprendizaje supervisado que aprende una función $f : R^m \rightarrow R^o$ mediante el entrenamiento en un conjunto de datos, donde m es el número de dimensiones para la entrada y o es el número de dimensiones para la salida. Dado un conjunto de características $X = x_1, x_2, ..., x_m$ y un objetivo Y, puede aprender un función de aproximación no lineal para clasificación o regresión. Su principal característica es que, entre la capa de entrada y la de salida, puede haber una o más capas no lineales, llamadas capas ocultas.

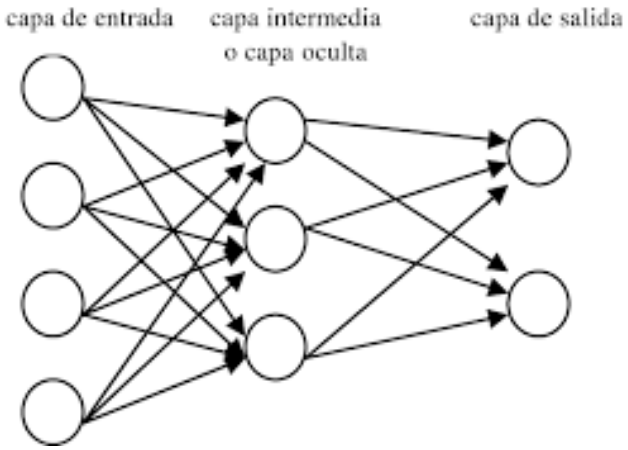

Figura 3.1: Esquema de un perceptrón multicapa simple, con una única capa oculta.

La capa de entrada, la primera por la izquierda en la Fig. 3.1, está integrada por un conjunto de neuronas $\{x_i|x_1, x_2, ..., x_m\}$ que representan los atributos de entrada. Cada neurona de la capa oculta transforma los valores de la capa anterior con una suma lineal ponderada $w_1x_1 + w_2x_2 + ... + w_mx_m$, seguida de una función de activación no lineal $g : R \rightarrow R$, como por ejemplo la función tangente hiperbólica. La capa de salida, la última por la izquierda en la Fig. 3.1, recibe los valores de la última capa oculta y los transforma en valores de salida.

En nuestros dos primeros modelos de clasificación, emplearemos el `MLPClassifier` de la librería *sklearn* para implementar un algoritmo de perceptrón multicapa entrenado mediante retropropagación (*Backpropagation*). Para la clasificación, MLP minimiza la función de pérdida de entropía cruzada (*Cross-Entropy*), dando un vector de estimaciones de probabilidad $P(x|y)$ por muestra x.

El entrenamiento del MLP puede emplear descenso de gradiente estocástico , *Adam* o *L-BFGS*. En nuestro caso emplearemos el **descenso de gradiente estocástico** (*Stochastic Gradient Descent*, SGD), el cual actualiza los parámetros utilizando el gradiente de la función de pérdida con respecto a un parámetro que necesita adaptación, es decir

$$w \leftarrow w - \eta(\alpha\frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w})$$

donde η es la tasa de aprendizaje que controla el tamaño del salto en la búsqueda de espacio de parámetros. *Loss* es la función de pérdida utilizada para la red. La función de pérdida para la clasificación es la entropía cruzada que, para casos binarios como los nuestros, se expresa como:

$$Loss(\hat{y}, y, W) = -y\ln\hat{y} - (1-y)\ln(1-\hat{y})) + \alpha \left\|W\right\|_2^2$$

dónde $\alpha \left\|W\right\|_2^2$ es un término de regularización $L2$ (también conocido como penalización) que penaliza los modelos complejos; y $\alpha > 0$ es un hiperparámetro no negativo que controla la magnitud de la penalización.

Los modelos generados por MLP serán sobre los que aplicaremos las SHAP values con el objetivo de extraer la importancia de los atributos utilizados en el proceso de clasificación. En un primer instante solamente conoceremos los parámetros y la precisión del modelo, comprobando si este responde bien a nuestro dataset y consigue clasificar correctamente nuestros conjuntos de test.

Posteriormente, la aplicación de las SHAP nos permitirá conocer que atributos son más relevantes para el modelo a la hora de tomar una decisión u otra, dándonos la capacidad de explicar cómo funciona el modelo. Finalmente, aprovecharemos esta capacidad en diversos propósitos, como la generación de confianza en el resultado del modelo, la satisfacción de los

requisitos reglamentarios, la depuración del modelo y la verificación de la seguridad del modelo, entre otras funcionalidades.

3.3. Red convolucional neuronal. VGG16

El **VGG16** [23] propuesto por el Visual Geometric Group de la Universidad de Oxford será el modelo que emplearemos en el Apartado 4.3 para la clasificación de imágenes. Se trata de una red convolucional neuronal (CNN) desarrollada durante el*ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) de 2014, un desafío para la construcción de arquitecturas de reconocimiento visual profundo. El modelo fue capaz de alcanzar una precisión en el conjunto de test del 92.7 % en el dataset de ImageNet, del que hablaremos con más detalle en el Apartado 4.3.1.

Las redes convolucionales neuronales son muy similares a los perceptrones multicapa vistos en el apartado anterior. Las CNN son una variación del perceptrón orientada a la aplicación en matrices bidimensionales, lo que las hace especialmente efectivas en tareas de visión artificial, como la clasificación y segmentación de imágenes. Expliquemos la estructura particular del la red convolucional que integra el modelo VGG16, cuyo arquitecutra podemos observar en la Fig. 3.2.

La entrada a la primera capa de la red es una imagen de tamaño fijado 224x224 píxeles y con composición del color en términos RGB. La imagen se procesa a través de un conjunto de capas convolucionales, donde los filtros actúan en un campo receptivo muy pequeño 3x3, el tamaño mínimo para capturar la noción de centro, derecha/izquierda y arriba/abajo. En una de la configuraciones también utiliza filtros de convolución 1x1, que aplican una transformación lineal a los canales de entrada. El *stride* (paso) convolucional se fija a un solo píxel; el *padding* (relleno) de la capa convolucional de entrada es tal que la resolución espacial se conserva tras la convolución, i.e. el *padding* para 3x3 capas de convolución es de un píxel. La agrupación espacial se lleva a cabo mediante cinco capas de agrupación máxima (*max-pooling*), situadas tras algunas de las capas de convolución (no todas irán seguidas de una capa de agrupación). La agrupación máxima se ejecutará sobre una ventana de 2x2 píxeles, con *stride* 2.

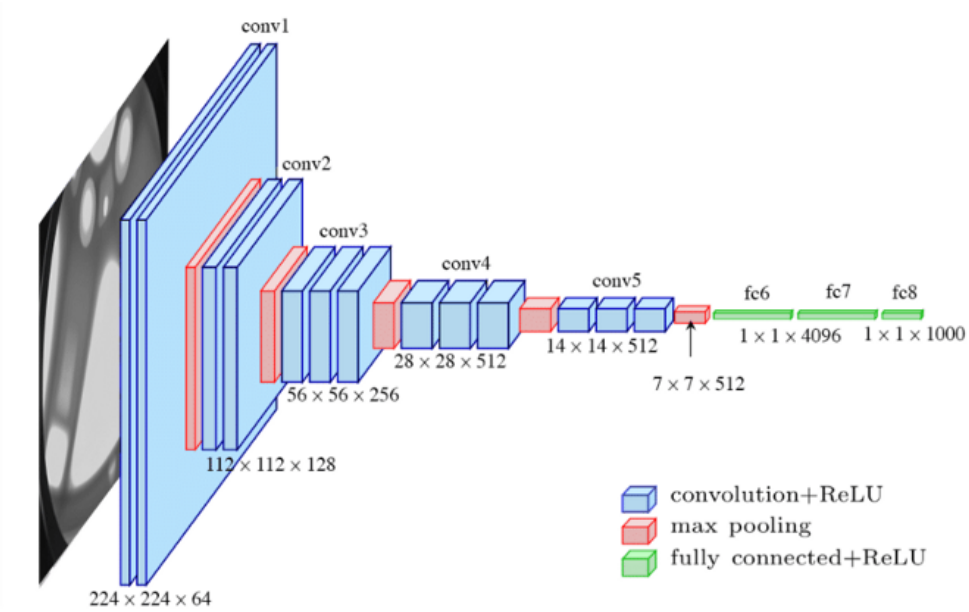

Figura 3.2: Esquema con la disposición de capas de la red convolucional neuronal para la clasificación de imágenes VGG16.

A la pila de capas convolucionales la siguen tres capas completamente conectadas (*fully connected*, FC), las cuales pueden presentar distintas profundidades en las diferentes arquitecturas. Las dos primeras tienen 4096 canales cada una, la tercera realiza una clasificación ILSVRC de 1000 vías y, por lo tanto, contiene 1000 canales (uno para cada clase). La última capa es la capa *soft-max*. La configuración de las capas totalmente conectadas es la misma en todas las redes. Todas las capas ocultas están equipadas con rectificación no lineal ($ReLU$).

Para implementar el modelo haremos uso de la librería `keras` [], que nos permitirá usar la red neuronal convolucional preentrenada VGG16, pudiendo clasificar directa y fácilmente nuestras imágenes, sin necesidad de entrenar previamente el modelo.

3.4. SHAP values. KernelExplainer

Las **SHAP** (**Explicaciones aditivas de SHapley**) son un enfoque basado en la teoría de juegos para explicar el resultado de cualquier modelo de aprendizaje automático. Conecta la asignación óptima de crédito con explicaciones locales utilizando los valores clásicos de Shapley de la teoría de juegos y sus extensiones relacionadas.

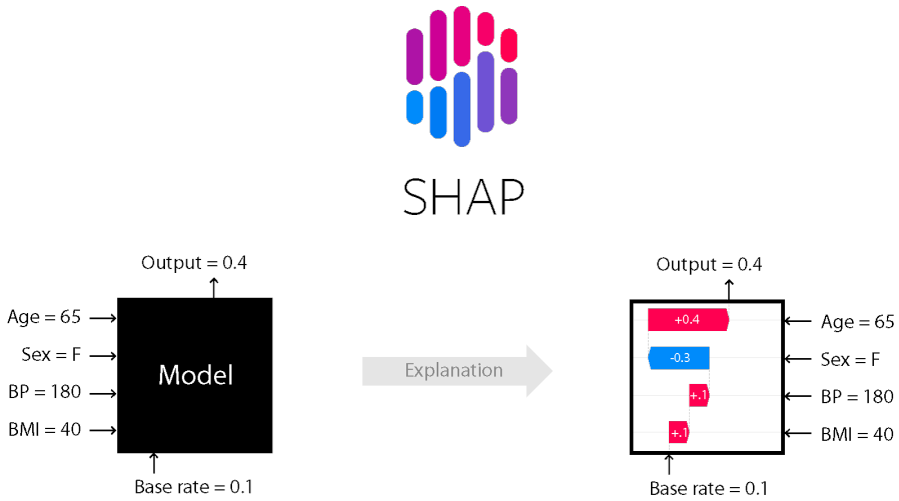

Figura 3.3: Esquema representativo de la labor de las SHAP values (caja derecha) en los modelos neuronales de caja negra (caja izquierda).

La librería `shap` ofrece diferentes explicadores básicos para abordar el problema de interpretabilidad de los modelos de aprendizaje automático. Los explicadores más habituales, entre otros, son:

- **TreeExplainer**: Utiliza algoritmos Tree SHAP para explicar el resultado de los modelos basados en conjuntos de árboles.
- **GradientExplainer:** Explica un modelo usando gradientes esperados (una extensión de gradientes integrados).
- **KernelExplainer:** Utiliza el método Kernel SHAP para explicar la salida de cualquier función. Se especializa en la aplicación a redes neuronales, lo que lo convierte en el más apropiado para nuestros modelos.
- **LinearExplainer:** Calcula los valores SHAP para un modelo lineal, contabilizando opcionalmente las correlaciones entre características.
- **PermutationExplainer:** Este método aproxima los valores de Shapley iterando a través de permutaciones de las entradas.

En nuestro caso, haremos uso del explicador **KernelExplainer** que, a su vez, hace uso del método **Kernel SHAP**. El Kernel SHAP basa su funcionamiento en una regresión lineal ponderada especial para calcular la importancia de cada característica. Los valores de importancia calculados son valores de Shapley de la teoría de juegos y también coeficientes de una regresión lineal local.

La función `KernelExplainer()` de la librería `shap` será la que nos permitirá calcular los valores de importancia. Se compone de los siguientes argumentos:

- Modelo: El modelo a explicar. La salida del modelo puede ser un vector de tamaño n muestras o una matriz de tamaño [n muestras x m clases de salida] (para un modelo de clasificación).
- Datos: Conjunto de datos de fondo para generar el conjunto de datos perturbados necesario para entrenar modelos sustitutos.

Finalmente, utilizaremos la función `explainer.shap_values()` para calcular las SHAP values que emplearemos en la interpretación de los resultados de los modelos. Se compone de los siguientes argumentos:

- Datos de salida: Conjunto de datos sobre el que explicar la salida del modelo.
- Número de muestras: Número de muestras a extraer para construir el modelo sustituto para explicar cada predicción.

Para problemas de clasificación, la función devuelve una lista de tamaño n clases (número de clases). Para nuestros modelos de clasificación binaria tendremos n clases $= 2$ (clase negativa y positiva). Cada objeto de la lista es una matriz de tamaño [n muestras x m atributos], y corresponde a los SHAP values para la clase respectiva. Para los modelos de regresión, obtenemos un único conjunto de SHAP values de tamaño [n muestras x m atributos].

Capítulo 4

Implementación del modelo

4.1. Modelo de clasificación sencillo: Iris dataset

En un primer momento, para familiarizarnos con el código y el empleo de las SHAP values, procederemos a aplicar las técnicas de extracción de información a un modelo de aprendizaje automático sencillo. En este caso trabajaremos con el clásico dataset de las flores de iris (*iris data set*). De este modo, podremos establecer las bases de funcionamiento de la herramienta y empezar conocer las capacidades de las SHAP values.

Si lo desea, puede acceder a la siguiente dirección para descargar el dataset:

https://archive.ics.uci.edu/ml/datasets/iris

4.1.1. Descripción del dataset

El conjunto de datos de flores de Iris o el conjunto de datos de *Fisher's Iris* es un conjunto de datos multivariados introducido por el estadístico y biólogo británico Ronald Fisher en su artículo de 1936 [11]. El conjunto de datos consta de 50 muestras de cada una de las tres especies de Iris (Iris setosa, Iris virginica e Iris versicolor). De cada muestra se midieron cuatro características: el largo y el ancho de los sépalos y pétalos, en centímetros.

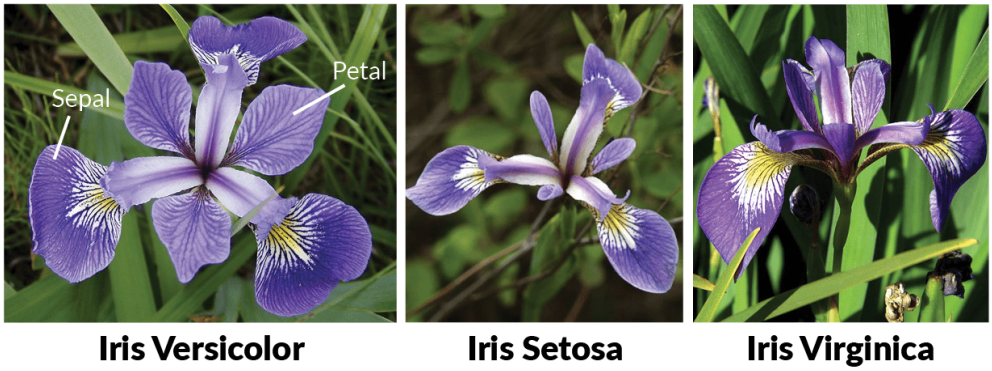

Figura 4.1: Las tres variedades de flores de iris que componen nuestro dataset. También aparecen señaladas las dos partes de la flor que usaremos en nuestro estudio, el sépalo y el pétalo.

4.1.2. Código

La implementación del código para este y el resto de modelos se puede consultar y descargar online desde la siguiente dirección de Github.

https://github.com/gyepescalero/TFM

En este apartado describiremos la implementación realizada de los modelos de clasificación para el dataset *Iris*, el cual se puede encontrar en el archivo *python/SimpleModel-Iris*. El programa se divide en cinco partes de código diferenciadas.

Análisis exploratorio de los datos.

Realizamos la exploración inicial de nuestro conjunto de datos. Las tareas propias de esta fase consistirán en mirar el tamaño de dataset y comprobar la existencia de valores nulos, calcular los principales estadísticos del dataset (número de registros, valor medio, desviación estándar, cuartiles...), ver la distribución de las clases y realizar visualizaciones de los datos que nos permitan familiarizarnos con el conjunto de datos.

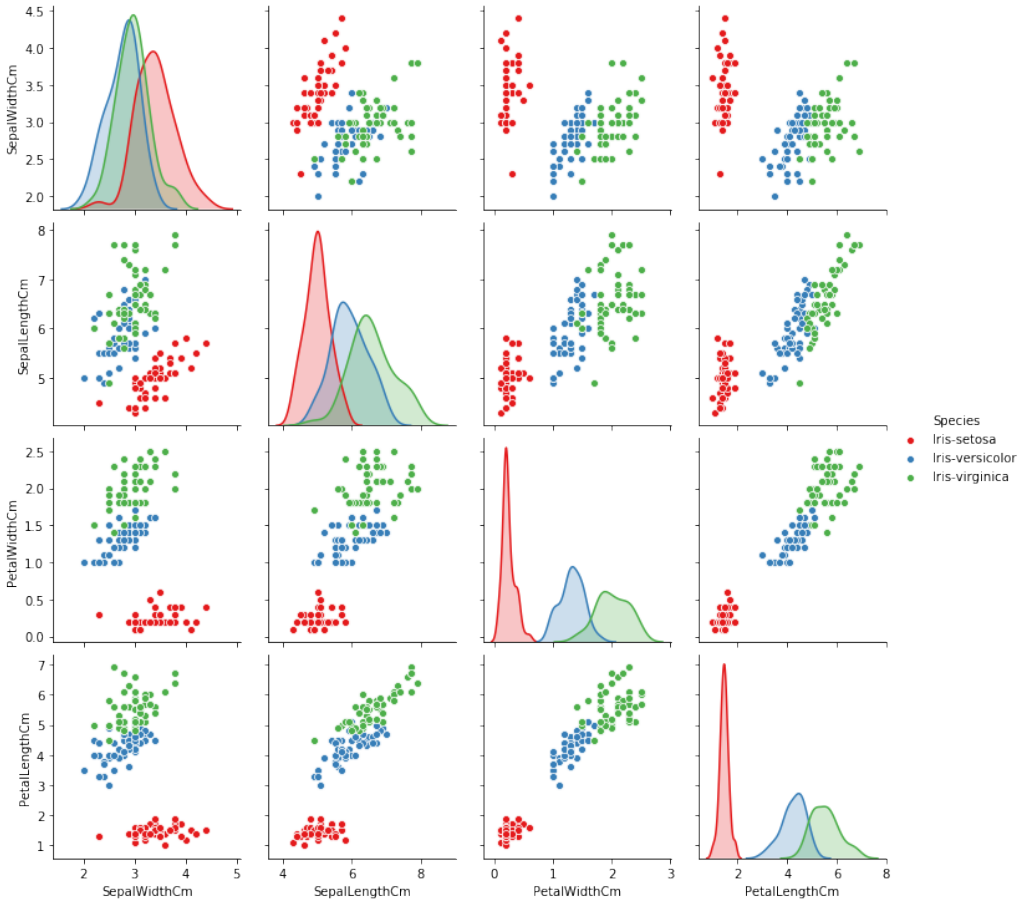

Figura 4.2: Pairplot con la distribución de las flores y especies para las posibles combinaciones de atributos del dataset Iris.

La simplicidad de este primer dataset nos permite extraer las primeras conclusiones sobre qué atributos será más conveniente usar en la clasificación de las especies. El *pairplot* de la Fig. 4.2 nos enseña la distribución de las muestras en función de las posibles combinaciones de parejas de atributos numéricos del dataset. La clase de cada muestra aparecerá reflejada por un color distintivo. En la diagonal podemos observar la distribución de la variable correspondiente para cada una de las especies.

En un primer análisis podemos ver que los atributos de anchura y longitud del pétalo son los que más diferencia presentan entre las distintas clases, permitiendo una mejor diferenciación a la hora de determinar las tres especies. En cambio, en la gráfica que enfrenta las dimensiones del sépalo, las diferencias entre especies disminuyen, endureciendo la tarea de separar unas clases de otras.

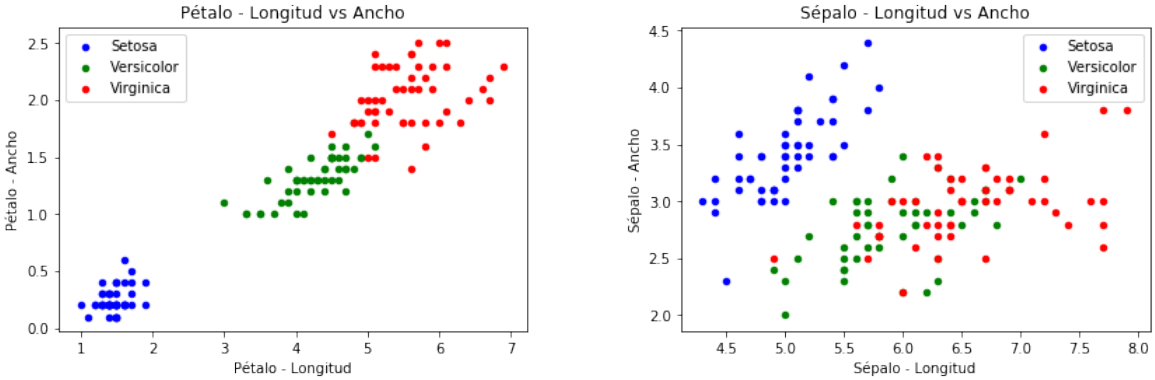

Figura 4.3: Scatterplot con la distribución de las flores y especies en función de la longitud y anchura del pétalo (izquierda), y la longitud y anchura del sépalo (derecha).

En la Fig. 4.3 podemos ver con mayor detalle las dos gráficas que hemos destacado del pairplot, consolidando lo expresado anteriormente: las dimensiones del pétalo serán de mayor utilidad en el proceso de clasificación del modelo en comparación a las dimensiones del sépalo.

En conclusión, las visualizaciones nos permitirán conocer de antemano cuáles deberían ser los atributos más relevantes en nuestros modelos. Los pesos otorgados en el proceso de clasificación a la anchura y longitud del pétalo deberían ser grandes en comparación a los pesos otorgados a la anchura y longitud del sépalo. Dicho de otro modo, los métodos de extracción de relevancias y explicabilidad de modelos deberán reflejar este comportamiento, situando la importancia del pétalo por encima de la del sépalo. Veremos si esto se cumple en los siguientes apartados.

Preparación de conjuntos de entrenamiento y test.

Antes de aplicar ningún modelo, tendremos que separar los datos entre los conjuntos de entrenamiento y test, además de estandarizar nuestros datos numéricos. En este caso, el conjunto de entrenamiento estará integrado por 120 muestras (80 %) y el de test por las 30 restantes (20 %). Para estandarizar nuestros datos haremos uso de `StandardScaler`, eliminando la media y escalando a la varianza unidad.

También es importante tener en cuenta que nuestra variable objetivo es categórica. Para facilitar la clasificación transformaremos estas etiquetas a números (*Iris-setosa*: 0, *Iris-versicolor*: 1 , *Iris-virginica*: 2)

Implementación de los modelos de comparación.

En este apartado clasificaremos los puntos empleando los modelos de evaluación descritos en el Apartado 3.1.

Comenzamos ajustando los hiperparámetros del modelo **Random Forest** mediante una

búsqueda en grid. Usaremos validación cruzada (*cross-validation*) para comprobar el rendimiento del algoritmo con cada combinación de hiperparámetros. Los hiperparámetros a ajustar serán:

- `max_depth`: profundidad máxima que pueden alcanzar los árboles.
- `n_estimators`: número de árboles incluidos en el modelo.

Una vez tenemos la selección de hiperparámetros que optimiza la clasificación de nuestro dataset, procedemos a ajustar el modelo con los datos de entrenamiento. A continuación, calculamos la precisión de nuestro modelo en el conjunto de test. Los resultados de los modelos aparecerán en el Apartado 4.1.3.

El siguiente paso será extraer los valores de relevancia de atributos establecidos por el Random Forest mediante el comando `feature_importance_`, los cuales serán empleados en el Apartado 4.1.4.

El proceso seguido con el modelo **Gradient Boosting** será muy similar al ya explicado para el Random Forest. Primero, identificaremos los hiperparámetros que generen una mejor clasificación del dataset. Los hiperparámetros ajustados son:

- `learning_rate`: reduce la contribución de cada árbol multiplicando su influencia original por este valor.
- `max_depth`: profundidad máxima que pueden alcanzar los árboles.
- `n_estimators`: número de árboles incluidos en el modelo.

A continuación, ajustamos nuestro modelo con el conjunto de entrenamiento y calculamos su precisión en el conjunto de test. Por último, extraemos los valores de relevancia de atributos establecidos por el Gradient Boosting.

Implementación de la red neuronal MLP.

La implementación del modelo neuronal **Perceptrón Multicapa MLP** no requerirá modificaciones en el dataset, pues ya ha sido procesado con anterioridad y el modelo podrá ser aplicado directamente sobre el mismo. La única dificultad será encontrar los parámetros que mejor se adapten a nuestro dataset. En este caso será fundamental escoger un número apropiado de capas ocultas y neuronas, así como un término de regularización *alpha* adecuado.

Una vez hemos establecido los mejores parámetros para nuestra red neuronal, procedemos a entrenar el modelo y a calcular su precisión con los conjuntos de entrenamiento y test, respectivamente.

Aplicación de las SHAP values.

La aplicación de las **SHAP values** se realizará únicamente sobre el modelo MLP de Deep Learning desarrollado anteriormente. Los otros dos modelos, Random Forest y Gradient Boosting, se utilizarán a modo de métrica para mostrar la eficacia y correcto funcionamiento de las SHAP values, comparando si los resultados son similares o se producen variaciones en las relevancias de los atributos de los tres algoritmos.

La aplicación de las SHAP values en este caso particular es muy sencilla, quedando resumida en las dos siguientes células.

```
import shap
shap.initjs()
```

Importamos la librería `shap` y cargamos el código de visualización JS en el cuaderno de Jupyter.

```
explainer = shap.KernelExplainer(MLP_model.predict_proba, X)
shap_values = explainer.shap_values(X)
```

Con el primer comando calculamos el explicador `explainer`, el cual utiliza el método para modelos agnósticos **Kernel SHAP** para explicar la salida del modelo de aprendizaje automático `MLP_model` introducido. Kernel SHAP es un método que emplea una regresión lineal ponderada especial para calcular la importancia de cada característica. Al tratarse de un dataset con un bajo número registros, no será necesario emplear muestras reducidas y podremos trabajar con todo el dataset sin preocuparnos por tiempos de ejecución demasiado largos.

Con el segundo comando calculamos las `shap_values`, con las que estimamos las SHAP values (diferentes a los valores de Shapley) para un conjunto de muestras, devolviendo una matriz de valores en función de la relevancia de cada atributo para cada observación. Cada fila se suma a la diferencia entre la salida del modelo para esa muestra y el valor esperado de la salida del modelo (que se almacena como atributo `expected_value` del explicador).

Una vez disponemos de las SHAP values, podremos comenzar con la **interpretabilidad de los resultados**. Esta será la sección que abordaremos con mayor detalle para cada modelo, intentando explicar los modelos de aprendizaje profundo mediante visualizaciones. El análisis exhaustivo de las SHAP values, así como de las múltiples visualizaciones, se realizará en el Apartado 4.1.4.

4.1.3. Rendimiento de los modelos

Antes de proceder a la evaluación de los resultados, expondremos el rendimiento obtenido para los tres modelos empleados para la clasificación del dataset iris. En el presente trabajo no prestaremos especial atención a la precisión o tiempo de ejecución de los modelos, pues supondremos que estos nos han sido entregados de antemano. Nuestro objetivo principal será comprender las decisiones elaboradas por los modelos de Deep Learning para que podamos

mejorar su interpretabilidad y la confianza en los mismos. En la siguiente tabla se muestran los resultados obtenidos por los diferentes modelos.

Modelos			
	Random Forest	**Gradient Boosting**	**MLP**
Precisión entrenamiento	1.0	1.0	1.0
Precisión test	0.967	0.967	0.933
Tiempo de ajuste (s)	13.03	13.37	Manual*
Tiempo de ejecución (s)	0.25	0.032	0.128
Hiperparámetros	max_depth = 11 n_estimators = 10	learning_rate = 0.01 max_depth = 9 n_estimators = 10	2 capas de 5 neuronas + 1 capa de 2 neuronas

Cuadro 4.1: Tabla con los resultados de los modelos implementados en el dataset Iris.

* El ajuste de las capas ocultas del perceptrón multicapa MLP se realizó probando de manera manual diversas combinaciones.

Del análisis de los modelos simplemente destacar que los tres presentan una precisión muy similar, superior siempre al 93 %. Este resultado, junto con las prácticamente inapreciables diferencias en sus tiempos de ajuste y ejecución, nos llevan a considerar todos los modelos como válidos para la tarea de clasificación encomendada.

4.1.4. Interpretabilidad de las SHAP values

En este apartado mostraremos la utilidad de las SHAP values a la hora de interpretar nuestros modelos. Pese a que en este caso particular nos centremos únicamente en los modelos de Deep Learning, los SHAP values pueden ser extraídos de cualquier modelo, incluidos los modelos Random Forest y Gradient Boosting.

Nuestro objetivo con los SHAP values será aumentar el grado de entendimiento de nuestro modelo de aprendizaje automático, atribuyendo valores de relevancia a nuestras variables de entrada, de modo que podamos darles un sentido analítico y comercial en el campo que tratamos.

Evaluación y comparación de la relevancia de atributos.

Comenzaremos comparando las relevancias de atributos de los modelos de evaluación Random Forest y Gradient Boosting con las generadas por las SHAP values sobre el modelo MLP.

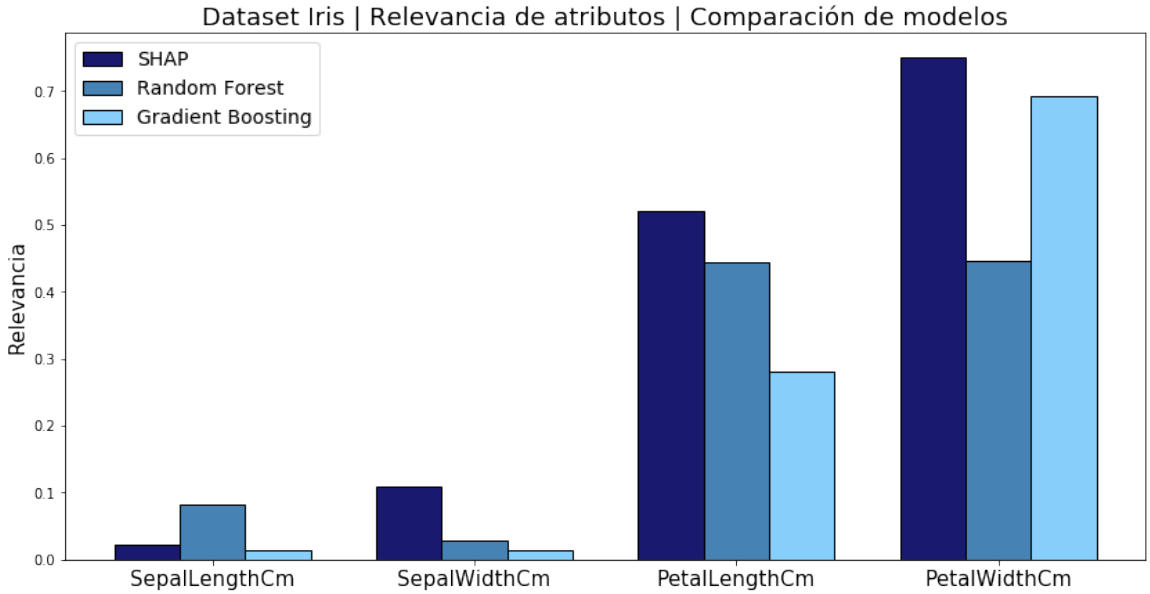

Figura 4.4: Diagrama de barras con los valores de relevancia de los atributos para los tres modelos del dataset Iris: Random Forest, Gradient Boosting y SHAP values.

Como ya anunciábamos en el análisis visual de los datos de la Fig. 4.3, los diagramas de barras muestran que los modelos presentan una clara predilección a clasificar las flores a partir de las dimensiones del pétalo. La existencia de una mayor varianza entre las especies para los atributos del pétalo lleva a los modelos a tomar estos como principal referencia a la hora de predecir la especie.

Los diferentes resultados en la selección del atributo más relevante, nos conducen a concluir que tanto la longitud como la anchura del pétalo pueden ser usados con garantías en la clasificación. Sin embargo, tanto el modelo Gradient Boosting como las SHAP values, sitúan la anchura del pétalo como la mejor variable predictiva, por lo que esta predominará en nuestros sucesivos análisis.

Relación entre los predictores y la variable objetivo. Summary plot.

El `shap.summary_plot` nos muestra las relaciones positivas y negativas de los predictores con la variable objetivo; a diferencia del gráfico del impacto promedio en la magnitud de salida de la Fig. 4.4, que únicamente nos indica el valor absoluto. Las variables aparecerán de manera descendente en función de su relevancia.

Todos los puntos del dataset aparecerán representados por una escala de color que indicará el valor del atributo correspondiente (rojo si es alto , azul si es bajo), y una posición x que reflejará el impacto de dicho valor en la decisión del modelo, es decir, su valor SHAP. La ubicación horizontal muestra si el efecto de ese valor está asociado con una predicción mayor o

menor. El código utilizado para las representaciones es el siguiente:

```
n = 0

shap.summary_plot(shap_values[n], X)
```

donde `n` indica con respecto a que clase de la variable objetivo son interpretados los SHAP values. En el primer ejemplo, Fig. 4.5, hemos introducido `n = 0`, por lo que los resultados tendrán como referencia la especie Setosa, mostrando valores positivos de x si el valor que toma el atributo ha llevado al modelo a clasificar el punto como Setosa y valores negativos si ha sido al contrario.

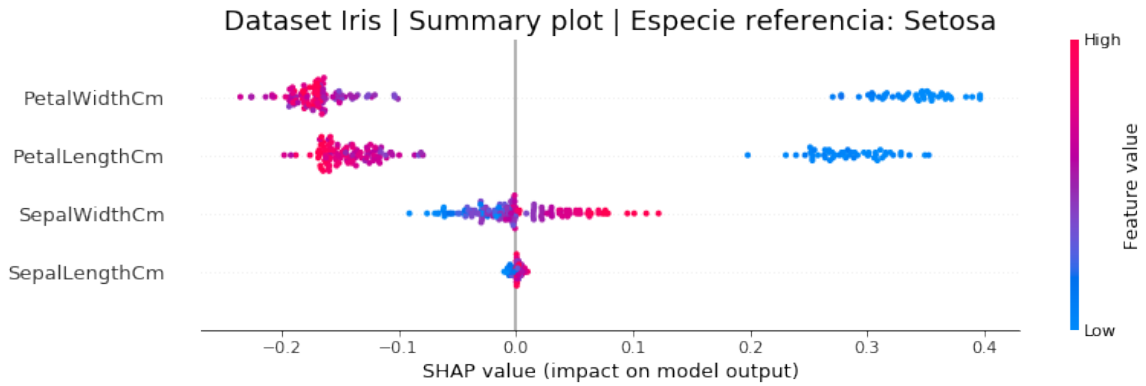

Figura 4.5: Summary plot para la especie Setosa con los valores de las SHAP values en función del impacto del valor de los atributos en la salida del modelo. Los atributos aparecen ordenados de manera descendente según su relevancia.

Del gráfico podemos concluir que unas dimensiones del pétalo pequeñas (color azul = valor del atributo bajo) tienen un impacto muy alto y positivo (valores altos y positivos en la coordenada x = SHAP values positivas) en la clasificación de las flores como especie Setosa. En cambio, los valores bajos en estos atributos tendrán el efecto contrario, favoreciendo que el modelo identifique la flor con otra especie. Las dimensiones del sépalo tendrán una influencia mucho menor en la predicción, únicamente importando algunos valores altos en la anchura del sépalo en la clasificación como Setosa.

En la siguiente visualización, cambiaremos `n` a `n = 2`, cambiando la variable objetivo a la especie Virginica, que es la que presenta una mayor diferenciación con la especie Setosa.

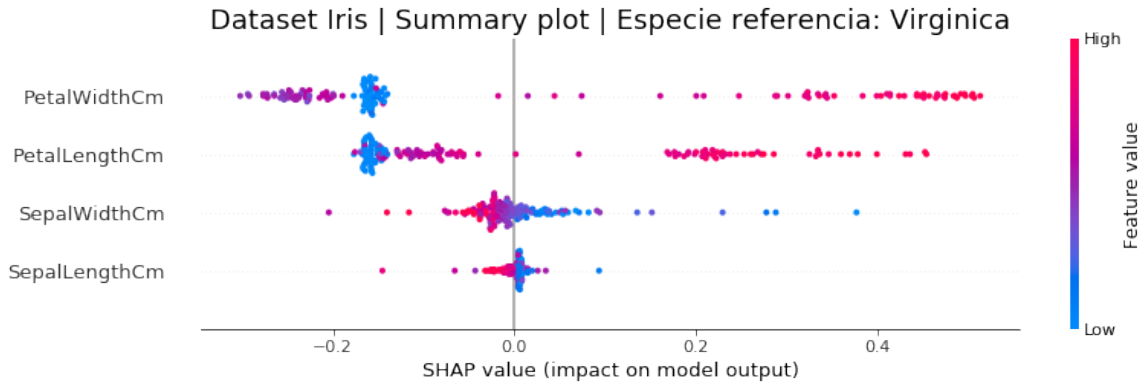

Figura 4.6: Summary plot para la especie Virginica con los valores de las SHAP values en función del impacto del valor de los atributos en la salida del modelo. Los atributos aparecen ordenados de manera descendente según su relevancia.

Los resultados son radicalmente opuestos a los de la distribución para la especie Setosa. En este caso, unas dimensiones del pétalo grandes (color rojo = valor del atributo alto) tienen un impacto muy alto y positivo (valores altos y positivos en la coordenada x = SHAP values positivas) en la clasificación de las flores como especie Virginica.

Efecto de las variables sobre el resultado previsto. Dependence plot.

El `shap.dependence_plot` nos muestra el efecto marginal que una o dos variables tienen sobre el resultado previsto. En general, nos indica si la relación entre el objetivo y la variable es lineal, monótona o más compleja, reflejando el efecto que tiene una sola característica en las predicciones realizadas por el modelo.

Las predicciones del dataset aparecen representadas por puntos en unas coordenadas concretas. La posición en el eje x nos indica el valor del atributo para la predicción. Mientras que la coordenada y nos muestra el SHAP value que se le otorga al valor que toma dicho atributo, representando cuánto ha influenciado el conocimiento del valor de esa característica en la salida del modelo para la predicción de esa muestra.

El color del punto se corresponde con una segunda característica que puede tener un efecto de interacción con el atributo que estamos graficando (por defecto, esta segunda característica se elige automáticamente). Si hay un efecto de interacción entre los dos atributos, se mostrará con una escala de coloración distinta. En este dataset sencillo no haremos uso de un segundo atributo de interacción, estableciendo `interaction_index=None`.

En los dos primeros dependence plot, Fig. 4.7, representaremos el efecto marginal de las dimensiones del sépalo en la clasificación de la especie Setosa (`n = 0`). A continuación, haremos lo propio con las dimensiones del pétalo, Fig. 4.8, y analizaremos los resultados obtenidos. El código utilizado para las representaciones es el siguiente:

```
shap.dependence_plot("SepalWidthCm", shap_values[n], X, interaction_index=None)
```

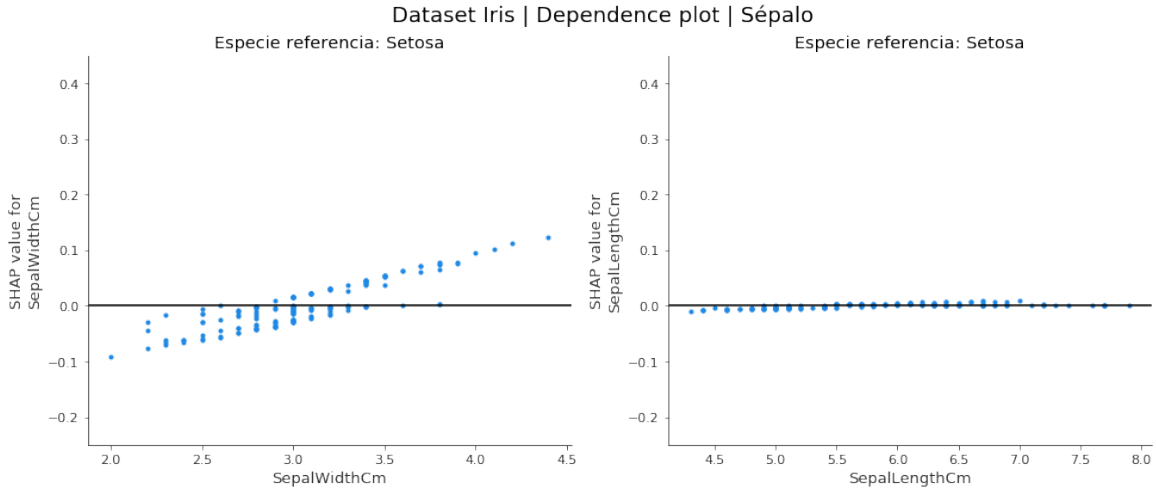

Figura 4.7: Dependence plot con la distribuciones de las SHAP values en función de los valores de los atributos anchura del sépalo (izquierda) y longitud del sépalo (derecha) para la especie Setosa.

La distribución de los puntos refleja la escasa relevancia que nuestra red neuronal otorga a las dimensiones del sépalo. Si nos fijamos en la posición en el eje y de los puntos, podemos apreciar que los SHAP values para la anchura y longitud del sépalo son muy cercanos a 0. Sólo para algunos valores muy extremos en la anchura del sépalo este llegará a ser útil en las predicciones de la especie Setosa.

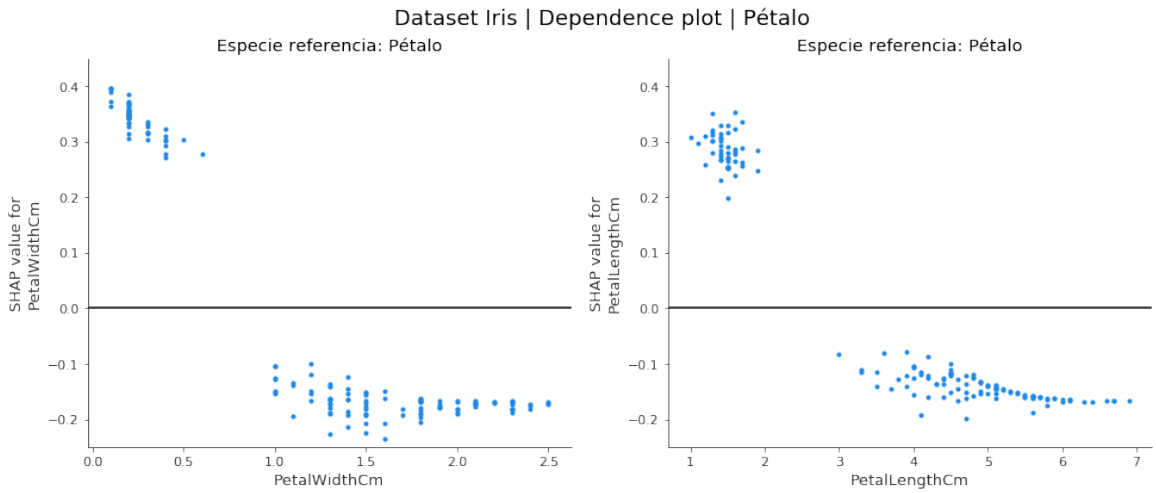

Figura 4.8: Dependence plot con la distribuciones de las SHAP values en función de los valores de los atributos anchura del pétalo (izquierda) y longitud del pétalo (derecha) para la especie Setosa.

La distribución de los puntos para las dimensiones del pétalo sí que indica la clara incidencia de estos atributos en la clasificación de las muestras. La posición en el eje y revela SHAP values destacados para la gran mayoría de puntos, lo que indica que estas características son especialmente útiles en la clasificación de la especie Setosa, caracterizada por la de ser la de menor tamaño en cuanto a anchura y tamaño del pétalo. De este modo observamos como, para valores pequeños en las variables del pétalo, el modelo se ve enormemente inclinado (valores SHAP de hasta +0.4) a etiquetar la muestra como Setosa. Justo al contrario ocurre para valores altos, donde las SHAP values se vuelven negativas.

A continuación, analizaremos como la longitud del pétalo influye en la clasificación del resto de especies, representando el dependence plot para `n = 1` (Versicolor) y `n= 2` (Virginica).

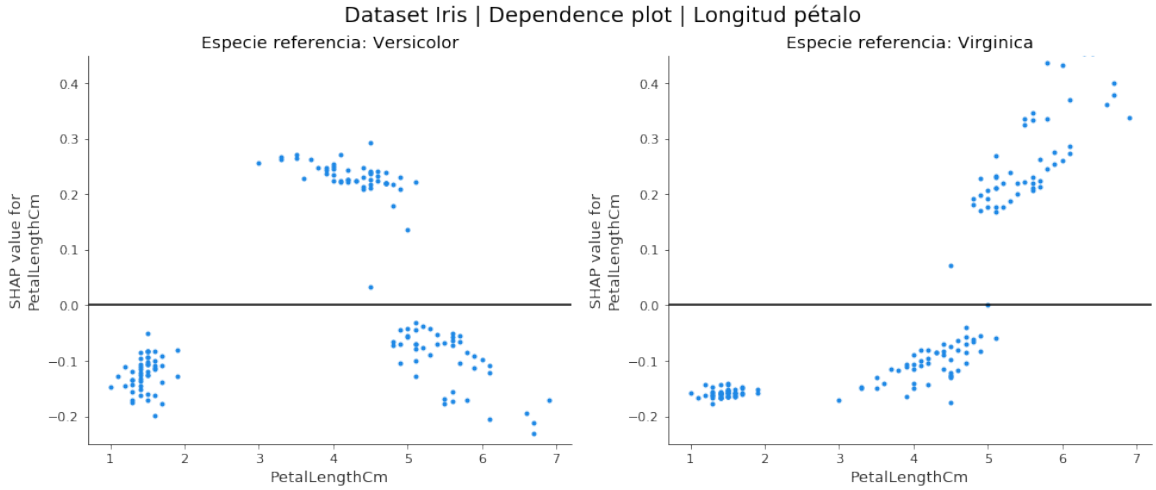

Figura 4.9: Dependence plot con la distribuciones de las SHAP values en función de los valores del atributo longitud del pétalo para las especies Versicolor (izquierda) y Virginica (derecha).

Comenzamos comentando el dependence plot correspondiente a la especie Versicolor, situado a la izquierda de la Fig. 4.9. Como sabemos de apartados anteriores, la longitud de su pétalo se sitúa entre la de las otras dos especies, lo que explica la distribución en el eje y de los puntos para valores céntricos, los cuales favorecerán la clasificación de la especie Versicolor. Para valores bajos y altos en la variable, el modelo deduce que se trata de otra especie.

La interpretación del segundo dependence plot es muy similar a la realizada en la Fig. 4.8 para la especie Setosa, excepto que con el efecto contrario: los valores altos en la longitud del pétalo llevarán a predicciones de la especie Virginica, que es la de mayor tamaño en sus dimensiones.

Análisis explicativo de predicciones individuales. Individual force plot.

El `shap.force_plot` nos permite explicar predicciones individuales dentro del dataset. Co-

mencemos explicando la clasificación del modelo para una flor de la especie Setosa (cualquiera de las 50 primeras muestras del dataset). Las SHAP values serán también interpretadas con respecto a la misma especie, `n = 0`. El código utilizado para las representaciones es el siguiente:

```
shap_values = explainer.shap_values(X.iloc[i,:])
shap.force_plot(explainer.expected_value[n], shap_values[n], X.iloc[i,:])
```

donde `i` indica el índice de la muestra cuya predicción queremos explicar. Para interpretar el resultado tendremos que tener en cuenta diversos aspectos de este tipo de gráficos:

- El valor de salida es la predicción que hace el modelo para la observación indicada. Si la predicción toma el valor 1, el modelo clasificará la observación como especie Setosa. Si el valor es 0, el modelo no clasificará la observación como especie Setosa.

- El valor base indica el valor que se predeciría si no conociésemos ninguna característica para la salida actual, es decir, la predicción media del modelo. En nuestro caso el valor base tomará el valor 1/3.

- Los atributos que favorecen la predicción son mostrados en rojo, y son aquellos que empujan el valor base hacia la derecha (hacia el valor 1 en la predicción). Los atributos que se oponen a la predicción aparecen en azul, y son aquellos que empujan el valor base hacia la izquierda (hacia el valor 0 en la predicción).

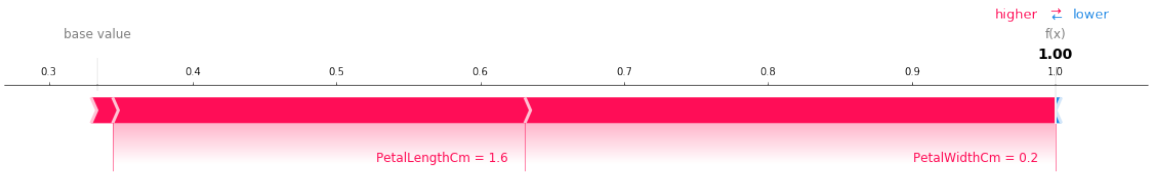

Figura 4.10: Force plot con las SHAP values de la predicción individual correspondiente a la flor de Iris número 25 para la especie Setosa.

Observamos como, para esta predicción, el resultado del valor de salida es 1, es decir, el modelo ha identificado la observación como especie Setosa. Los atributos que han empujado a una clasificación favorable son las dimensiones del pétalo, con una anchura de 0.2 y una longitud de 1.6 cm. Al tratarse de medidas pequeñas dentro de las variables, estas estarán relacionadas positivamente con la especie Setosa, la de menor tamaño. Ni la anchura ni la longitud del sépalo serán relevantes en esta predicción particular.

A continuación, explicaremos una flor de la especie Virginica (cualquiera de las últimas 50 muestras del dataset) y volveremos a interpretar su predicción con respecto a la especie Setosa, `n = 0`, esperando un resultado negativo.

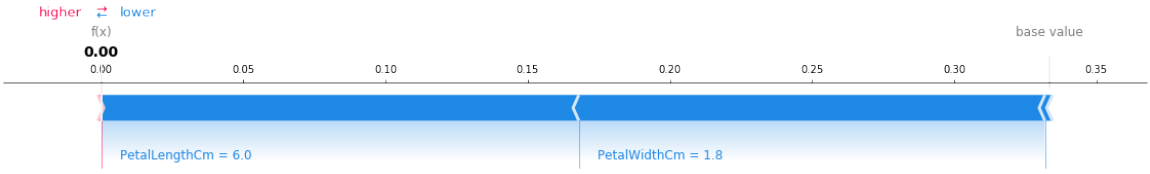

Figura 4.11: Force plot con las SHAP values de la predicción individual correspondiente a la flor de Iris número 125 para la especie Setosa.

En este caso, el valor de salida será 0, lo que nos indica que el modelo no clasificará la flor dentro de la especie Setosa. Las dimensiones del pétalo vuelven a predominar sobre las del sépalo. La anchura de 1.8 cm y la longitud de 6.0 cm empujan la predicción a una especie distinta a la Setosa, pues se trata de una flor con un pétalo de tamaño mayor al esperado.

Veamos que ocurre si ahora interpretamos la misma predicción tomando como variable objetivo la especie Virginica (`n = 2`).

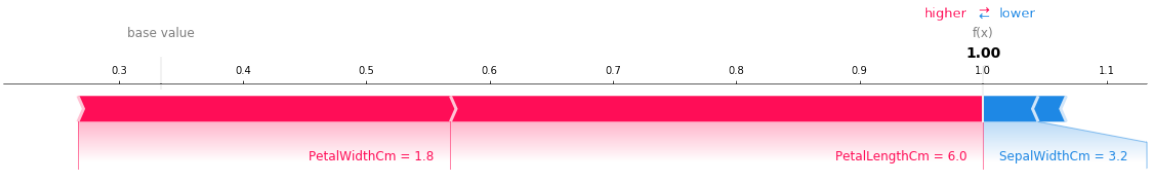

Figura 4.12: Force plot con las SHAP values de la predicción individual correspondiente a la flor de Iris número 125 para la especie Virginica.

Como era de esperar, la salida del modelo es 1, pues ahora las dimensiones del pétalo tendrán un impacto positivo en la clasificación. La especie Virginica se identifica con flores con pétalos de mayor tamaño.

Análisis explicativo de predicciones colectivas. Collective force plot.

El `shap.force_plot` también nos permite explicar de manera colectiva un conjunto de predicciones. Este gráfico integrará una muestra o el total de las observaciones individuales que

hemos visto en el apartado anterior. En nuestro caso, representaremos el dataset al completo, incluyendo los 150 registros.

Comencemos generando el gráfico tomando como referencia la especie Setosa, `n = 0`, y ordenando las observaciones por grado de similitud, es decir, agrupando las predicciones que tengan resultados parecidos. El código utilizado para las representaciones es el siguiente:

```
shap.force_plot(explainer.expected_value[n], shap_values[n], X)
```

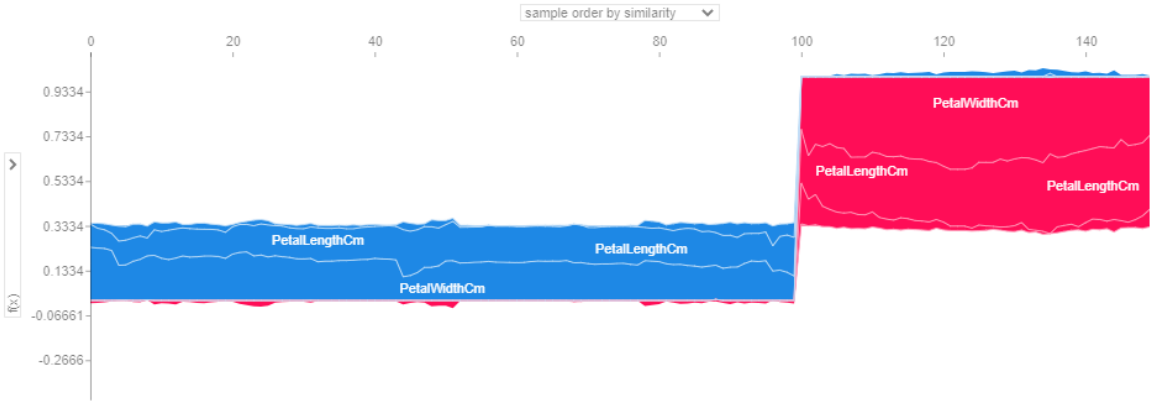

Figura 4.13: Force plot con las SHAP values de la predicción colectiva de todas las flores de Iris para la especie Setosa. Distribución ordenada por grado de similitud.

La gráfica se forma al combinar los force plots individuales para todos los datos de `X`, rotarlos 90 grados y apilarlos horizontalmente. El objetivo de la agrupación en clústeres es encontrar grupos de instancias similares. En la visualización podemos distinguir claramente dos grupos:

- Por un lado tendremos las predicciones que han tomado como valor de salida 0, representadas en azul, y que no se corresponden con la especie Setosa. Esta predicciones se caracterizarán por valores altos en las dimensiones del pétalo.

- Por otro lado tendremos las predicciones con valor de salida 1, representadas en rojo, y que se corresponden con la especie Setosa. Las observaciones de este clúster se caracterizarán por valores bajos en las dimensiones del pétalo.

Si ahora establecemos Virginica, `n = 2`, como especie objetivo y representamos las predicciones del dataset obtendremos el resultado contrario.

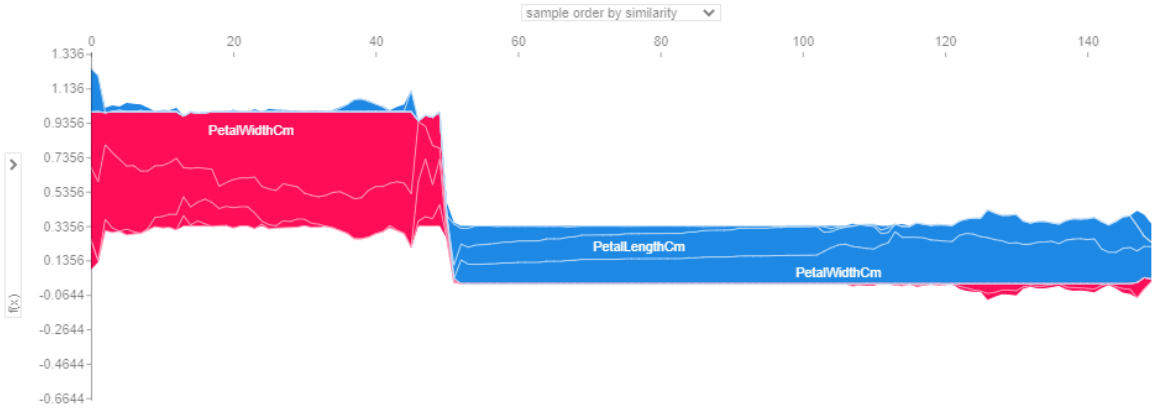

Figura 4.14: Force plot con las SHAP values de la predicción colectiva de todas las flores de Iris para la especie Virginica. Distribución ordenada por grado de similitud.

En definitiva, las conclusiones son las mismas que las extraídas en los apartados anteriores. Las dimensiones del pétalo actuarán como principales predictores del modelo neuronal, situándose muy por encima de las del sépalo. La anchura y longitud del pétalo tendrán un gran impacto en la clasificación, siendo las que realmente identifican la especie de la flor de Iris. La anchura y longitud del sépalo influirán en algunas predicciones pero en mucha menor medida.

Las SHAP values nos han permitido afianzar estas conclusiones, dándonos la posibilidad de profundizar en un dataset que, aunque fácilmente explicable, nos será muy útil de cara a interpretar las representaciones SHAP del modelo complejo del siguiente apartado.

4.2. Modelo complejo: Census Income dataset

Una vez nos hemos familiarizado con la implementación e interpretación de las SHAP values, procederemos a aplicar las técnicas en un caso de aprendizaje automático de mayor complejidad. El objetivo será resolver un problema de interpretabilidad más cercano al mundo real y de mayor impacto analítico y comercial. Para ello, desarrollaremos y analizaremos un modelo neuronal de clasificación basado en el dataset *Census Income*, también bastante popular en el mundo de la ciencia de datos. Mediante las SHAP values aumentaremos la explicabilidad y la confianza en el modelo, para finalmente extraer conclusiones útiles en el campo que tratamos.

Si lo desea, puede acceder a la siguiente dirección para descargar el dataset:

https://www.kaggle.com/uciml/adult-census-income

4.2.1. Descripción del dataset

El conjunto de datos seleccionado recopila el censo de adultos de Estados Unidos. Es un repositorio de más de 32000 entradas del año 1994, que nos da información tanto económica como social de los individuos de la muestra, como por ejemplo su raza, su nivel de estudios o los ingresos de los mismos. Así, la base de datos nos resultará útil para realizar predicciones sobre los ingresos que va a tener un individuo con unas características dadas, o bien para ver qué influencia tiene una característica concreta en los ingresos del individuo.

Los atributos de nuestra base de datos son los siguientes:

- **age**: edad del individuo. Número natural.
- **workclass:** término para presentar la situación y el ámbito laboral del individuo. Posibles valores: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov,Without-pay, Never-worked.
- **fnlwgt:** peso final. Es el número de personas que el censo cree que la entrada representa. Número natural.
- **education:** nivel de educación más alto alcanzado por el individuo. Posibles valores: Bachelors, Somecollege, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- **education.num:** nivel educativo más alto alcanzado en forma numérica. Número natural.
- **marital.status:** estado civil del individuo. Posibles valores: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- **occupation:** trabajo del individuo. Posibles valores: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- **relationship:** tipo de relación sentimental del individuo. Posibles valores: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- **race:** raza del individuo. Posibles valores: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- **sex:** sexo biológico del individuo. Posibles valores: Male, Female.
- **capital.gain:** ganancias del individuo. Número natural.

- **capital.loss:** pérdidas del individuo. Número natural.

- **hours.per.week:** horas trabajadas a la semana. Número natural.

- **native.country:** país de origen del individuo.

- **income:** salario del individuo. Variable objetivo de nuestro dataset. Dos posibles valores: $<= 50k$ ó $> 50k$.

4.2.2. Código

En este apartado describiremos la implementación de los modelos de clasificación para el dataset *Census Income* y la extracción de las SHAP values para la red neuronal. El programa completo se puede encontrar en el archivo *python/ComplexModel-CensusIncome* del proyecto Github (Apartado 4.1.2). El programa se divide en seis partes de código diferenciadas.

Limpieza de los datos.

El conjunto de datos deberá ser sometido a una limpieza para la correcta aplicación de los modelos. Las tareas de limpieza consistirán en: sustitución de datos irregulares por valores vacíos ("NaN"), reemplazo de valores vacíos por el valor más común y comprobación del tipo de cada atributo.

Análisis exploratorio de los datos.

Realizamos la exploración inicial de nuestro conjunto de datos. Primero estudiaremos si hay una variedad similar, o al menos comparable, entre los registros de cada etiqueta de la variable objetivo ïncome". Un cuarto de los registros del dataset se corresponderán la clase »50k", mientras que el resto pertenecerán a la clase «=50k". La proporción entre las clases no debería suponer un problema pues disponemos de suficientes ejemplos para cada una de ellas.

A continuación, estudiaremos por un lado las variables numéricas, 6 en total, calculando los principales estadísticos que las definen (media, desviación estándar, cuartiles, etc.). Además comprobaremos las correlaciones existentes y visualizaremos su distribución mediante histogramas.

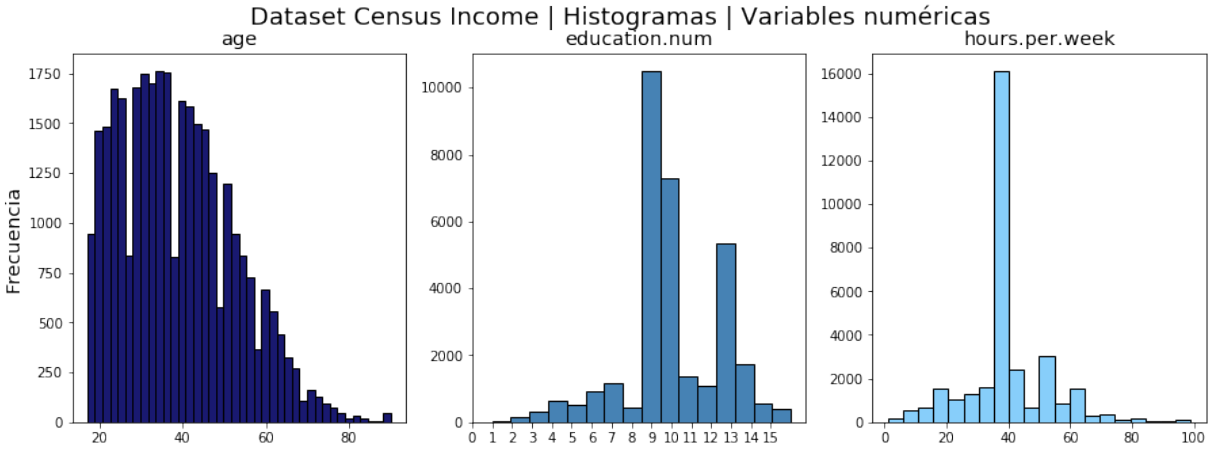

Figura 4.15: Histogramas con la distribución de las variables numéricas; edad (izquierda), nivel de educación (centro) y horas laborales semanales (derecha), para el dataset Census Income.

No se aprecia ningún valor de correlación interesante, ninguna de las parejas de variables numéricas supera el 0.15 en esta métrica. En cuanto a los histogramas, hemos representado únicamente aquellos atributos que estudiaremos en mayor detalle con las SHAP values.

Por otro lado, estudiaremos las variables categóricas, las restantes 9, representando su distribución en las distintas categorías mediante diagramas de barras.

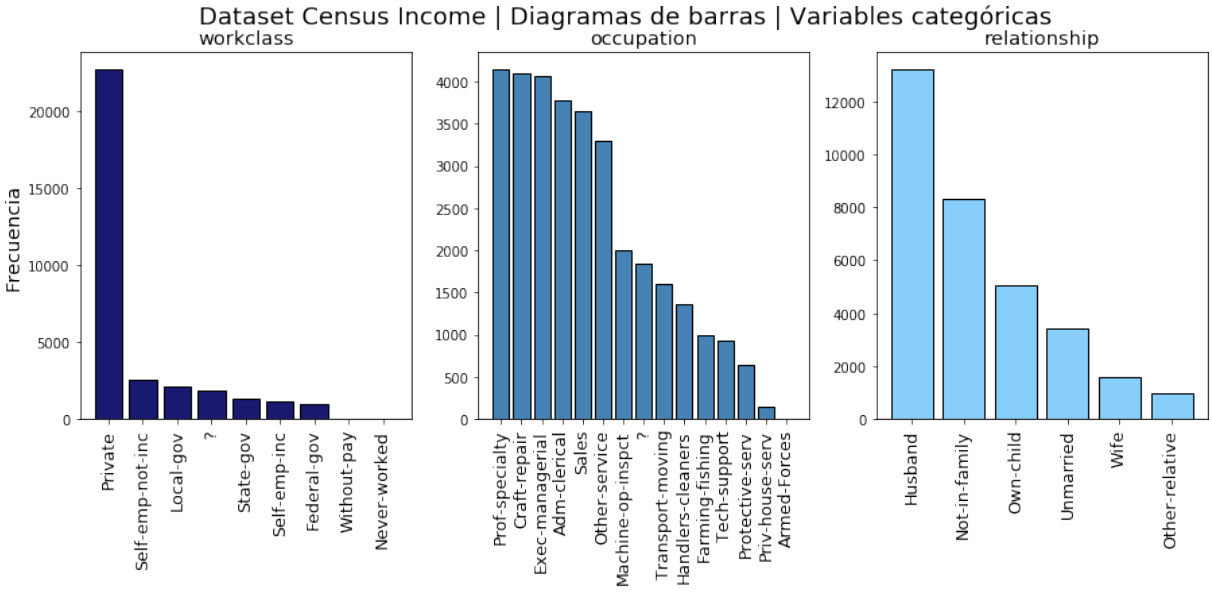

Figura 4.16: Diagramas de barras con la distribución de las variables categóricas; ámbito laboral (izquierda), profesión (centro) y estado civil (derecha), para el dataset Census Income.

Aquí también hemos escogido únicamente las variables que estudiaremos en la interpretación de las SHAP values en el Apartado 4.2.4.

Por último, exploraremos el dataset en busca de valores outliers, aquellos que escapan de la distribución esperada del atributo y presentan un comportamiento extremo. En este caso no consideraremos deshacernos de ninguno de estos valores outliers, pues confiamos en la robustez de los modelos y queremos ver como reaccionan las SHAP values ante ellos.

Cabe destacar que del análisis exploratorio no nos será posible extraer conclusiones directas como para el dataset Iris. Tendremos que apoyarnos en nuestros modelos predictivos y en las SHAP values para comprender mejor el dataset.

Preparación de conjuntos de entrenamiento y test.

Dividimos el dataset entre variables independientes y la variable objetivo o dependiente. Eliminaremos algunas de las columnas predictivas, ya sea porque se hallan implícitas de manera directa en otras columnas, o porque no tendrán relevancia en el modelo. En este caso nos desharemos de las columnas 'education', 'marital.status' y 'native.country'.

Las variables categóricas restantes serán codificadas mediante etiquetas con un valor entero entre 0 y el número de clases menos uno. Para aquellos atributos con más de una clase aplicaremos el *One Hot Encoder*, codificando las características categóricas como una matriz numérica de un solo uso. El objetivo de estas transformaciones será favorecer la clasificación de los modelos y facilitar la explicabilidad del modelo mediante las SHAP values.

Procedemos a separar los datos entre los conjuntos de entrenamiento y test, además de estandarizar nuestras variables predictivas numéricas. En este caso, el conjunto de entrenamiento estará integrado por 24420 muestras (75 %) y el de test por las 8141 restantes (25 %). Para estandarizar nuestros datos haremos uso de `StandardScaler`, eliminando la media y escalando a la varianza unidad.

Implementación de los modelos de comparación.

En este apartado clasificaremos los puntos empleando los modelos de evaluación descritos en el Apartado 3.1. El procedimiento será similar al realizado para el dataset Iris, por lo que en esta ocasión no entraremos en detalles. La implementación constará de las tres etapas habituales: ajuste de los hiperparámetros del modelo, entrenamiento del modelo y cálculo de la precisión del modelo sobre el conjunto de test. Los modelos utilizados volverán a ser Random Forest y Gradient Boosting.

Para ambos modelos de evaluación, extraeremos los valores de relevancia de atributos asociados, los cuales serán empleados en el Apartado 4.2.4.

Implementación de la red neuronal MLP.

Buscamos la estructura del perceptrón multicapa que mejor se adapte a nuestra tarea de clasificación. En este caso, la red neuronal estará constituida por tres capas ocultas; las dos

primeras estarán integradas por 20 neuronas cada una y la última por 10, generando un total de 930 parámetros para explicar el modelo.

Una vez hemos establecido los mejores parámetros para nuestra red neuronal, procedemos a entrenar el modelo y a calcular su precisión con los conjuntos de entrenamiento y test, respectivamente.

Aplicación SHAP values.

La aplicación de las **SHAP values** se realizará únicamente sobre el modelo MLP de Deep Learning desarrollado anteriormente. Los otros dos modelos, Random Forest y gradient Boosting, volverán a ser utilizados como métricas de evaluación.

La única diferencia en la implementación de las SHAP values con respecto al modelo anterior será el elevado número de registros que presenta el dataset. El cálculo del KernelSHAP es un proceso lento, lo que hace no sea práctico de usar cuando se desea calcular valores de Shapley para muchas instancias.

Para solventar el problema del coste computacional del cálculo de las SHAP values no utilizaremos todo el conjunto de datos para estimar los valores esperados, sino que nos limitaremos a explicar una muestra del conjunto de entrenamiento. La muestra se seleccionará mediante el método de agrupamiento *k-means* (k-medias), que consiste en la partición de un conjunto de n observaciones en k grupos, en el que cada observación pertenece al grupo cuyo valor medio es más cercano. De este modo, reduciremos el modelo a un conjunto de k-medias ponderadas por el número de puntos que representan. El código utilizado para aplicar el método k-means es el siguiente:

```
X_train_summary = shap.kmeans(X_train, 150)
```

En nuestro caso, nos quedaremos con un total de 150 k-medias del conjunto de entrenamiento para crear el `explainer` y usaremos todo el conjunto de prueba para calcular las SHAP values con `explainer.shap_values()`. De este modo, reduciremos las instancias de la muestra de fondo (*background data samples*), disminuyendo el tiempo de ejecución. Aún así, la complejidad de nuestra red neuronal, junto con el elevado número de atributos de entrada, nos llevará a tiempos de ejecución prolongados, de entorno a las 4 horas.

4.2.3. Rendimiento de los modelos

En la siguiente tabla se muestran los resultados obtenidos por los diferentes modelos.

Modelos			
	Random Forest	**Gradient Boosting**	**MLP**
Precisión entrenamiento	0.879	0.858	0.887
Precisión test	0.86	0.854	0.842
Tiempo de ajuste (s)	340.16	13.37	Manual*
Tiempo de ejecución (s)	1.27	6.819	191.688
Hiperparámetros	max_depth = 12 n_estimators = 50	learning_rate = 0.01 max_depth = 6 n_estimators = 100	2 capas de 20 neuronas + 1 capa de 10 neuronas

Cuadro 4.2: Resultados de los modelos implementados en el dataset Census Income.

* El ajuste de las capas ocultas del perceptrón multicapa MLP se realizó probando de manera manual diversas combinaciones.

Del análisis de los modelos simplemente destacar que los tres presentan una precisión muy similar, de alrededor del 85 %. En este caso, si que existe una diferencia notable en los tiempos de ejecución para la red neuronal, que ha visto aumentado su complejidad y número de parámetros en gran medida.

4.2.4. Interpretabilidad de las SHAP values

En este apartado emplearemos las SHAP values calculadas, así como las diferentes visualizaciones que ofrece la librería `shap`, para interpretar nuestro modelo y extraer conclusiones útiles del mismo. A la hora de abordar el análisis, plantearemos una situación real en la que se nos ha encomendado explicar el modelo neuronal de un estudio socio-económico sobre los factores más relevantes a la hora de obtener sueldos elevados.

El objetivo de nuestro trabajo será, por lo tanto, aumentar el grado de confianza en el modelo de clasificación, destapando en la medida de lo posible la caja negra que conforma el modelo neuronal implementado. Con esto en mente, deberemos emplear los resultados de las SHAP values para alcanzar conclusiones que den sentido a las predicciones obtenidas, además de obtener información relevante sobre el campo en el que trabajamos. En resumen, tendremos que aumentar la explicabilidad de nuestro modelo, mejorando su fiabilidad y consistencia.

Evaluación y comparación de la relevancia de atributos.

Comenzamos comparando las relevancias de atributos de los modelos de evaluación Random Forest y Gradient Boosting con las generadas por las SHAP values sobre el modelo MLP.

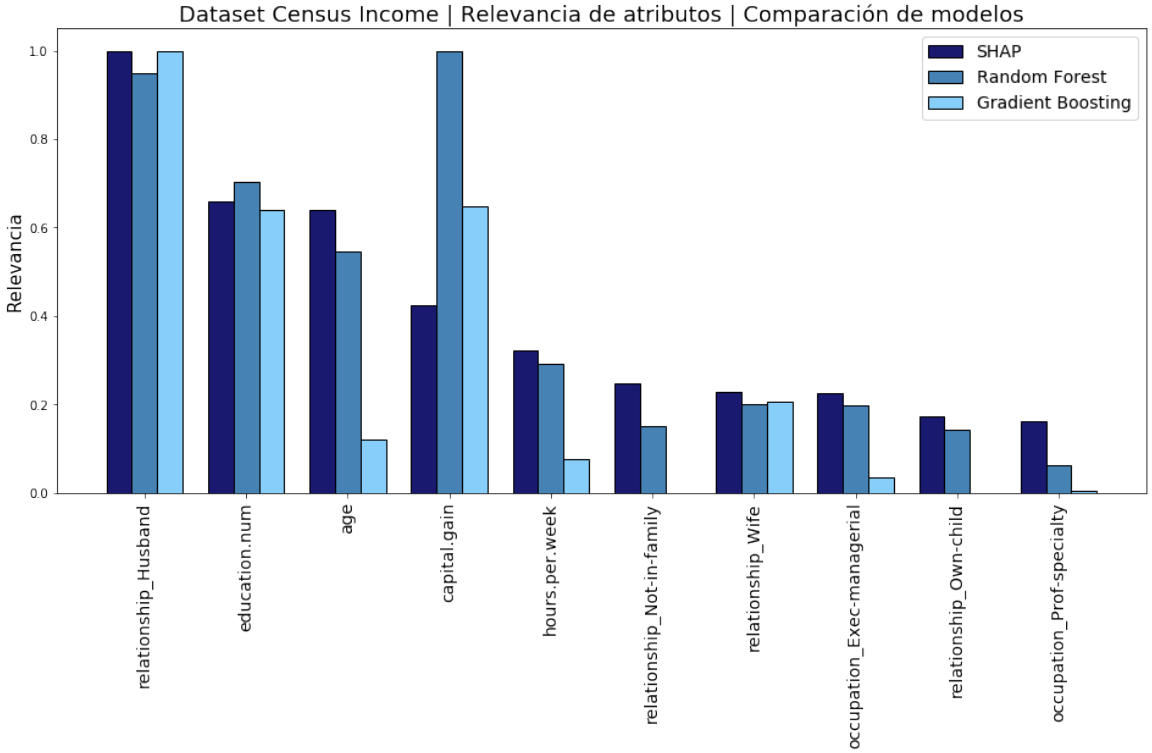

Figura 4.17: Diagramas de barras con los valores de relevancia de los atributos para los tres modelos del dataset Census Income: SHAP values, Random Forest y Gradient Boosting.

La característica más relevante para los tres modelos, a excepción del Random Forest que sitúa un poco por encima la ganancia de capital, es el estado civil de la persona, en concreto si el individuo se identifica como marido (*relationship_Husband*) de otra. También tendremos en sexta posición, y con influencia similar para los tres modelos, el estado civil de casado pero atribuido a la esposa (*relationship_Wife*). En la Fig. 4.19 comprobaremos que tipo de relación existe entre ambos estados civiles y si estos influyen positiva o negativamente en las predicciones de sueldos elevados.

El nivel académico (*education.num*) también es un factor importante en los tres modelos. En este caso esperamos una relación directa entre un mayor grado de educación y una mejor remuneración. La edad (*age*) muestra valores altos de relevancia en los modelos, exceptuando el Gradient Boosting. Aquí supondremos un comportamiento parecido al del nivel académico: a mayor edad, mejor será la remuneración recibida.

Podemos ver diferencias entre nuestro modelo SHAP y los modelos de evaluación para la variable ganancia de capital (*capital.gain*). Las SHAP values la sitúan como un atributo importante pero no le conceden el mismo valor de relevancia que los otros dos modelos, especialmente el Random Forest.

Por último, tenemos campos como las horas de trabajo semanales (*hours.per.week*), estado

civil 'No casado y sin familia' (*relationship_Not-in-family*) y atributos referidos al campo donde la persona ejerce su profesión.

Relación entre los predictores y la variable objetivo. Summary plot.

Una vez han quedado establecidas las variables predictivas más importantes, procederemos a mostrar el tipo de relación que tienen con la variable objetivo mediante el `summary_plot`.

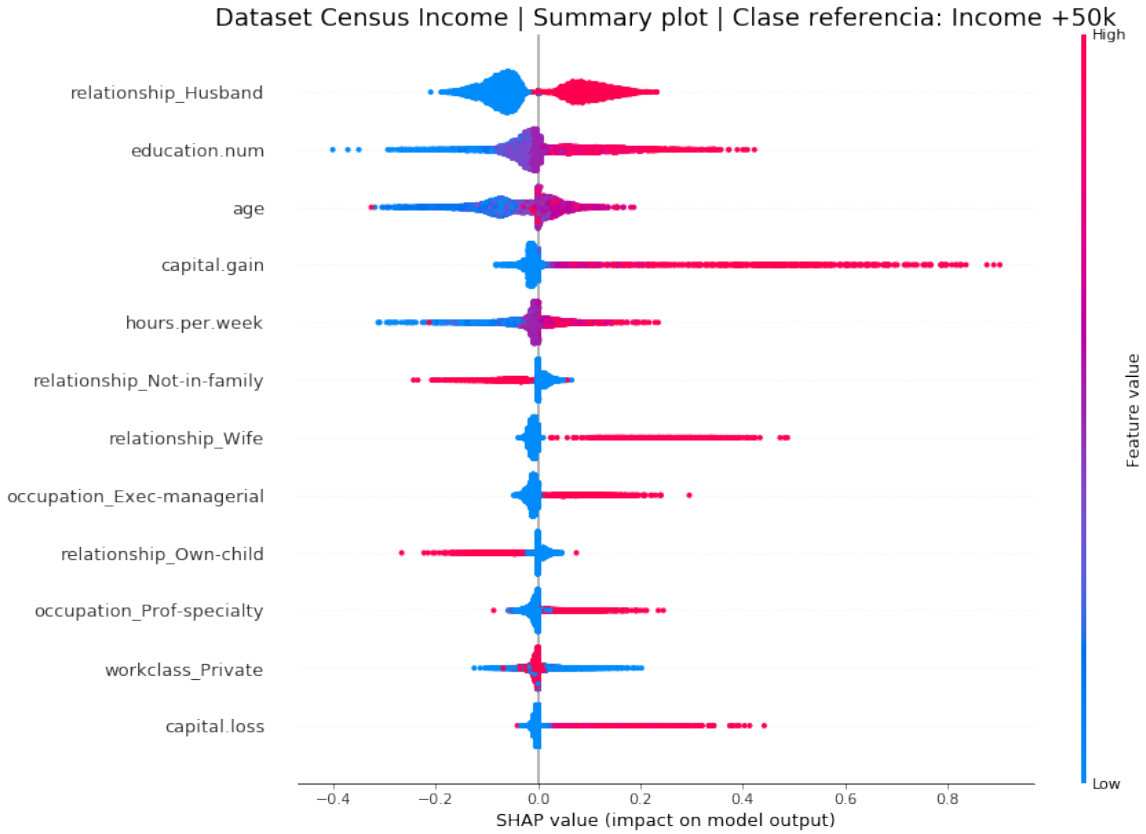

Figura 4.18: Summary plot para la clase Income +50k con los valores de las SHAP values en función del impacto del valor de los atributos en la salida del modelo. Los atributos aparecen ordenados de manera descendente según su relevancia.

Son muchas las conclusiones que podemos extraer de la visualización. La principal de todas es que prácticamente todas las variables tienden a favorecer los sueldos elevados cuando toman valores altos. En el caso de las variables binarias, el valor alto, representado en rojo, significará que la persona se identifica con dicha característica, y el valor bajo, en azul, que no. Por ejemplo, para el caso del estado civil 'Marido', observamos un claro desequilibrio en la distribución de sueldos, siendo los hombres casados los que suelen estar mejor remunerados. En cambio, los hombres no casados tenderán a valores negativos en las SHAP values.

Las únicas excepciones aparecerán para los campos de estado civil 'No casado y sin familia'

e 'Hijo propio', los cuales afectarán negativamente a la predicción de sueldos altos cuando la persona se identifique con los mismos. Las principales razones para este comportamiento podrían ser las inestabilidades que puedan surgir al no pertenecer a una familia y el consumo de tiempo y esfuerzo que supone tener hijos a cargo, respectivamente.

En el caso de las variables numéricas, los valores altos irán siempre acompañados de un incremento en la predicción a favor de sueldos superiores a 50 mil dólares. Por ejemplo, para la ganancia de capital, los valores bajos, generalmente 0, supondrán pequeñas disminuciones en las predicciones positivas. En cambio, para ganancias de capital elevadas, las cuales llegan hasta los 100 mil dólares, el modelo neuronal otorgará máximos en las SHAP values, con valores de hasta +0.9.

Efecto de las variables sobre el resultado previsto. Dependence plot.

A continuación, analizaremos de manera individual mediante `dependence_plot` los atributos que han obtenido resultados más altos en sus SHAP values. Nuestro objetivo será extraer comportamientos en la distribución de los puntos que expliquen la salida del modelo y extrapolarlos al mundo real con conclusiones que amplíen el estudio socio-económico que estamos desarrollando.

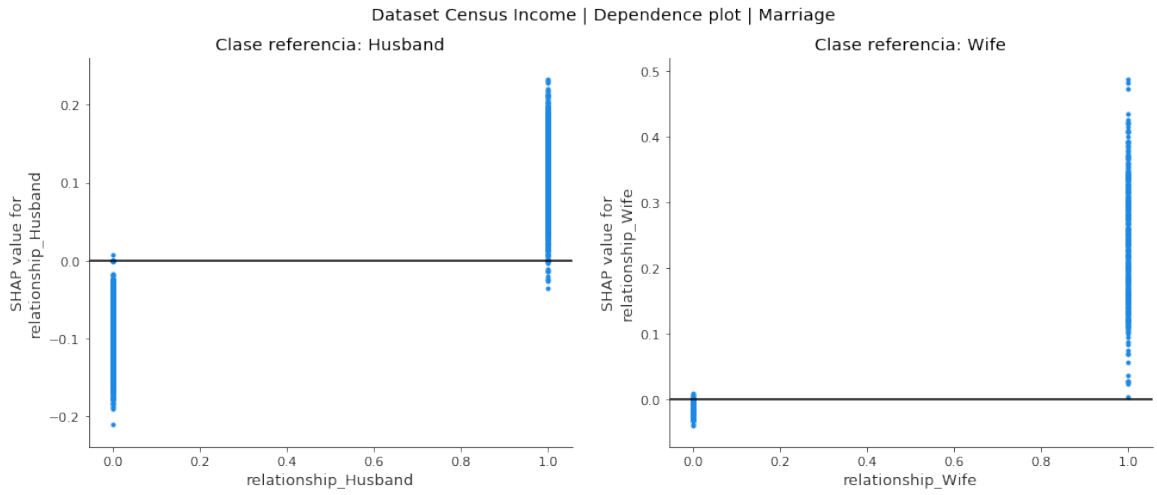

Figura 4.19: Dependence plot con la distribuciones de las SHAP values en función de los valores del atributo 'Marido' (izquierda) y el atributo 'Esposa' (derecha).

En los dos primeros dependence plot hemos comparado las variables correspondientes al estado civil. Por un lado tendremos la característica 'Marido', que presenta claros desequilibrios entre los hombres casados y los que no. Se aprecia una clara tendencia en el modelo a predecir sueldos por debajo de los 50 mil dólares para aquellos hombres no casados, con SHAP values de hasta -0.2. En cambio, para los hombres casados el modelo asignará SHAP values positivos

y las predicciones de sueldos altos se verán beneficiadas.

Por otro lado, tendremos la característica 'Esposa', que presenta un comportamiento similar al atributo 'Marido', pero reduciendo en gran medida el impacto de no estar casada en las mujeres (SHAP values muy cercanos a 0) y extremando el impacto positivo en la predicción cuando sí está casada (SHAP values de hasta +0.5).

Del análisis de la pareja de gráficos de dependencia podemos concluir que, tanto en un sexo como en otro, el estado civil 'Casado' generalmente viene acompañado de sueldos más elevados. Al contrario de lo que ocurre para los individuos que no forman parte de un matrimonio, especialmente en los hombres, que suelen presentar sueldos más bajos. La causa de esto podría ser la estabilidad y seguridad que supone el matrimonio dentro de la sociedad, el cual es fundamental para la integración del individuo en la misma. Otra posible explicación es que los matrimonios suelen formarse una vez la pareja ha alcanzado sus objetivos en el ámbito académico y laboral; primero se alcanza la estabilidad económica y después la familiar y social.

En la siguiente visualización incluiremos una segunda variable interactiva que nos permitirá extraer conclusiones más específicas. En este caso reflejaremos el estado civil 'Marido' con color rojo si la persona se identifica con dicha etiqueta.

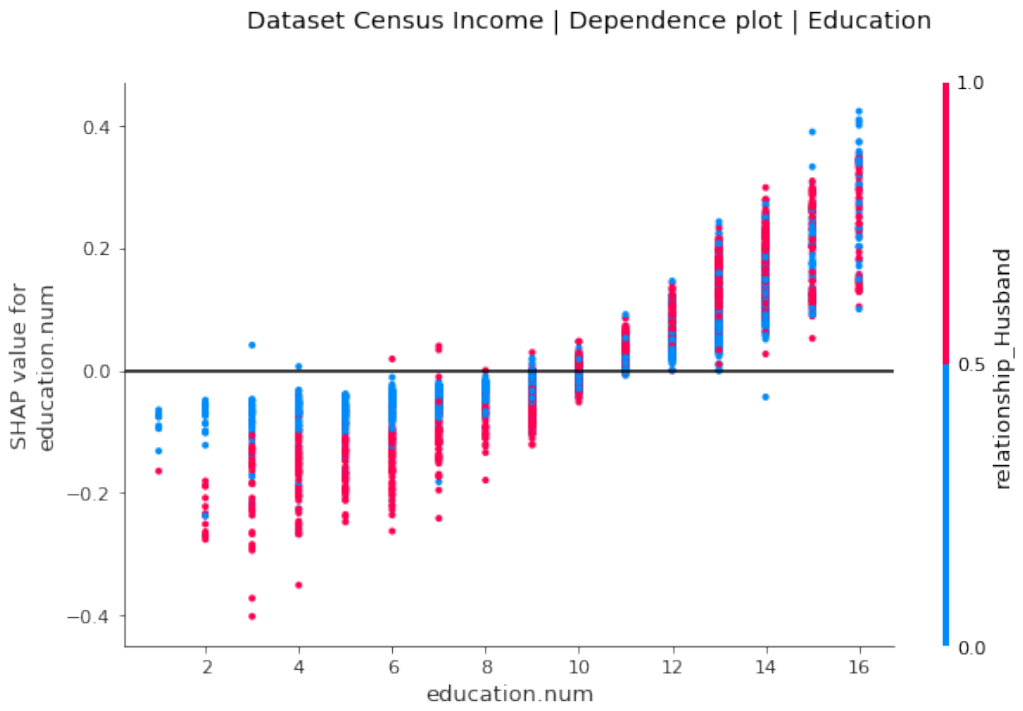

Figura 4.20: Dependence plot con la distribución de las SHAP values en función de los valores del atributo nivel de educación. El color de los puntos representa el estado civil 'Marido' del individuo; rojo si se cumple la característica y azul si no es el caso.

La distribución de los puntos refleja una clara dependencia lineal entre el nivel académico de la persona y las predicciones del modelo. Como era de esperar, cuanto mayor sea el grado de estudios del individuo, más aumentarán sus SHAP values, influenciando positivamente en la predicción. Para las personas con peor educación académica ocurrirá lo contrario mostrando valores negativos para las SHAP values. El punto de inflexión, donde se produce el cambio entre la influencia positiva y negativa, se dará para el nivel 10 'Some-college', correspondiente a personas que iniciaron la universidad pero no la acabaron.

En cuanto a la distribución de colores referidos a la variable 'Husband', la gran parte de valores positivos (en rojo) se hayan en los puntos más alejados del valor 0 para las SHAP values. Es decir, el estado civil de casado para los hombres parece intensificar la asociación de SHAP values en los distintos niveles académicos, aumentando el impacto negativo para los valores educativos bajos y el positivo para los valores educativos altos.

Resumiendo, la preparación y la educación recibida es un factor muy relevante y claramente proporcional al sueldo recibido, suponiendo variaciones de entre -0.4 y +0.4 en las SHAP values. El estado civil volverá a tener un papel importante, terminando de establecer si el nivel académico influye en mayor o menor medida en el individuo. Esto último puede deberse a la estabilidad económica y social que generalmente aporta el matrimonio.

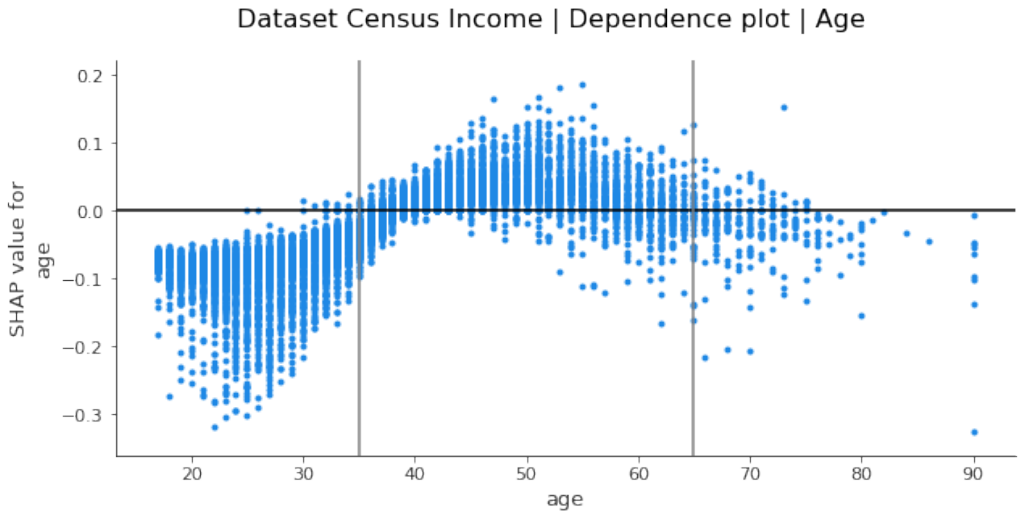

Figura 4.21: Dependence plot con la distribución de las SHAP values en función de los valores del atributo edad. Las líneas verticales indican los 35 años (izquierda) y los 65 años (derecha).

La interpretación de la gráfica para la edad nos lleva a conclusiones aplicables al ámbito laboral y de gran impacto en el panorama socio-económico. Para abordar el análisis, diferenciaremos tres fases que pueden ser aplicadas a la vida laboral de cualquier persona:

- **Fase de formación**, desde los 17 hasta los 35. Durante la primera etapa, los SHAP

values presentan valores negativos, por lo que el modelo interpreta que las personas entre estas edades recibirán sueldos bajos. Los primeros años reflejan SHAP values más altas que a partir de los 22-24 años. Desde ese momento, la influencia de la edad se vuelve cada vez más positiva.

Los primeros años tendremos personas que empiezan a trabajar muy jóvenes, en trabajos mal remunerados y sin preparación. Sin embargo, en un inicio, tenderán a ganar más que los que empiecen a trabajar más tarde con algunos estudios completados (i.e. los trabajos científicos suelen estar mal remunerados al comenzar la carrera profesional). Por lo general, durante estos años el valor de la persona en el ámbito laboral es bajo, debido a la inexperiencia, la falta de estudios o por encontrarse todavía en aprendizaje. Conforme el individuo va incorporando nuevas habilidades laborales y completando sus estudios, su rendimiento crece, aumentando su valor laboral y sueldo.

- **Fase de rendimiento**, desde los 35 hasta los 65. Durante la segunda etapa, los SHAP values pasan a tomar valores positivos, alcanzando el punto máximo de influencia positiva de la variable edad sobre la predicción de salarios altos. Destacamos un pico en los 50 años. A partir de ese momento, las SHAP values permanecen positivas pero disminuye su impacto.

  El individuo alcanza su máximo potencial en este segundo intervalo. La madurez, experiencia y consecución de sus objetivos académicos, además de otras influencias positivas como la estabilidad del matrimonio, permiten al trabajador dar su máximo rendimiento, convirtiéndose en un activo altamente valioso y eficiente en la profesión que desarrolla. Como es lógico, todo esto dará lugar a la correspondiente subida en el salario.

- **Fase de recesión**, a partir de los 65 años. Durante la tercera y última etapa, los SHAP values decrecerán hasta alcanzar valores negativos.

  El rendimiento del individuo comienza a disminuir pese a que su conocimiento sobre el campo y experiencia laboral sean altos. La pérdida de facultades asociadas a edades avanzadas devalúan la labor del trabajador y disminuyen su rendimiento. En algunos casos, las SHAP values permanecerán positivas, posiblemente en profesiones de perfil académico. Los sueldos tenderán a ser menores estos últimos años, ya sea por jubilación o por la recesión del valor del individuo.

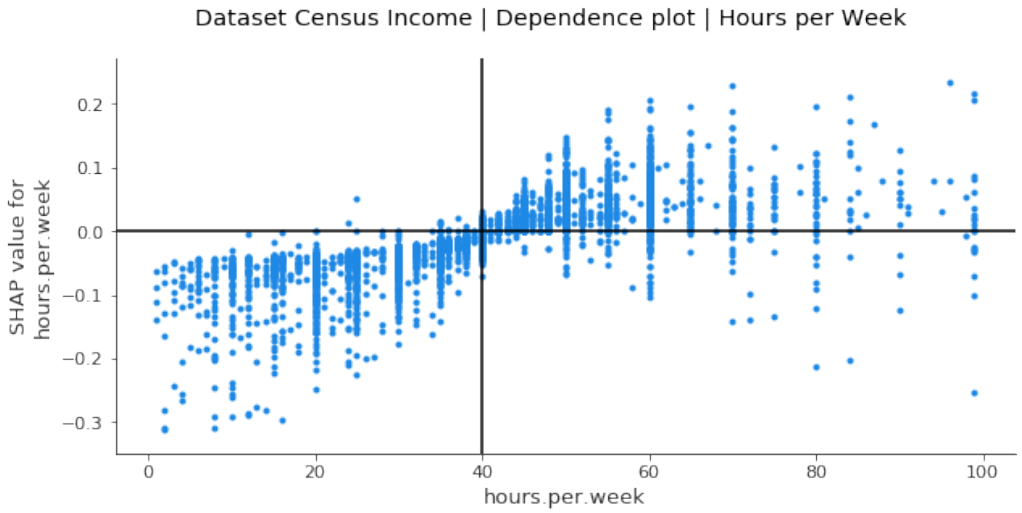

Figura 4.22: Dependence plot con la distribución de las SHAP values en función de los valores del atributo horas laborales semanales. La línea vertical indica la jornada completa, 40 horas semanales.

Como podemos apreciar en la gráfica, el modelo neuronal vuelve a asociar lo que parece una relación lineal directa entre las SHAP values y el número de horas laborales a la semana. Conforme aumenta el tiempo de trabajo de la persona, mayor será el valor positivo que el modelo atribuirá en la clasificación de salarios elevados.

Las SHAP values explican de manera lógica como se distribuyen los salarios en función de las diferentes jornadas laborales. Si tomamos como punto de referencia el trabajo más común, el de jornada completa, correspondiente a 40 horas semanales, este podrá ser interpretado como origen de coordenadas de nuestros ejes. De este modo, dependiendo de si la jornada del trabajador es inferior o superior a las 40 horas, este recibirá un salario más bajo o más alto, respectivamente.

También cabe destacar que para valores superiores a las 40 horas existen cierto número de individuos que reciben SHAP values negativas en sus predicciones. Posiblemente se trate de trabajos con contratos abusivos y mal remunerados, en los que el empleado invierte un gran número de horas sin recibir una compensación justa a cambio.

Conclusiones.

Lo que en un principio era un modelo de caja negra que clasificaba nuestras instancias basándose en unos parámetros desconocidos, ha podido ser interpretado parcialmente gracias al empleo de las SHAP values. Pese a que inicialmente la precisión de la red neuronal era alta, entorno al 84.2 %, no era suficiente para asegurar la fiabilidad del modelo ni satisfacer las necesidades de la tarea encomendada. Mediante el análisis realizado hemos sido capaces de

reforzar la solidez y consistencia con explicaciones SHAP que facilitan el entendimiento de la toma de decisiones del modelo. Todo esto desembocará en una mayor confianza del cliente o en una mayor seguridad en la toma de medidas basadas en los resultados del modelo.

Además, no solo hemos conseguido interpretar parte del modelo, sino que en el proceso también hemos extraído conclusiones e información que pueden ser útiles en el ámbito socioeconómico que tratamos. Hemos podido relacionar la distribución de sueldos con variables como el estado civil, el grado académico, la edad o las horas laborales. Las visualizaciones SHAP permiten gran flexibilidad en la disposición de los puntos, habilitando comportamientos como el de la Fig. 4.21, que difícilmente podrían haber sido observadas de otro modo.

En definitiva, las SHAP values tienen un gran valor en la interpretación de modelos neuronales. Mediante su aplicación sencilla y sin necesidad de realizar un gran análisis previo, hemos alcanzado un alto grado de explicabilidad en el modelo.

4.3. Modelo de clasificación de imágenes: VGG16

4.3.1. Descripción del dataset

El conjunto de datos ImageNet [] recopila más de 15 millones de imágenes de alta resolución pertenecientes a aproximadamente 22.000 categorías. Las imágenes se organizan siguiendo la jerarquía para sustantivos de WordNet[1], en la que cada nodo de la jerarquía está representado por cientos y miles de imágenes. ImageNet consta de imágenes de resolución variable. Por lo tanto, las imágenes se reducen a una resolución fija de 256×256. Dada una imagen rectangular, la imagen se redimensiona y se recorta el parche central de 256×256 de la imagen resultante.

Para el desafío de reconocimiento visual profundo ILSVRC, se empleó un subconjunto de ImageNet con aproximadamente 1000 imágenes de cada una de las 1000 categorías. En total, había aproximadamente 1,2 millones de imágenes de entrenamiento, 50.000 imágenes de validación y 150.000 imágenes de prueba. En nuestro caso no trabajaremos directamente con este dataset, pues emplearemos el modelo preentrenado disponible en la librería `keras` de Python.

Las imágenes usadas para ejemplificar la interpretación de las SHAP values sobre las predicciones del modelo serán extraídas de distintas fuentes, las cuales serán propiamente referenciadas en la descripción de la imagen. Trabajaremos con imágenes de diversa índole, con características que nos permitan exprimir al máximo las facetas de las SHAP values para ver hasta dónde son capaces de llegar en sus explicaciones. Las imágenes pueden ser descargadas desde la carpeta 'Images' de la dirección Github.

[1] Es una gran base de datos léxica en inglés que agrupa los tipos de palabras en conjuntos de sinónimos cognitivos, cada uno de los cuales expresa un concepto distinto. Su estructura lo convierte en una herramienta útil para la lingüística computacional y el procesamiento del lenguaje natural.

4.3.2. Código

En este apartado describiremos la implementación realizada del modelo VGG16 para clasificación de imágenes y las respectivas explicaciones SHAP, la cual se puede encontrar en el archivo *python/ImageModel_VGG16*. El programa se divide en tres partes de código diferenciadas.

Carga del modelo y segmentación de la imagen.

La biblioteca `keras` nos permite hacer uso de modelos de aprendizaje profundo junto con pesos previamente entrenados. Los pesos se descargan automáticamente al crear una instancia de un modelo. En nuestro caso, cargaremos el modelo VGG16 con los parámetros ajustados para la clasificación de imágenes, los cuales han sido calculados mediante el preentrenamiento realizado en el dataset ImageNet. También crearemos una lista con los nombres de las etiquetas del dataset, es decir los posibles objetos con los que puede ser identificadas las imágenes.

Cargamos la imagen que vamos a clasificar con nuestro modelo. La imagen deberá ser ajustada para su correcta entrada en el modelo VGG16, reescalándola a una imagen cuadrada de tamaño 224 x 224 píxeles. También convertimos la imagen en una matriz Numpy, sobre la que aplicaremos el modelo.

A continuación, segmentamos la imagen usando la función `slic`. Para cada imagen estableceremos los parámetros `n_segments`, `compactness` y `sigma` que mejor se adapten a las formas de la imagen y muestren una mayor información de la misma. En la siguiente Figura podemos ver un claro ejemplo de como quedaría segmentada una de las imágenes.

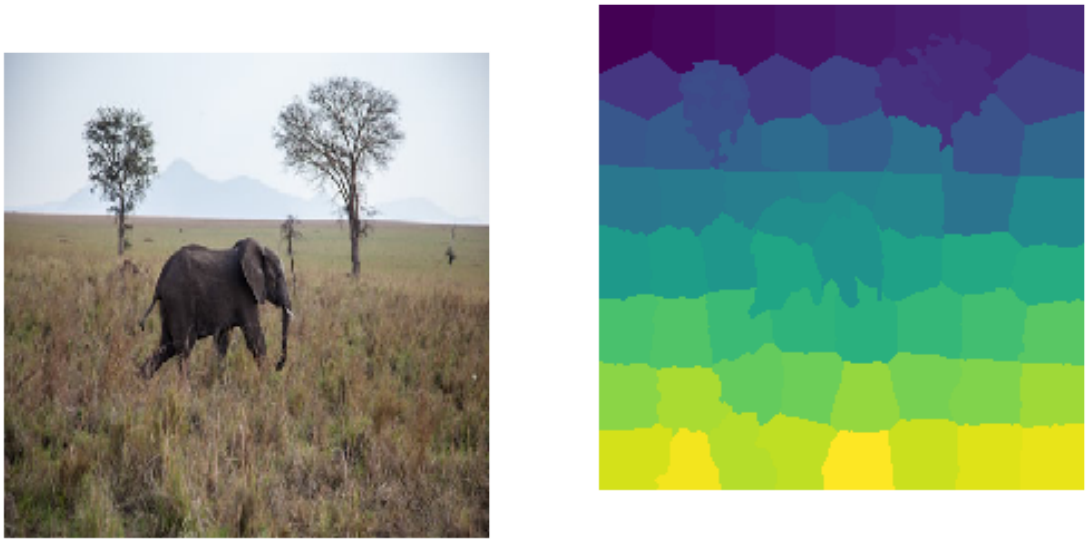

Figura 4.23: Imagen original a clasificar (izquierda) e imagen tras la segmentación (derecha).

Como podemos observar, nuestra imagen consistirá en un elefante africano que se desplaza

por la sabana. La segmentación nos permite seguir viendo las formas más relevantes de la imagen, distinguiéndose claramente las secciones que representan al elefante, los árboles del fondo y el horizonte.

La segmentación facilitará y agilizará la clasificación de las imágenes causando que el modelo no tenga que explicar todos los píxeles y pueda centrarse en las zonas más relevantes de la imagen. Cada una de estas zonas actuará como un 'atributo' al que se le asignará un peso en la clasificación.

Predicción de la clasificación.

La predicción irá precedida por una función con la que crearemos una máscara binaria sobre la imagen, indicando si una determinada región de la imagen está oculta. Aplicando la función al array que contiene la imagen original y la imagen segmentada, podremos calcular la predicción de clasificación del modelo VGG16. Para ello, creamos otra función `f` que será la que explicaremos mediante las SHAP values:

```
def f(z):
```

```
 return model.predict(preprocess_input(mask_image(z, segments_slic, img_orig,
255)))
```

A continuación, extraeremos las predicciones más valoradas del modelo, es decir, aquellas etiquetas que el modelo considera que clasifican mejor la imagen de entrada. Para visualizar los resultados de la clasificación crearemos un diagrama de barras con las ocho etiquetas que el modelo ha considerado más significativas y sus respectivos valores de predicción. Para la imagen de la Fig. 4.23 obtenemos el siguiente diagrama:

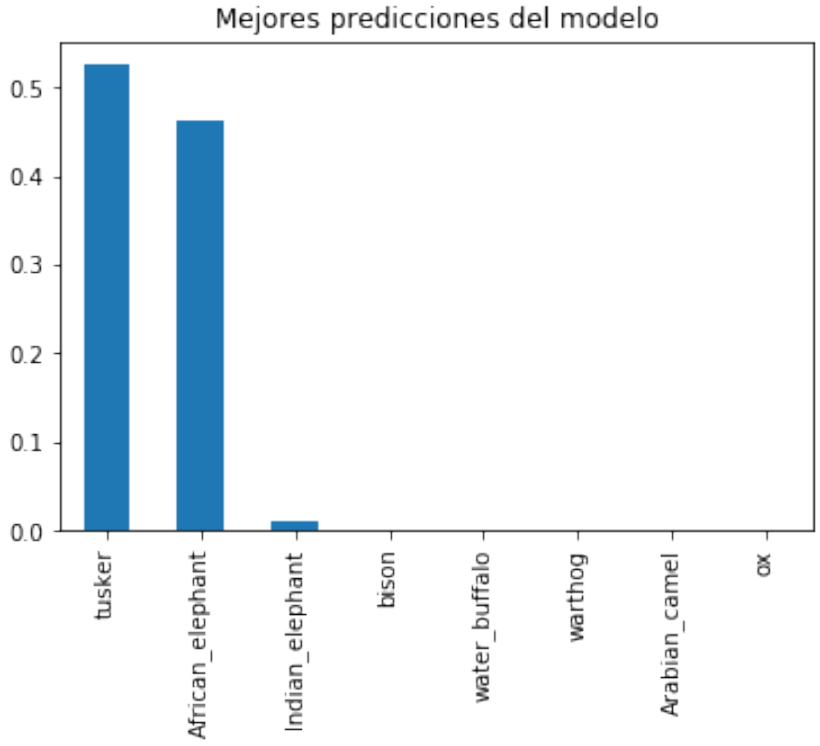

Figura 4.24: Diagrama de barras con las predicciones mejor valoradas del modelo VGG16.

El diagrama nos muestra que la salida del modelo de clasificación se encuentra entre dos predicciones: 'tusker', que podría ser descrito como animal con colimllos y 'african_elephant', elefante africano. También considera la posibilidad, aunque en menor medida, de que se trate de un 'indian_elephant', elefante indio.

En definitiva, el modelo ha descrito correctamente la pieza central de la imagen: nos encontramos ante un elefante africano.

Aplicación SHAP values y visualización de los resultados.

La aplicación de las SHAP values se realizará sobre la función `f` definida anteriormente. El conjunto de datos de fondo (*background*) que se utilizará para explicar la función será una matriz de ceros. Dado que la mayoría de los modelos no están diseñados para manejar datos faltantes arbitrarios en el momento de la prueba, simularemos atributos "faltantes", pues así es como funcionan los valores de Shapley, reemplazando la función con los valores que toma en el conjunto de datos de fondo. En este caso, como el conjunto de datos de fondo es una muestra de todo ceros, simulamos la ausencia de una característica configurándola a cero. Para obtener las SHAP values a partir del `explainer` haremos lo mismo, estableciendo una matriz del mismo tamaño pero de unos en lugar de ceros. Repetiremos el cálculo un número `nsamples` de veces.

```
explainer = shap.KernelExplainer(f, np.zeros((1,50)))
```

```
shap_values = explainer.shap_values(np.ones((1,50)), nsamples=1000)
```

El tiempo de ejecución de este código será de alrededor de 2 minutos para cada imagen.

La visualización de la interpretación SHAP de los resultados del modelo consistirá en la representación de la imagen original, más las tres explicaciones de las etiquetas que han conseguido un mayor peso en la clasificación.

Primero crearemos un mapa de color que nos permitirá escalar la importancia de las diferentes regiones de la imagen. Según el color y la intensidad que presente una sección de la imagen tendrá ligado un mayor o menor SHAP value. El color verde representará SHAP values positivos que aumentan la probabilidad de la clase, mientras que el color rojo representará SHAP values negativos que reducen la probabilidad de la clase.

A continuación, definiremos una función `fill_segmentation` que asignará a las distintas partes de la imagen segmentada su respectivo SHAP value. Todos los píxeles de cada región individual delimitada en la segmentación tendrán un mismo valor, lo que determinará el color de dicha sección.

Por último, creamos la visualización representando las tres predicciones mejor valoradas. Para ello establecemos el valor máximo que toman las SHAP values como referencia, estableciendo la escala de color en función del mismo. Cada una de las etiquetas seleccionadas mostrará la imagen segmentada y las zonas coloreadas según su SHAP value correspondiente. También mostraremos la escala para poder relacionar los colores con su respectivos valores. Siguiendo con el ejemplo de la Fig. 4.23, nuestra visualización de resultados quedaría de la siguiente forma:

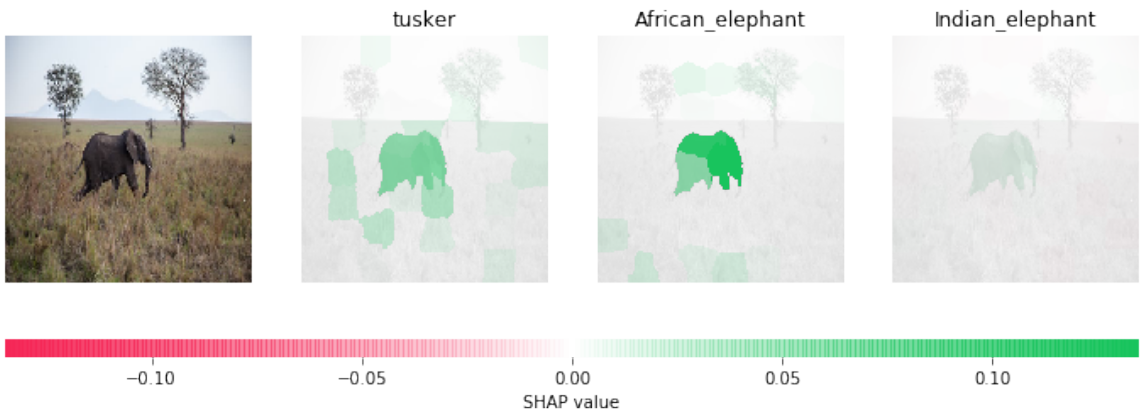

Figura 4.25: Visualización de las tres predicciones más valoradas del modelo VGG16; imagen original (izquierda), 'tusker' (centro-izquierda), 'African_elephant' (centro-derecha) e 'Indian_elephant' (derecha), con las SHAP values asignadas a las distintas regiones. La escala de colores en función de las SHAP values aparece representada debajo.

Para la etiqueta con un mayor peso en la predicción del modelo, 'tusker', tendremos clara-

mente señaladas en color verde las regiones correspondientes al elefante. También apreciamos regiones con influencia positiva en la predicción alrededor del elefante, las cuales representan el paisaje en el que se encuentra.

Para la siguiente etiqueta mejor valorada, 'African_elephant', observamos un comportamiento similar. En este caso las SHAP values alcanzarán valores máximos en la figura del elefante, el cual es identificado por la red neuronal en la subespecie africana. El impacto del paisaje es inferior para esta etiqueta, el modelo no interpreta que se trata de la sabana africana y no termina de situar esta etiqueta como principal predicción.

La última etiqueta, 'Indian_elephant', recibe ciertos SHAP values positivos en la figura del elefante, pero el modelo termina por determinar que no se trata de la subespecie india.

Capítulo 5

Futuras mejoras y conclusiones

Bibliografía

[1] *A VALUE FOR N-PERSON GAMES*. Defense Technical Information Center, 1952.

[2] S. Sen A. Datta e Y. Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems." En: *2016 IEEE Symposium on Security and Privacy (SP)* (mayo de 2016), págs. 598-617.

[3] J. Bleich A. Goldstein A. Kapelner y E. Pitkin. *Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation.* 2013.

[4] L. Asker A. Henelius K. Puolamaki y P. Papapetrou. "A peek into the black box: exploring classifiers by randomization." En: *Data Mining and Knowledge Discovery* (2014), 28(5-6):1503-1529.

[5] J. Adebayo y L. Kagal. "Iterative orthogonal feature projection for diagnosing bias in black-box models." En: (2016).

[6] Osbert Bastani, Carolyn Kim y Hamsa Bastani. *Interpretability via Model Extraction.* 2018. arXiv: 1706.09773 [cs.LG].

[7] Vaishak Belle y Ioannis Papantonis. *Principles and Practice of Explainable Machine Learning.* 2020. arXiv: 2009.11698 [cs.LG].

[8] Dillon Bowen y Lyle Ungar. *Generalized SHAP: Generating multiple types of explanations in machine learning.* 2020. arXiv: 2006.07155 [cs.LG].

[9] P. Cortez y M. J. Embrechts. "Opening black box data mining models using sensitivity analysis." En: *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)* (2011), págs. 341-348.

[10] P. Cortez y M. J. Embrechts. "Using sensitivity analysis and visualization techniques to open black box data mining models." En: *Information Sciences* (2013), 225:1-17.

[11] R. A. FISHER. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS". En: *Annals of Eugenics* 7.2 (1936), págs. 179-188. DOI: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

[12] J. H. Friedman. *Greedy function approximation: A gradient boosting machine.* 2001.

[13] R. Kush G. Su D. Wei y D. Malioutov. *Interpretable two-level boolean rule learning for classification.* 2016.

[14] Julià Minguillón Alfonso Jordi Gironés Roig Jordi Casas Roma y Ramon Caihuelas Quiles. *Minería de datos: modelos y algoritmos.* Editorial UOC, 2017.

[15] S. Krishnan y E. Wu. Palm. *Machine learning explanations for iterative debugging.* New York, NY, USA, 2017.

[16] S. M. Lundberg y S.-I. Lee. "A unified approach to interpreting model predictions." En: *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17.* (2015), págs. 4768-4777.

[17] C. Guestrin M. T. Ribeiro S. Singh. *"why should i trust you?": Explaining the predictions of any classifier.* New York, NY, USA, 2016.

[18] C. Guestrin M. T. Ribeiro S. Singh. *Anchors: High-precision model-agnostic explanations.* 2018.

[19] Christoph Molnar. *A Guide for Making Black Box Models Explainable.* URL: https://christophm.github.io/interpretable-ml-book/index.html.

[20] F. Pedregosa y col. "Scikit-learn: Machine Learning in Python". En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[21] G. Hooker S. Tan R. Caruana e Y. Lou. *Distill-and-compare: Auditing black-box models using transparent model distillation.* 2017.

[22] B. Mittelstadt S. Wachter y C. Russell. "Counterfactual explanations without opening the black box: Automated decisions and the gdpr". En: *Harvard journal of law and technology* (2018), 31:841-887.

[23] Karen Simonyan y Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* 2015. arXiv: 1409.1556 [cs.CV].

[24] E. Strumbelj e I. Kononenko. "An efficient explanation of individual classifications using game theory". En: (2010), 11:1-18.