



Escuela Técnica Superior de
Ingeniería Informática

TRABAJO DE FIN DE MÁSTER

Aprendizaje supervisado explicable

Realizado por

Ana Conde Marrón

Para la obtención del

Máster Universitario en Lógica, Computación e Inteligencia Artificial

Dirigido por

Álvaro Romero Jiménez

En el departamento de

Ciencias de la Computación e Inteligencia Artificial

Sevilla, junio de 2023

Me gustaría dedicar este trabajo de fin de máster a todas las personas que, en mi día a día, me recuerdan que no todo es blanco o negro, pues hay grises en los que es precioso vivir.

Abstract

Este trabajo de fin de máster busca exponer algunas de las distintas técnicas que hoy día están disponibles para conferir explicabilidad. Se consideraron dos enfoques: explicabilidad local, en la que la explicación se genera a partir de predicciones, y explicabilidad global en la que la explicación se extrae del modelo de forma global. Para cada uno de los enfoques se utilizó una técnica especializada. Además se organizó este estudio según la tarea a desarrollar: clasificación binaria, clasificación multiclase o regresión y dependiendo del modelo de caja negra utilizado: modelo de bosque aleatorio o redes neuronales.

Todos los resultados se han apoyado en los gráficos que estos métodos proporcionan para explicitar la explicabilidad y se han adaptado los procedimientos acordes al tipo de tarea llevada a cabo.

Lo primero que se abordará será un pequeño estado del arte en el que se ponga en situación sobre el aprendizaje automático actual y la necesidad de la explicabilidad en momento que estamos viviendo en el que la inteligencia artificial está llegando a tantos campos en los que se necesitan respuestas consistentes. Se seguirá con los enfoques posibles para alcanzar la explicabilidad y la explicación, ventajas e inconvenientes de las técnicas de explicabilidad que formarán parte del estudio.

Las siguientes secciones expondrán los resultados de utilizar estas técnicas para tres tareas distintas: clasificación binaria, multiclase y regresión, para dos modelos distintos: modelos de bosques aleatorios y redes neuronales. En esta sección se discutirán los resultados indicando para qué casos las explicaciones son factibles o para qué casos los modelos de explicabilidad no han sido capaces de detectar los patrones.

Por último, se terminará con unas conclusiones en las que destacará para las distintas tareas qué técnicas dan mejores resultados.

Índice general

Índice de figuras	ix
1. Introducción	1
1.1. Aprendizaje automático	1
1.2. Modelos de caja blanca y de caja negra	2
1.3. Necesidad de interpretabilidad de los modelos	4
2. Explicabilidad	7
2.1. Dimensiones de la explicabilidad	7
2.1.1. Generación de explicaciones	7
2.1.2. Construcción de explicadores	8
2.2. Técnicas de explicabilidad	10
2.2.1. Explicabilidad global	10
2.2.2. Explicabilidad local	14
3. Ejemplos de Uso	21
3.1. Modelos de caja negra utilizados	22
3.1.1. Bosque aleatorio	22
3.1.2. redes neuronales	22
3.2. Tarea: clasificación binaria	23
3.2.1. Conjuntos de datos utilizado	23
3.2.2. Modelo de caja negra: modelo de bosque aleatorio	24
3.2.3. Modelo de caja negra: redes Neuronales	30
3.3. Tarea: clasificación multiclase	35
3.3.1. Conjunto de datos utilizado	35
3.3.2. Modelo de caja negra: modelo de bosque aleatorio	36
3.3.3. Modelo de explicabilidad: permutación de atributos	46
3.3.4. Modelo de explicabilidad: importancia de atributos	47

3.3.5. Modelo de caja negra: redes neuronales	48
3.3.6. Modelo de explicabilidad: permutación de atributos	57
3.4. Tarea: regresión	58
3.4.1. Conjunto de datos utilizado	58
3.4.2. Modelo de caja negra: modelo de bosque aleatorio	59
3.4.3. Modelo de explicabilidad: permutación de atributos	70
3.4.4. Modelo de explicabilidad: importancia de atributo	71
3.4.5. Modelo de caja negra: redes neuronales	71
3.4.6. Modelo de explicabilidad: permutación de atributos	81
4. Conclusiones	83
Bibliografía	87

Índice de figuras

1.1. Ejemplo de árbol de decisión	3
1.2. Ejemplo de red neuronal	4
1.3. Caso práctico árbol de decisión	4
2.1. Esquema ilustrativo explicabilidad global y local	8
2.2. Esquema explicativo modelo subrogado global y local	9
2.3. Ejemplo explicación salida importancia de atributos mediante permutaciones	11
2.4. Ejemplo explicación salida LIME	16
2.5. Explicación parámetros de LIME	16
2.6. Ilustración explicativa valores de <i>Shapley</i>	17
2.7. Ejemplo de las coaliciones	18
2.8. Explicación gráfica <i>TreeSHAP</i>	19
2.9. Ejemplo explicación salida SHAP	19
3.1. Esquema desglose casos de uso	21
3.2. Ilustración explicación modelo bosque aleatorio	22
3.3. Ilustración explicativa red neuronal	23
3.4. Extracto datos preprocesados para Clasificación Binaria	24
3.5. Extracto datos post-procesados para Clasificación Binaria	24
3.6. Clasificación binaria: árbol de decisión para modelo de bosque aleatorio	25
3.7. Clasificación binaria: sistema basado en reglas para modelo de bosque aleatorio	26
3.8. Clasificación binaria: LIME para modelo de bosque aleatorio 1	27
3.9. Clasificación binaria: LIME para modelo de bosque aleatorio 2	27
3.10. Clasificación binaria: SHAP para modelo de bosque aleatorio	28
3.11. Clasificación binaria: permutación de atributos para modelo de bosque aleatorio	29

3.12. Clasificación binaria: importancia de atributos para modelo de bosque aleatorio	30
3.13. Clasificación binaria: árbol de decisión para modelo de red neuronal	31
3.14. [Clasificación binaria: sistema basado en reglas para modelo de red neuronal	32
3.15. Clasificación binaria: LIME para modelo de red neuronal 1	32
3.16. Clasificación binaria: LIME para modelo de red neuronal 2	33
3.17. Clasificación binaria: LIME para modelo de red neuronal 3	33
3.18. Clasificación binaria: SHAP para modelo de red neuronal 1	34
3.19. Clasificación binaria: SHAP para modelo de red neuronal 2	34
3.20. Clasificación binaria: permutación de atributos para modelo de bosque aleatorio	35
3.21. Extracto datos para Clasificación Multiclase	36
3.22. Clasificación multiclase: árbol de decisión para modelo de bosque aleatorio	37
3.23. Clasificación multiclase: sistema basado en reglas para modelo de bosque aleatorio 1	38
3.24. Clasificación multiclase: sistema basado en reglas para modelo de bosque aleatorio 2	39
3.25. Clasificación multiclase: sistema basado en reglas para modelo de bosque aleatorio 3	39
3.26. Clasificación multiclase: sistema basado en reglas para modelo de bosque aleatorio 4	40
3.27. Clasificación multiclase: LIME para modelo de bosque aleatorio 1	41
3.28. Clasificación multiclase: LIME para modelo de bosque aleatorio 2	41
3.29. Clasificación multiclase: LIME para modelo de bosque aleatorio 3	42
3.30. Clasificación multiclase: SHAP para modelo de bosque aleatorio 1	44
3.31. Clasificación multiclase: SHAP para modelo de bosque aleatorio 2	45
3.32. Clasificación multiclase: SHAP para modelo de bosque aleatorio 3	46
3.33. Clasificación multiclase: permutación de atributos para modelo de bosque aleatorio	47
3.34. Clasificación multiclase: importancia de atributos para modelo de bosque aleatorio 1	48
3.35. Clasificación multiclase: árbol de decisión para modelo de red neuronal	49
3.36. Clasificación multiclase: sistema basado en reglas para modelo de red neuronal 1	50

3.37. Clasificación multiclasa: sistema basado en reglas para modelo de red neuronal 2	50
3.38. Clasificación multiclasa: sistema basado en reglas para modelo de red neuronal 3	51
3.39. Clasificación multiclasa: sistema basado en reglas para modelo de red neuronal 4	51
3.40. Clasificación multiclasa: LIME para modelo de red neuronal 1	52
3.41. Clasificación multiclasa: LIME para modelo de red neuronal 2	53
3.42. Clasificación multiclasa: LIME para modelo de red neuronal 3	54
3.43. Clasificación multiclasa: SHAP para modelo de red neuronal 1	55
3.44. Clasificación multiclasa: SHAP para modelo de red neuronal 2	56
3.45. Clasificación multiclasa: SHAP para modelo de red neuronal 3	57
3.46. Clasificación multiclasa: permutación de atributos para modelo de red neuronal 1	58
3.47. Extracto datos preprocesados para Regresión	59
3.48. Regresión: árbol de decisión para modelo de bosque aleatorio	60
3.49. Regresión: histograma de la distribución de datos de predicción bosque aleatorio	61
3.50. Regresión: sistema basado en reglas para modelo de bosque aleatorio 1	62
3.51. Regresión: histograma sistema basado en reglas para modelo de bosque aleatorio 1	63
3.52. Regresión: sistema basado en reglas para modelo de bosque aleatorio 2	63
3.53. Regresión: histograma sistema basado en reglas para modelo de bosque aleatorio 2	64
3.54. Regresión: sistema basado en reglas para modelo de bosque aleatorio 3	64
3.55. Regresión: histograma sistema basado en reglas para modelo de bosque aleatorio 3	65
3.56. Regresión: sistema basado en reglas para modelo de bosque aleatorio 4	65
3.57. Regresión: histograma Sistema basado en reglas para modelo de bosque aleatorio 4	66
3.58. Regresión: LIME para modelo de bosque aleatorio 1	67
3.59. Regresión: LIME para modelo de bosque aleatorio 2	68
3.60. Regresión: LIME para modelo de bosque aleatorio 2	68
3.61. Regresión: pesos LIME para modelo de bosque aleatorio 3	69
3.62. Regresión: SHAP para modelo bosque aleatorio 1	69
3.63. Regresión: SHAP para modelo bosque aleatorio 2	70

3.64. Regresión: SHAP para modelo bosque aleatorio 3	70
3.65. Regresión: permutación de atributos para modelo bosque aleatorio 1 . .	71
3.66. Regresión: importancia de atributos para modelo bosque aleatorio 1 . .	71
3.67. Regresión: árbol de decisión para modelo de red neuronal	73
3.68. Regresión: histograma de la distribución de datos de predicción red neuronal	74
3.69. Regresión: sistema basado en reglas para modelo de red neuronal 1 . .	75
3.70. Regresión: sistema basado en reglas para modelo de red neuronal 2 . .	75
3.71. Regresión: histograma sistema basado en reglas para modelo de red neuronal 2	76
3.72. Regresión: sistema basado en reglas para modelo de red neuronal 3 . .	76
3.73. Regresión: histograma sistema basado en reglas para modelo de red neuronal 3	77
3.74. Regresión: sistema basado en reglas para modelo de red neuronal 4 . .	77
3.75. Regresión: sistema basado en reglas para modelo de red neuronal 5 . .	77
3.76. Regresión: LIME para modelo de red neuronal 1	78
3.77. Regresión: LIME para modelo de red neuronal 2	79
3.78. Regresión: pesos LIME para modelo de red neuronal 2	79
3.79. Regresión: LIME para modelo de red neuronal 3	80
3.80. Regresión: SHAP para modelo red neuronal 1	80
3.81. Regresión: SHAP para modelo red neuronal 2	80
3.82. Regresión: SHAP para modelo red neuronal 3	81
3.83. Regresión: permutación de atributos para modelo red neuronal	82

Capítulo 1

Introducción

El objetivo de este trabajo de fin de máster es el estudio de distintas modelos y técnicas que nos permiten conferir explicabilidad a modelos conocidos como caja negra. Para un mejor entendimiento de la problemática se explicarán algunos conceptos clave, que son: qué es el aprendizaje automático y qué engloba, qué son los modelos de caja negra y los modelos de caja blanca y la necesidad de la explicabilidad en los modelos.

1.1. Aprendizaje automático

El aprendizaje automático es un campo de estudio de la inteligencia artificial que trata sobre el desarrollo y estudio de algoritmos estadísticos que pueden aprender de los datos y generalizar para datos nuevos y por ende desarrollar tareas sin instrucciones explícitas.

Enfoques de aprendizaje automático se han aplicado a múltiples campos como el procesamiento del lenguaje natural, la visión por ordenador, el filtrado de correos, la medicina, y la economía, entre otros. [1]

Un modelo de aprendizaje supervisado se puede entender formalmente como una función $h(x) = y$ que relaciona un vector de características $x \in X \subseteq \mathbb{R}^d$ con un objetivo $y \in Y \subseteq \mathbb{R}$. El proceso de aprendizaje del modelo se realizará a partir de un conjunto de entrenamiento que relaciona distintos vectores de características con el valor del objetivo pretendido para ellos. La construcción de un modelo de aprendizaje supervisado se puede formular, por tanto, como un problema de optimización.

$$h^* = \arg \min_{h \in H} \frac{1}{n} \sum_{i=1}^n (S(h(x_i), y_i))$$

Donde h^* es el modelo resultante, h es una hipótesis del espacio de hipótesis H , n es el número total de instancias pertenecientes al conjunto de entrenamiento y S es la función de pérdida o medida de error.

Cada modelo de aprendizaje automático sigue un algoritmo para el entrenamiento del mismo. Por ejemplo, el algoritmo por detrás de un árbol de decisión puede ser el llamado CART. A través de este algoritmo se entrena el modelo consiguiendo una configuración interna que permitirá al modelo hacer predicciones. Dependiendo del tipo de modelo la obtención de esa configuración interna se puede traducir de distintas formas, por ejemplo, en el caso de los árboles de decisión tras el entrenamiento obtenemos el árbol en el que los nodos contienen los atributos y los límites de los mismo por los cuales clasificar y las hojas finales la clasificación del objeto pasado al árbol. O en el caso de las redes neuronales, tras el entrenamiento se obtiene la configuración de los pesos de la red que permitirá clasificar las instancias.

1.2. Modelos de caja blanca y de caja negra

Se pueden identificar dos tipos de modelos según su explicabilidad: los modelos de caja blanca y los modelos de caja negra. Los modelos de caja blanca o modelos interpretables son modelos que tras su entrenamiento son fácilmente interpretables el porqué de sus decisiones. Es decir, podemos comprender por qué a partir de las entradas dadas la salida del sistema es la obtenida.

Un ejemplo paradigmático de modelo de caja blanca es el de los árboles de decisión. Una vez entrenado, un árbol de decisión divide los datos varias veces en función del resultado medio de los datos de entrenamiento en ese nodo. Por lo tanto, la predicción del valor objetivo para una instancia concreta es una simple consecuencia de que las características que describen a la instancia cumplen las condiciones asociadas a una serie de nodos internos (véase, por ejemplo, la figura 1.1).

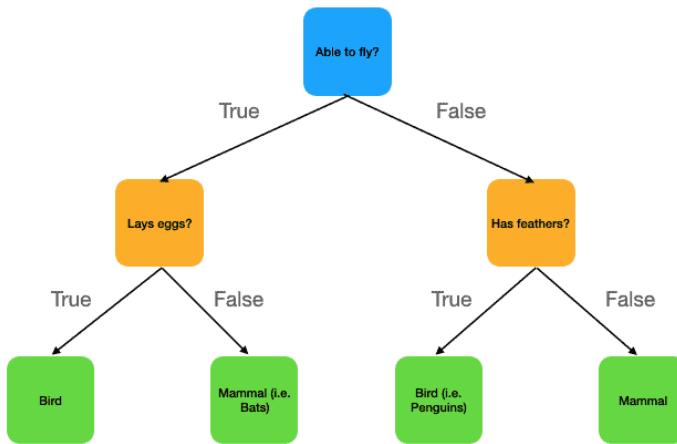


Figura 1.1 Se puede observar cómo este modelo realiza cuatro predicciones fácilmente interpretables: un animal capaz de volar y que pone huevos es un pájaro; un animal capaz de volar y que no pone huevos es un mamífero; un animal incapaz de volar y que tiene plumas es un pájaro; un animal incapaz de volar y que no tiene plumas es un mamífero.

Los modelos de caja negra son aquellos modelos de aprendizaje supervisado que no son intrínsecamente interpretable, de tal forma que no hay una relación evidente entre la salida obtenida para una entrada y los atributos que describen a esta última. Ejemplos paradigmáticos de modelos de caja negra serían las redes neuronales y los bosques aleatorios (*random forests*, en inglés). Las primeras establecen una relación no lineal entre las entradas y las salidas, lo que, aun conociendo los valores de los pesos de todas las conexiones, dificulta entender el desempeño del modelo (véase, por ejemplo, la figura 1.2). Los segundos son ensambles de árboles de decisión en los que, al obtenerse la salida como la moda o el promedio de las salidas proporcionadas por estos últimos, es complicado relacionarla directamente con los valores de los atributos que describen las entradas (véase, por ejemplo, la figura 1.3).

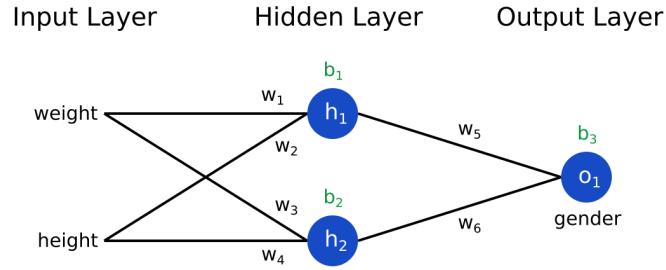


Figura 1.2 En esta figura podemos ver un ejemplo muy simple de red neuronal, relaciona la altura y el peso para obtener el género de una persona. Pese a que no es un ejemplo complejo, a primera vista no se puede interpretar la configuración de la red como para comprender qué valores de las entradas determinan que será de un género u otro.

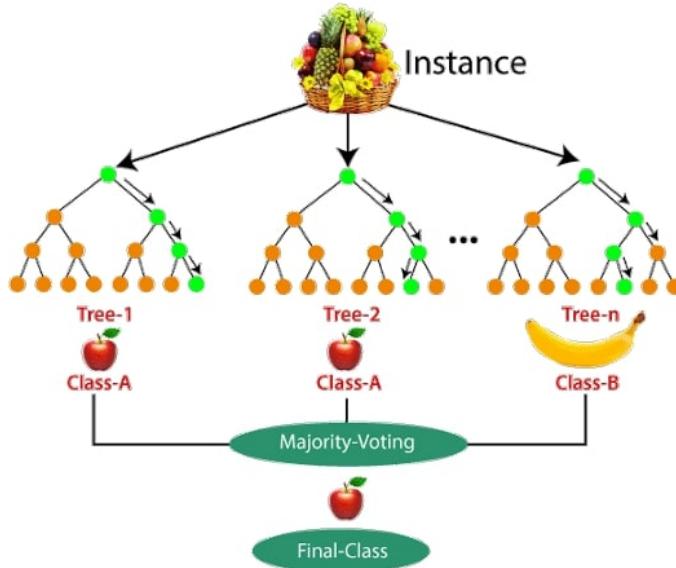


Figura 1.3 El primer árbol predirá la clase manzana al igual que el segundo y el tercero la clase plátano. No obstante, en la escala que suelen tener estos modelos hablaríamos de decenas de árboles los cuales tendrían unas condiciones para cada clase y sumando al sistema de votación, se difumina mucho la explicabilidad.

1.3. Necesidad de interpretabilidad de los modelos

En los últimos años los modelos de aprendizaje automático han conseguido resultados sorprendentes tanto en precisión como en rapidez, pero ahora el ser humano se enfrenta a un reto: la fiabilidad.

La precisión de un modelo no es lo único que tiene una gran importancia. En según qué dominios, el entender las decisiones tomadas por un modelo de aprendizaje supervisado es tan o más importante que su rendimiento. Los modelos de aprendizaje automático y aprendizaje profundo están consiguiendo muy buenos resultados, pero, en general, resultan ser cajas negras de cara a los humanos en lo que a explicabilidad se refiere. La explicabilidad suele ser necesaria cuando hay cierto nivel de incompletitud, y esta surge cuando hay algo en el problema que no puede ser codificado en el modelo debido a su subjetividad, por ejemplo, en campos en los que la ética está en juego y debemos separar lo que es justo de lo que no. También aparece cuando no podemos testar al cien por cien nuestro modelo pues las características ambientales para que ello ocurra son imposibles de reproducir, o el propio ser humano y su forma de entender el mundo también lo dista de la lógica del modelo a la hora de operar.

Conforme el desarrollo de modelos más complejos avanza, el salto para poder entender cómo toman sus decisiones se hace más grande, dificultando comprender por qué fallan para ciertas instancias o impidiendo validarlos previamente a su implementación y despliegue. Por otra parte, en contextos donde las decisiones tengan consecuencias importantes será necesario saber el porqué de las mismas. Por ejemplo, el Reglamento General de Protección de Datos (RGPD), de aplicación en la Unión Europea y el Espacio Económico Europeo desde el 25 de mayo de 2018, exige que cualquier información o comunicación relativa al tratamiento de datos personales sea concisa, transparente, comprensible y de fácil acceso, utilizando un lenguaje claro y sencillo.

Se presentan dos conceptos: la interpretabilidad, que hace referencia a entender cómo actúa el modelo en su conjunto y la explicabilidad, que se utiliza cuando las explicaciones dadas por el modelo son incomprensibles de por sí. Estos dos conceptos juegan un papel muy importante en modelos de caja negra cuya configuración de los pesos se puede alejar del entendimiento humano.

Una explicación es relevante si responde a los objetivos y necesidades del usuario, tiene que aportar toda la información necesaria para cumplimentar el objetivo, pero no debe dar información de más. Tiene que convencer al usuario y eso ocurre si los hechos que la avalan son los mismos que él considera como veraces. Dependiendo del tipo de dato la forma de representación de la explicación deberá ser una u otra, además se busca que pueda ser simulable por un humano. Es decir, si cogemos los datos de entrada y realizamos el mismo razonamiento, debemos llegar a las mismas conclusiones que el modelo. A día de hoy son muchas las empresas que se fían de modelos entrenados con conjuntos de datos con sesgos que en las pruebas obtienen buenos resultados, pero que están claramente sesgados por su entrenamiento.

A parte de las razones expuestas hay otras que radican en la propia naturaleza humana que no dará el visto bueno de estas nuevas técnicas a menos que actúe con coherencia, o al menos de forma previsible. Parte de la no aceptación de los modelos de toma de decisión por parte de la sociedad proviene de la falta de explicabilidad, es algo muy humano el querer comprender el porqué de las decisiones. Factores como los que se citan a continuación fomentan la explicabilidad: la confianza que hace referencia a entender las fortalezas y debilidades del modelo, la causalidad siendo la importancia de los atributos de entrada de cara a tomar decisiones, la transferibilidad, al cerciorarse de la capacidad de generalización del modelo ya se puede comenzar a utilizar con datos desconocidos y ponerlo en producción, la informatividad, si sirve para nuevos ejemplos o sólo para aquellos para los que fue entrenado, toma de decisiones éticas y justas que se refiere a saber las consecuencias éticas y justas para las decisiones tomadas, así como saber a quién afectan, responsabilidad, el sistema debe tomar responsabilidad y explicar las decisiones tomadas, ajustar, la explicabilidad podría permitir a expertos poder ajustar el modelo y por último la funcionalidad proxy, que da explicabilidad a otros criterios que de por sí no pueden explicarse.

Hay dos formas de atajar este problema y están relacionadas con los modelos de caja blanca y caja negra expuestos anteriormente. Explicar un modelo como un todo o reemplazarlo por un modelo explicativo en sí como un árbol de decisión o dirigir el modelo a un estado más explicativo o buscar esa explicabilidad en sus predicciones individuales.

Capítulo 2

Explicabilidad

En el ámbito de la explicabilidad hay distintos aspectos que caracterizarán la forma de abordarla. El tipo de explicación, por ejemplo, es decir, local o global o el modelo concreto que se quiera utilizar. Es importante tener en cuenta las ventajas y desventajas de cada modelo pues acorde a ello podremos saber si por ejemplo el tipo de explicaciones podrá ser más concisa o que de pie a una mejor interpretación. [12] [11]

2.1. Dimensiones de la explicabilidad

La explicabilidad de un modelo se puede abordar desde distintas dimensiones, bien desde un punto de vista del funcionamiento del modelo en conjunto o bien desde la explicación de instancias individuales. También podemos diferenciar en base a si el propio modelo es interpretable de por sí o, por el contrario, es necesario o el uso de modelos «auxiliares» que nos permitan comprender su comportamiento. En las siguientes secciones se expondrán estos enfoques.

2.1.1. Generación de explicaciones

Tanto modelos interpretables como no interpretables necesitan lo que se conoce como un explicador, que se formaliza de la siguiente manera:

$$e : (X \rightarrow Y) \times (X \times Y) \rightarrow \varepsilon$$

El explicador toma un modelo y unos datos como entradas y provee de una explicación perteneciente al conjunto ε de todas las posibles explicaciones como salida.

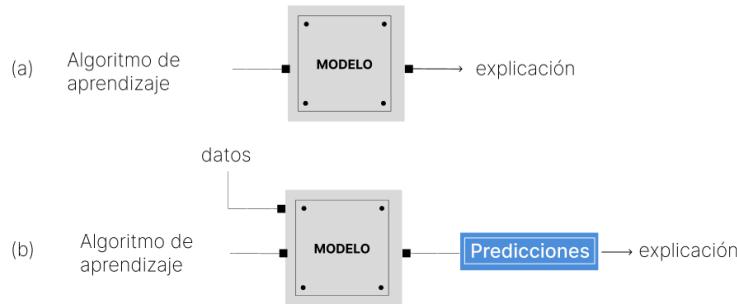


Figura 2.1 (a) Explicabilidad global las explicaciones se generan en base a un modelo que utilizará un algoritmo de aprendizaje concreto, (b) Explicabilidad local, por otro lado, en este caso las explicaciones se generarán a partir de las predicciones de un modelo

Dentro de esta dimensión encontramos dos posibles problemas: explicaciones globales y explicaciones locales. Como ilustra la figura 2.1, podemos estar interesados en dos tipos de explicaciones:

- ExPLICACIONES GLOBALES: la explicación se extrae de forma global del modelo y es representativa para un conjunto de datos ya sea para un modelo interpretable o no interpretable.
- ExPLICACIONES LOCALES: la explicación se genera a partir de una instancia en específico, x y a su correspondiente predicción y . Del mismo modo, es aplicable tanto para modelos de caja negra como modelos de caja blanca.

Los explicadores globales no dependen de las predicciones del modelo y son agnóstico con respecto a este último. Esto quiere decir que el mecanismo de interpretabilidad se puede aplicar a muchos modelos distintos. En ocasiones no requieren de un conjunto de datos para generar la explicación del modelo global.

Por su parte, las explicaciones locales se obtienen a partir de las predicciones generadas por un modelo. Dada una predicción para una instancia, el explicador local provee de una explicación que sólo es válida para esa instancia concreta.

2.1.2. Construcción de explicadores

Los modelos de caja blanca, como los árboles de decisión o los modelos de regresión lineal, al ser intrínsecamente interpretables, permiten ser usados como explicadores, globales o locales, de sí mismos. Sin embargo, en la actualidad los mejores resultados

prácticos se están obteniendo con modelos de aprendizaje profundo, que son modelos de caja negra.

Podemos abordar la interpretabilidad de modelos de caja negra con lo que se conoce como interpretabilidad *post-hoc*, en la que se construye un modelo subrogado que proporcione las explicaciones, pero sin descartar el modelo original, que se seguirá utilizando para realizar nuevas predicciones. De esta forma se conservará la precisión de las predicciones realizadas, a la vez que se tendrá acceso a explicaciones de las mismas. El modelo subrogado deberá ser un modelo de caja blanca, de tal manera que proporcione una interpretación del modelo de caja negra a partir de su interpretación intrínseca. Como se puede ver en la figura 2.2, al modelo subrogado se le proporcionan los mismos datos que al modelo de caja negra a interpretar, ya que los primeros son entrenados para aproximar las predicciones de los segundos.

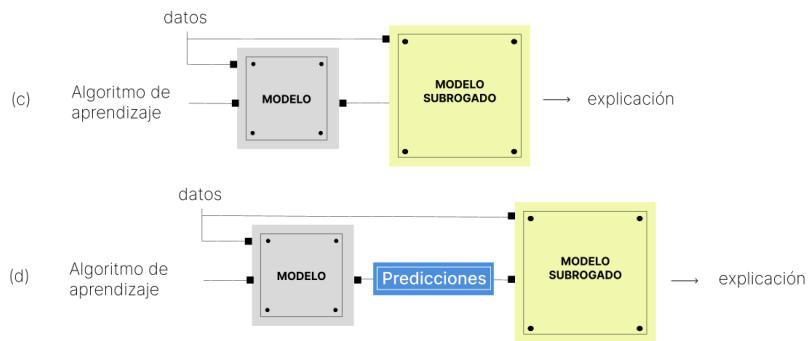


Figura 2.2 (c) Modelo subrogado global el modelo subrogado necesitará el propio modelo de caja negra como entrada además del conjunto de entrenamiento para generar la explicación (d) Modelo subrogado local, en este caso se partirá de las predicciones del modelo de caja negra y de los datos de entrenamiento para la creación del modelo subrogado

En el enfoque global (figura 2.2 (c)) el modelo subrogado se aproxima al modelo de caja negra en todos los datos de entrenamiento. De esta forma, se genera una explicación global del modelo a partir de todas las predicciones realizadas por el mismo.

En el enfoque local (figura 2.2 (d)) el modelo subrogado se aproxima al modelo de caja negra a partir de las instancias de prueba.

2.2. Técnicas de explicabilidad

El auge de los modelos de aprendizaje profundo y la necesidad asociada de su explicabilidad ha dado lugar al desarrollo de diversas técnicas para alcanzarla, algunas de las cuales se describen a continuación. [10]

2.2.1. Explicabilidad global

Importancia de atributos mediante permutaciones

Esta técnica mide la variación del error de predicción del modelo después de permutar los valores de un atributo, lo que rompe la relación entre el atributo y la salida real.

Se considerará un atributo como «importante» si variar su valor hace que se produzca un cambio en la salida del modelo, del mismo modo, será «no importante» si variar su valor no produce ningún cambio en la salida.

Explicación de la salida

Se explicará la salida a partir de un ejemplo práctico. Se ha obtenido la tabla de la figura 2.3 tras entrenar un modelo con datos estadísticos de fútbol.

Los atributos que se encuentran en puestos más altos serán interpretados como más importantes y viceversa para los que se encuentran más abajo en la lista.

El primer número de cada fila muestra el impacto de la permutación aleatoria de los valores del atributo en el rendimiento del modelo, es decir si el impacto es muy alto eso querrá decir que dicho atributo es importante. El segundo número indica la variación obtenida de calcular la métrica para ese atributo, no se puede pasar por alto la aleatoriedad que existe en las predicciones de un modelo, no siempre tiene por qué dar el mismo resultado exactamente. Además, se harán varias mezclas aleatorias lo que también hará que no haya una dependencia hacia la disposición concreta.

También puede haber valores negativos, entonces es cuando se obtienen mejores valores con los datos barajados aleatoriamente que con los datos iniciales.

Weight	Feature
0.1750 ± 0.0848	Goal Scored
0.0500 ± 0.0637	Distance Covered (Kms)
0.0437 ± 0.0637	Yellow Card
0.0187 ± 0.0500	Off-Target
0.0187 ± 0.0637	Free Kicks
0.0187 ± 0.0637	Fouls Committed
0.0125 ± 0.0637	Pass Accuracy %
0.0125 ± 0.0306	Blocked
0.0063 ± 0.0612	Saves
0.0063 ± 0.0250	Ball Possession %
0 ± 0.0000	Red
0 ± 0.0000	Yellow & Red
0.0000 ± 0.0559	On-Target
-0.0063 ± 0.0729	Offsides
-0.0063 ± 0.0919	Corners
-0.0063 ± 0.0250	Goals in PSO
-0.0187 ± 0.0306	Attempts
-0.0500 ± 0.0637	Passes

Figura 2.3 Ejemplo de importancia de atributos mediante permutaciones [3], se devuelve una lista con todos los atributos y la ponderación de cada uno, así como su variación.

Por tanto, en la figura 2.3 expuesta las variables más importantes acabarían siendo: *Goal Scored*, *Distance covered (Kms)* y *Yellow Card*.

Ventajas

- La interpretación es directa, global y muy comprimida.
- Tiene en cuenta el efecto del atributo de interés y las interacciones de éste con otros atributos. Lo cual también puede ser desventajoso pues la interacción se tiene en cuenta tanto en la suma total del atributo de interés en ese momento y del atributo que interacciona por lo que la suma de importancias se hace mayor.
- No requiere de un reentrenamiento.

Desventajas

- Este modelo está unido al error del propio modelo predictivo escogido. Dependiendo de la situación esto puede ser positivo o negativo, pero muchas veces se quiere medir la importancia dejando a un lado que esta variación afecte al buen rendimiento del modelo.
- Se necesitan los valores reales, es decir, los datos etiquetados.
- La forma en la que se permutan los datos añade aleatoriedad a la explicación. En relación con esto se pueden crear instancias que puedan no tener sentido en su

propia definición, por ejemplo, si estamos tratando con personas decir que una persona de 2 metros pesa 10 kilogramos.

- Añadir atributos correlacionados con otros atributos puede hacer que se divida la importancia otorgada a estos lo que no haría que se le diese la importancia que realmente tienen.

Modelo subrogado Global

Esta clase de acercamiento a la explicabilidad es agnóstica con respecto al modelo pues no se necesita saber nada sobre la estructura interna del modelo, tan sólo un conjunto de datos y las predicciones del modelo.

La mejor forma de comprender cómo funciona es a través de los pasos que hay que seguir para crearlo:

1. Se selecciona un conjunto de datos X , puede ser tanto el mismo que se utilizó para entrenar el modelo como uno nuevo, pero tiene que tener la misma distribución.
2. Para X obtenemos las predicciones del modelo de caja negra escogido.
3. Como modelo subrogado utilizaremos un modelo interpretable, (árboles de decisión, sistemas basados en reglas...).
4. Entrenamos el modelo de caja blanca con X y las predicciones del modelo de caja negra.
5. Una vez entrenado tenemos el modelo subrogado, podemos ahora medir como de bien se adapta el modelo de caja blanca al modelo de caja negra.
6. Interpretamos el modelo de caja blanca.

Ventajas

- Agnosticismo con respecto al modelo de caja negra.
- Flexibilidad en la selección del modelo subrogado.
- Facilidad de implementación.

Desventajas

- Se pueden tomar decisiones sobre los datos antes que sobre el modelo.

- Puede darse el caso de que el modelo subrogado funcione bien para un subconjunto de los datos, pero no para otros, lo que puede hacer que no tenga un buen rendimiento global.
- Hereda las ventajas y desventajas del modelo subrogado a implementar.

En los ejemplos de uso realizados en este trabajo se han utilizado como modelos subrogados globales los árboles de decisión y los sistemas basados en reglas.

Árboles de decisión

Los árboles de decisión partitionan el conjunto de datos múltiples veces en función de límites establecidos para los distintos valores que pueden tomar los atributos. En cada partición se va dividiendo el conjunto de entrenamiento, estos serían los nodos intermedios. Para predecir la salida de cada hoja final (es decir los nodos terminales del árbol) se utilizar la salida media de la misma.

Explicación de la salida

La explicación de la salida de este modelo es bastante intuitiva, dependiendo de las condiciones que se den en los nodos las instancias se irán dividiendo en subconjuntos, hasta llegar a los nodos finales en los que se obtiene el resultado de la predicción. Por ejemplo, en la figura 1.1 vemos como si se cumple que un animal puede volar y pone huevos, es categorizado como pájaro. No obstante, puede darse el caso de que, de otro nodo intermedio, con otras condiciones también se llegue a la misma predicción, por ejemplo, si hubiese un nodo que preguntase si tiene pico y clasificase entre pájaros y animales que no son pájaros.

Como se puede observar la explicación es bastante directa. La implementación que se utilizará será la de *scikit-learn* [6].

Sistemas basados en reglas

Una regla de decisión es una declaración condicional de tipo SI-ENTONCES. Por ejemplo, SI suben las temperaturas Y está nublado ENTONCES mañana llueve. Una regla de decisión o la combinación de estas puede ser usada para hacer predicciones.

Las reglas de decisión suelen seguir la siguiente estructura: SI se cumplen unas condiciones entonces se hace una determinada predicción. Esta clase de condiciones se asemejan al lenguaje natural y la forma que tenemos de pensar, teniendo en cuenta que la condición se construye con atributos inteligibles, la longitud de las condiciones es corta y no suele haber muchas reglas.

Algunos problemas que surgen son:

- Las reglas se pueden solapar, pues varias reglas pueden cumplirse.

- Ninguna regla puede cumplirse.

Hay dos estrategias principalmente para combinar múltiples reglas: listas de decisión (ordenadas) y conjuntos de decisión (no ordenados).

- Lista de decisión: da orden a las reglas. Si la condición de la primera regla es verdadera para una instancia, usamos la predicción de esta primera regla. Si no, vamos a la siguiente regla y comprobamos si se cumple y así sucesivamente. Las listas de decisión resuelven el problema de las reglas que se solapan, pues solo se devuelve la predicción de la primera regla en la lista que se aplique.
- Conjunto de decisión: se asemeja a la democracia de las reglas, excepto por el hecho de que algunas reglas tienen mayor poder de voto. En un conjunto las reglas son mutuamente excluyentes, o hay una estrategia para resolver los conflictos, como la mayoría de voto.

Existen distintos algoritmos para construir listas de decisión, habiéndose escogido para su utilización en este trabajo el de construcción de listas de reglas bayesianas.

Este algoritmo se hace una primera búsqueda de patrones frecuentes en los datos que serán usados para crear las condiciones de las reglas de decisión. Posteriormente, se aprende una lista de decisión a partir de una selección de las reglas extraídas anteriormente. Se hará uso de la implementación de *imodels*[4].

Explicación de la salida

La explicación de la salida es muy similar al caso de los árboles de decisión, si se cumplen las reglas escogidas en la lista de decisión, el resultado de la predicción será el resultado de la regla que se aplique. Las reglas generadas son para aquellas instancias cuya salida es 1 y están ponderadas en función de su importancia.

En las siguientes secciones donde se vean aplicaciones de este modelo a distintas tareas se expondrá cómo habría que adaptarlo pues originalmente está diseñado para clasificación binaria.

2.2.2. Explicabilidad local

Explicaciones locales interpretables independientes del modelo (LIME, por sus siglas en inglés)

LIME es un modelo subrogado local que, como se describió en la sección anterior, permite explicar predicciones individuales de un modelo de caja negra.

Para generar una explicación, LIME [9] analiza qué efectos se producen sobre las predicciones del modelo cuando se realizan variaciones en el conjunto de datos. Para ello, LIME crea un nuevo conjunto de datos basado en perturbaciones de los ejemplos que lo conforman y los datos predichos por el modelo de caja negra. A partir de este nuevo conjunto de datos LIME entrena un modelo interpretable, que se pondrá en función de la proximidad de las instancias muestreadas a la instancia de interés. Como modelos interpretables se pueden usar por ejemplo árboles de decisión. Según este procedimiento, el modelo interpretable será una buena aproximación del modelo de caja negra, pero estará basado en las instancias concretas del conjunto de datos de entrenamiento.

Los pasos a seguir para entrenar modelos subrogados son los siguientes:

- Seleccionar la instancia para la cual se quiere obtener la explicación.
- Alterar el conjunto de datos y conseguir las predicciones para los nuevos puntos generados.
- Ponderar las nuevas instancias generadas relaciona a la instancia de interés, es decir, la instancia de la que queremos obtener su interpretabilidad.
- Entrenar un modelo interpretable y ponderado a partir del conjunto de datos con variaciones.
- Explicar la predicción de la instancia de interés a partir del modelo interpretable.

La dificultad se puede encontrar en buscar la medida que nos defina la cercanía de una instancia con respecto a la instancia de interés. Este modelo utiliza un núcleo de suavizado exponencial para determinar la proximidad entre dos instancias.

Explicación de la salida

A continuación, se incluye un ejemplo del conjunto de datos que se utilizará para la tarea de clasificación binaria. En este caso LIME indica que esa vivienda es considerada como Básica, debido a que ni ha sido recientemente construida ni tiene patio, y el hecho de que tenga piscina ponderaría positivamente pero no lo suficiente como para ser de lujo.

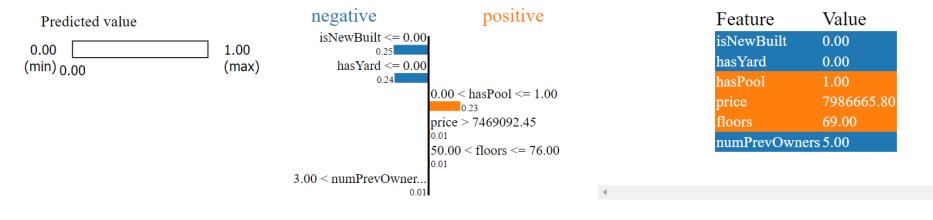


Figura 2.4 Explicación salida LIME, se proporciona la predicción del modelo de caja negra, junto con un gráfico que indica cómo afectan a la predicción cada atributo y las ponderaciones en la parte derecha.

Además, se incluye como salida la siguiente información:

```
Intercept 0.24875962370168486
Prediction_local [-0.00703704]
Right: 0
```

Figura 2.5 Explicación parámetros salida LIME. Se devuelven tres parámetros; *Intercept*, *Prediction local* y *Right*

Lo que indica *Intercept* es el valor que ha calculado LIME en base a sus ponderaciones, *Prediction Local* es la salida de la predicción local generada por el explicador y por último la salida original del modelo de caja negra.

Ventajas

- El modelo es caja negra es intercambiable pues la explicación se focaliza en las predicciones del mismo.
- Usando modelos sencillos como árboles de decisión sin demasiada profundidad las explicaciones son sencillas y concisas.
- Puede trabajar con datos tabulares, textos e imágenes.
- Hay múltiples bibliotecas en distintos lenguajes que lo implementan.
- Se puede basar en atributos derivados de los atributos presentes en el conjunto de entrenamiento para desarrollar sus explicaciones, por ejemplo, en el caso de datos de tipo texto se pueden usar los conocidos *embeddings* para la interpretabilidad.

Desventajas

- La definición de la vecindad, es decir como configuramos el núcleo de suavizado varía en función de cada escenario lo que en muchas ocasiones es un gran problema.

- A la hora de crear el nuevo conjunto de datos se hace en función de una distribución gaussiana, ignorando totalmente la correlación entre atributos, lo que puede derivar en explicaciones equivocadas.
- Las explicaciones presentan gran inestabilidad. Por ejemplo, las explicaciones para dos instancias cercanas pueden ser muy diferentes entre sí. Las explicaciones también pueden variar al reiterar el proceso de alteración del conjunto de datos.

Explicaciones aditivas de Shapley (SHAP, por sus siglas en inglés)

Se basa en los valores de *Shapley*. Su objetivo es explicar la contribución de una instancia a partir de la contribución de los atributos que lo conforman. El funcionamiento se puede entender a través de la teoría de juegos de coalición. En este caso los jugadores serán los atributos que conforman una instancia si es el caso de datos tabulares, no obstante, también puede ser un grupo de atributos.

Antes de poder explicar algunas de las variantes de *SHAP* se ha visto conveniente explicar en qué consisten los valores de *Shapley*.



Figura 2.6 Ejemplo valores de *Shapley*, donde se relacionan instancias con distintos valores en sus atributos con el precio que toman como salida.

Pongamos el caso de que tenemos un conjunto de datos con distintas propiedades de una casa, si tiene garaje, piscina y si es una casa. Como se puede observar en la figura 2.6 el hecho de que tenga garaje incrementa en 30 000 € el valor total de la vivienda.

Este caso es un ejemplo simplificado, habría que hacer todas las posibles coaliciones con todos los posibles valores que pueden tomar el resto de atributos.

A continuación, se explicarán dos formas de implementar este modelo, ambas variaciones se encuentran implementadas en la biblioteca [2].

KernelShap

Con esta técnica lo que se hace es generar un vector de binario en los que los unos indican aquellos atributos para los cuales se está estudiando su valor, es decir el atributo que queremos explicar y 0 en caso contrario. Para datos tabulares para aquellos atributos que haya ceros se muestrearan valores escogidos aleatoriamente. Estos vectores de ceros y unos se denominan coaliciones.

	COALICIONES			VALORES DE LOS ATRIBUTOS			
	EDAD	PESO	COLOR		EDAD	PESO	COLOR
Instancia x	1	1	1	→	8	30	AZUL
Instancia x con valores sustituidos	1	0	0	→	8	25	VERDE
					30	AZUL	

Figura 2.7 En la figura se ve cómo los valores que toman cero se sustituyen. Tiene la misma problemática que el método de importancia de atributos mediante permutaciones expuesto anteriormente, en la creación de nuevas coaliciones pueden aparecer combinaciones anómalas que acaban teniendo más peso del que deberían. La solución que se le da es en vez de muestrear el atributo que toma valor cero (en la ilustración) con un valor aleatorio, que tome valor igual a cero.

A diferencia de LIME que pondera las instancias en función de su cercanía a la instancia original este método lo hace en función del valor que la coalición obtendría en una estimación usando valores de *Shapley*. De esta forma se aprenderá cómo actúa una característica en solitario si la coalición se presenta con todo ceros menos en la característica de estudio. Si una coalición consiste de todo unos menos un cero se aprenderá el efecto total de este atributo.

TreeShap

Es una variante de SHAP para modelos basados en árboles de decisión, desde árboles de decisión clásicos hasta *Random Forest*. Esta técnica es computacionalmente menos costosa. *TreeSHAP* utiliza la esperanza condicional.

A continuación, se explicará brevemente la mecánica que sigue este algoritmo para un único árbol, una instancia x y un conjunto S de atributos. Si condicionamos todos los atributos, entonces la predicción asignada a ese nodo será la predicción esperada, es decir, la etiqueta de esa instancia. Si no condicionamos la predicción a ningún atributo, es decir S está vacía, usaremos como valor de la predicción la media ponderada de todas las posibles predicciones de los nodos terminales. Si S contiene algunos, pero no todos los atributos se ignoran los atributos de nodos no alcanzables. Un nodo no alcanzable significa que el camino de decisión para llegar a dicho nodo contradice valores tomados en x_S que sería la instancia que contiene tan sólo los atributos contenidos en el

subconjunto S . Al resto de nodos se le hace la media de las predicciones ponderada por el tamaño de sus nodos. La media de los nodos terminales, ponderada por el número de instancias por nodos, es la predicción esperada de x dado S .

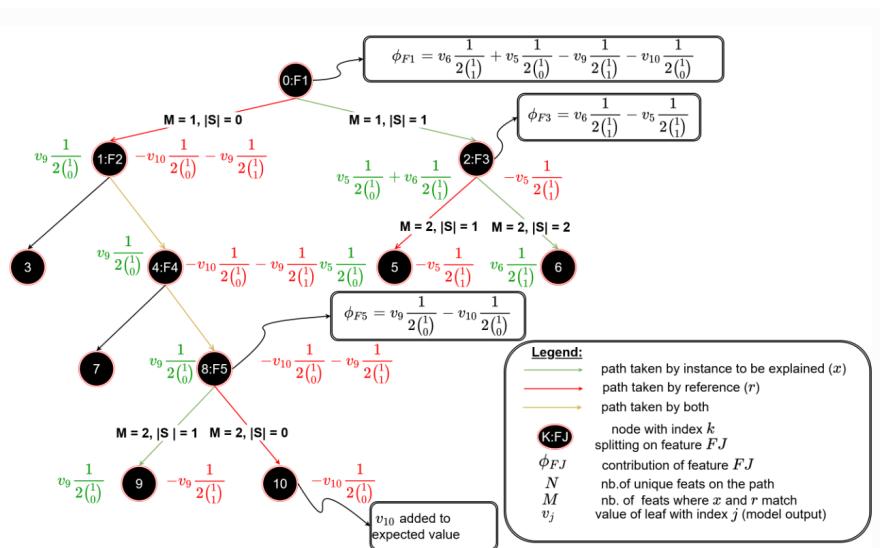
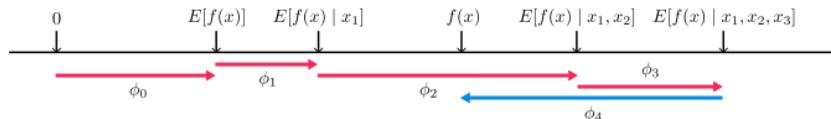


Figura 2.8 TreeSHAP explicación gráfica. Por medio de la generación del árbol se calculan los valores de *Shapley*.

Explicación de la salida



SHAP (SHapley Additive exPlanation) values explain the output of a function f as a sum of the effects ϕ_i of each feature being introduced into a conditional expectation. Importantly, for non-linear functions the order in which features are introduced matters. SHAP values result from averaging over all possible orderings. Proofs from game theory show this is the only possible consistent approach where $\sum_{i=0}^M \phi_i = f(x)$. In contrast, the only current individualized feature attribution method for trees satisfies the summation, but is inconsistent because it only considers a single ordering.

Figura 2.9 Explicación salida SHAP. SHAP explica la salida para una función f como la suma de los efectos de ϕ de cada atributo.

La figura 2.9 muestra un esquema de cómo se interpreta su representación, se encuentran indicadas las esperanzas condicionales y como los *shapley values* (ϕ) afectan al valor de la predicción. Se acostumbra a representar en rojo los valores que incrementan el valor de la predicción positivamente y en azul los que lo decrementan.

Además, en SHAP suele hablarse de un valor base que sería la media de las salidas del modelo a partir del conjunto de datos de entrada.

Ventajas

- Tiene una base teórica sólida en teoría de juegos. Se entiende la predicción como justamente distribuida entre los atributos. Las explicaciones se pueden contrastar y comparar con la predicción media, que es proporcionada.
- Es una combinación entre LIME y *Shapley values* lo que lleva a la consolidación y buen entendimiento de ambos métodos.
- Pese al costo exponencial de *KernelSHAP* la implementación de *TreeSHAP* es mucho menos costosa.
- Tiene implementaciones tanto locales como globales, y como ambos se basan en valores de *Shapley* ambas interpretaciones son consistentes.

Desventajas

- El alto costo computacional de *KernelSHAP* hace que sea lento.
- *KernelSHAP* ignora la dependencia entre las variables, la sustitución por valores aleatorios es parte del problema. *TreeSHAP* lo resuelve.
- *TreeSHAP* resuelve el problema anterior, pero hace que atributos que no tienen peso en la predicción tengan un valor distinto de cero al usar la esperanza condicional.

Capítulo 3

Ejemplos de Uso

En este capítulo se detallarán casos de usos para algunas de las técnicas expuestas en la sección anterior. Se han seleccionado tres conjuntos de datos distintos, uno para cada tipo de área considerada: clasificación binaria, clasificación multiclas y por último regresión. Cada conjunto de datos ha sido utilizado en modelos tanto de caja blanca como de caja negra. De entre los modelos de caja blanca utilizados encontramos: árboles de decisión y sistemas basados en reglas. En el caso de los modelos de caja negra se utilizarán: bosques aleatorios y redes neuronales. Para inferir la explicabilidad de los modelos de caja negra se utilizarán las siguientes técnicas: modelo subrogado, LIME, SHAP, importancia de atributos mediante permutaciones e importancia de atributos (derivado de bosques aleatorios).

Se muestra a continuación un esquema de cuál será la estructura que se llevará a cabo a la hora de exponer los distintos tipos de modelos explicables:



Figura 3.1 Esquema explicabilidad, se desglosa entre tipo de tarea a llevar a cabo, el modelo de caja negra a utilizar y el método de explicabilidad utilizado.

3.1. Modelos de caja negra utilizados

3.1.1. Bosque aleatorio

Es un modelo que usa a técnica de *ensamble* de modelos, es decir, con el objetivo de obtener un mejor rendimiento, sus predicciones se determinan a partir de las predicciones realizadas por un conjunto de modelos. En ese caso tratándose del modelo bosque aleatorio, se basa en múltiples árboles de decisión y utiliza muestreo con reemplazamiento (*bootstrap*) de los ejemplos de entrenamiento y aleatoriedad de características para crear árboles de decisión no correlacionados.

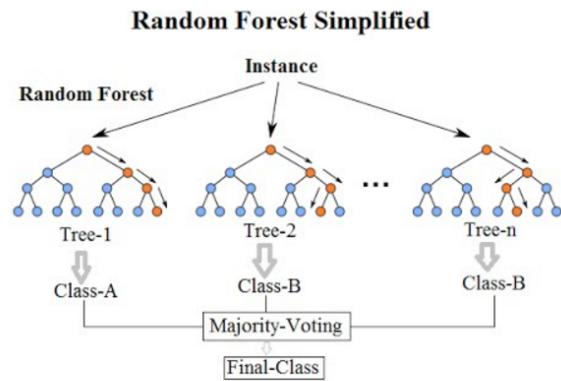


Figura 3.2 Explicación esquema del modelo bosque aleatorio. Se crean distintos árboles y las predicciones de cada uno se resumen utilizando un sistema de voto

La técnica de aleatoriedad de características crea un subconjunto con baja correlación entre los árboles de decisión. Mientras que un árbol de decisión considerará todas las características en esta técnica se toma un subconjunto de los atributos para cada nodo. Además, cada árbol del bosque se entrenará con un subconjunto de los datos. Y posteriormente en la etapa de validación se combinan los resultados de todos los árboles y así obtener la estructura de un árbol con mejor rendimiento sobre los datos. Se hará uso de la implementación de *scikit-learn* [7][8].

3.1.2. redes neuronales

Una red neuronal es un grafo dirigido donde los nodos están ocupados por las neuronas y las aristas representan las conexiones entre neuronas. Las conexiones están etiquetadas con un peso. Usualmente, las neuronas se estructuran en capas, las neuronas pertenecientes a una misma capa no se encuentran conectadas entre sí. La disposición de

nodos y aristas en la red constituye su arquitectura. Hay muchos tipos de arquitectura y cada una se adaptará al problema a resolver, al tipo de tarea a realizar o al conjunto de datos, entre otros factores.

En la figura 3.3 se muestra una red neuronal de 4 capas, la de entrada, la salida y las dos ocultas. En el tipo de red neuronal que se utilizará, las redes neuronales con alimentación hacia delante, cada neurona de una capa se encuentra conectada con todas las neuronas de la siguiente capa. Cada conexión tendrá un peso asociado que será lo que aprenderá la red durante el entrenamiento.

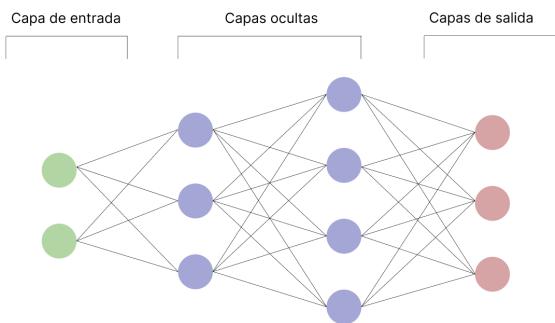


Figura 3.3 Ejemplo de red neuronal. Esta red está compuesta por dos neuronas de entrada, dos capas densas de 3 y 4 neuronas respectivamente y una capa de salida formada por 3 neuronas.

Las redes neuronales han sido desarrolladas con la implementación de Keras [5].

3.2. Tarea: clasificación binaria

3.2.1. Conjuntos de datos utilizado

Una tarea de clasificación binaria consiste en clasificar elementos en dos grupos. El conjunto de datos escogido para esta tarea es Clasificación de la vivienda en París¹, este conjunto de datos se ha creado a partir de datos inventado de precios de casa en un entorno urbano, concretamente París. Se escogió este conjunto pues la mayoría de sus atributos (a excepción de la columna objetivo) tenían valores numéricos y sin excesiva complicación para poder abordar los casos bases de la explicabilidad en primer lugar.

Como se hará con el resto de conjuntos de datos en primer lugar se hará un limpiado del mismo, comprobando si hay valores que haya que imputar o columnas no numéricas

¹<https://www.kaggle.com/datasets/msssmartypants/paris-housing-classification>

que transformar (este paso permitirá usar árboles de decisión que requieren que los datos tabulares sean numéricos). En la figura 3.4 vemos un ejemplo de algunas de las columnas previo al procesamiento. Y en la figura 3.5 tras el procesamiento de los datos.

index	squareMeters	numberOfRooms	hasYard	hasPool	floors	cityCode	...	category
0	75523	3	0	1	63	9373	...	Basic
1	80771	39	1	1	98	39381	...	Luxury
2	55712	58	0	1	19	34457	...	Basic
3	32316	47	0	0	6	27939	...	Basic
4	70429	19	1	1	90	38045	...	Luxury

Figura 3.4 Ejemplo de datos pre-procesado de Clasificación Binaria. Algunos de los atributos como *category* eran de tipo categórico y no numérico por lo que hubo que transformar algunas columnas.

index	squareMeters	numberOfRooms	hasYard	hasPool	floors	cityCode	...	category
0	75523	3	0	1	63	9373	...	0
1	80771	39	1	1	98	39381	...	1
2	55712	58	0	1	19	34457	...	0
3	32316	47	0	0	6	27939	...	0
4	70429	19	1	1	90	38045	...	1

Figura 3.5 Ejemplo de datos post-procesado de Clasificación Binaria. Aquí el atributo *category* ya toma valores numéricos, cero o uno.

3.2.2. Modelo de caja negra: modelo de bosque aleatorio

Detalles de la implementación

En lo que se refiere a la implementación se utilizó el modelo con la configuración por defecto.

Rendimiento

Para esta tarea y este modelo se consigue un rendimiento del 100 % sobre el conjunto total de datos. Como la explicabilidad se va a hacer sobre todos los datos no se buscaba probar el modelo con datos nuevos.

Modelo de explicabilidad: modelo subrogado

Para abordar la explicabilidad usando un modelo subrogado lo que se hizo fue usar el mismo conjunto de datos utilizado para el entrenamiento del modelo de caja negra (el conjunto de datos expuesto en la sección 3.2.1) y tras predecir todas las instancias

de dicho conjunto de datos se han utilizado como variable objetivo para el modelo subrogado.

Como modelo de caja blanca se han utilizado dos modelos distintos: árboles de decisión y sistemas basados en reglas.

Árbol de clasificación binaria

Se entrenó un árbol de decisión y posteriormente se obtuvo su estructura. En la figura 3.6 se ve que los atributos que se están utilizando para clasificar las instancias serán: si tiene piscina, si ha sido recientemente construida y si tiene jardín. En una rápida visual vemos como el índice de Gini en todas las hojas terminales es igual a 0, lo cual es un buen resultado pues lo que se busca es minimizar la impureza. Puesto que en ese algoritmo se sitúan a la derecha aquellas instancias que no cumplen la condición y a la izquierda las que sí. Por tanto, se puede deducir que el hecho de que no se tenga piscina es condición suficiente en algunos casos para considerarse vivienda básica. Seguidamente se comprueba si ha sido recientemente construida, de ser así será considerada como vivienda básica, en otro caso se obtiene una impureza de 0.5. Por último, se clasifica en función de si tiene patio o no, en el caso de no tener serán viviendas básicas, en otro caso de lujo.

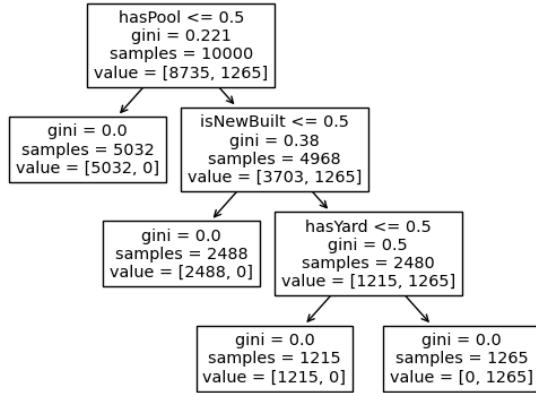


Figura 3.6 Modelo subrogado: árbol de decisión clasificación binaria con bosque aleatorio. Se crea un árbol de decisión en el que casas con piscina, sean recientemente construidas y tengan patio son clasificadas como lujosas y como básicas de no cumplir alguna de estas condiciones.

En primera instancia, es coherente que el modelo categorice como no lujoso directamente por el hecho de no tener piscina. No obstante, el modelo subrogado indica que para el modelo de bosque aleatorio esta no es una condición suficiente para ser una casa lujosa. Se podría considerar como condiciones suficientes tener piscina, ser

recientemente construida y tener jardín pues de ser así para el conjunto de datos con el que ha sido entrenado el modelo se consigue una impureza de 0.

Sistemas de Reglas

Se entrenó un sistema basado en reglas con los datos del conjunto de datos y las predicciones del modelo de bosque aleatorio, tras visualizar el modelo se obtiene las reglas que se indican en la figura 3.7.

La que mayor peso (22) se le ha dado es aquella que combina que tenga jardín, tenga piscina y haya sido recientemente construida. Si nos fijamos en el árbol de decisión de la figura 3.6 vemos cómo esa condición se corresponde con una de las ramas. Las reglas 20 y 21 tienen una ponderación mucho menor y de hecho no podemos encontrar una rama que represente ninguna de esas dos reglas. Sin embargo, las reglas con índice 19, 17 y 18 se encuentran ponderadas negativamente, indicando que esas condiciones no contribuyen a ser clasificadas con valor 1 es decir viviendas lujosas.

	rule	coef
8	made	-0.01
19	hasPool <= 0.5	-0.15
17	isNewBuilt <= 0.5	-0.35
18	hasYard <= 0.5	-0.11
21	hasYard > 0.5 and hasPool > 0.5	0.07
20	hasYard > 0.5 and isNewBuilt > 0.5	0.02
22	hasYard > 0.5 and hasPool > 0.5 and isNewBuilt > 0.5	22.07

Figura 3.7 Sistema basado en reglas Bosque aleatorio clasificación binaria. Se han generado 6 reglas (la primera no tiene condición), pero 1 22 es la que con diferencia se ha ponderado más alta, coincide con lo expuesto en el modelo subrogado.

Modelo de explicabilidad: LIME

Para esta técnica utilizaremos el modelo ya entrenado y los datos originales y lo que obtendremos serán explicaciones locales.

En primer lugar, consideremos una casa que el modelo de bosque aleatorio predice que es de tipo básica. La figura 3.8 representa la explicabilidad local proporcionada por LIME, mostrando en azul aquellos atributos que ha determinado característicos de casas básicas y en naranja los que ha determinado característicos de casas lujosas. Se observa entonces que la explicación de que se haya predicho que la casa es básica es porque, aunque sí tiene piscina, sin embargo no tiene patio ni ha sido recientemente construida.

Hay otros atributos que también aparecen en la explicabilidad, aunque con un peso muy cercano a cero. Uno de ellos es el precio de la casa, lo que en principio parece tener sentido, pues se suele considerar a las casas lujosas como casas de precio elevado. Esto, sin embargo, es un artefacto de la aleatoriedad inherente al modelo de explicabilidad local LIME y, de hecho, aplicaciones repetidas del modelo a este ejemplo no siempre incluyen el atributo precio en la explicabilidad. Por otra parte, los atributos tener patio, ser de nueva construcción y tener piscina siempre están presentes en las explicaciones y con un peso elevado. Nótese cómo esto es consistente con lo indicado por los modelos subrogados globales.

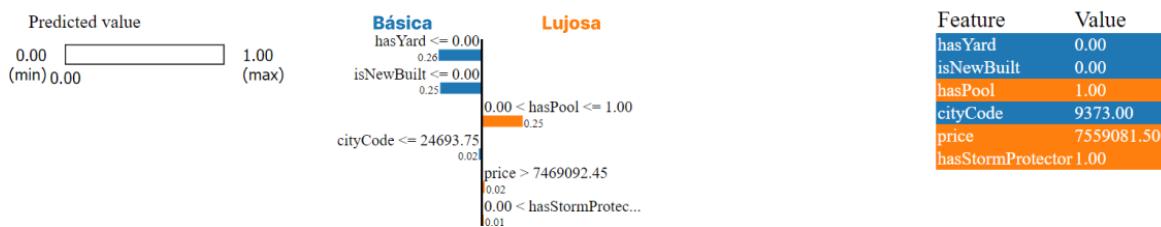


Figura 3.8 Explicación LIME bosque aleatorio 1 clasificación binaria. Los atributos mayormente ponderados son si tiene jardín, si tiene piscina y si ha sido recientemente construida, y como no se cumplen los tres se considera básica.

Consideremos ahora una casa que el modelo de bosque aleatorio predice que es de tipo lujosa.

En la figura 3.9 se observa cómo los atributos que se tuvieron en cuenta con mayores pesos en el ejemplo anterior se siguen escogiendo para explicar. Se ha tomado esta explicación en concreto pues de nuevo se interpreta el atributo precio como determinante, pero en este caso además espera mayores precios para casas básicas. La razón por la que el tener en cuenta este atributo y con ese umbral no afecta es porque le confiere poco peso y al final son los tres primeros atributos los que acaban realmente categorizando la vivienda.

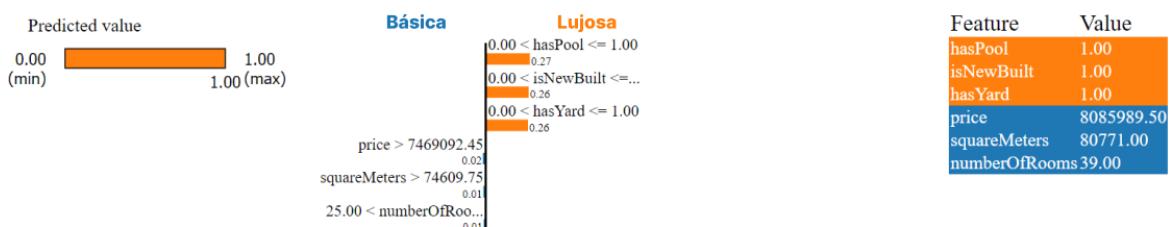


Figura 3.9 Explicación LIME bosque aleatorio 2 clasificación binaria. Se muestran en el gráfico una alta ponderación para los atributos pertenecientes a viviendas de lujo y una ponderación casi inexistente para el resto.

Modelo de explicabilidad: SHAP

De entre las variantes que se han expuesto anteriormente, debido a que el modelo de bosque aleatorio se basa en árboles de decisión se utilizará **TreeSHAP**.

En la figura 3.10, se han representado las contribuciones de SHAP para 4 casas y se han ordenado estas en función de la similaridad con respecto a su valor en la variable respuesta, es decir se sitúan al principio las instancias consideradas como lujosas y seguidamente las básicas. El propio método de la biblioteca permite ordenar según similaridad u orden original de entrada de las instancias entre otras ordenaciones.

Por tanto, en las dos primeras casas que son categorizadas como lujosas, las razones que da es que tiene piscina, ha sido recientemente construida y tiene jardín.

Las siguientes instancias obtienen valores próximos a ceros pues son básicas. Y esto es debido a que ni han sido recientemente construidas, ni tienen jardín, sin embargo, algunos tienen piscina lo que hace que el valor no se mantenga en cero.

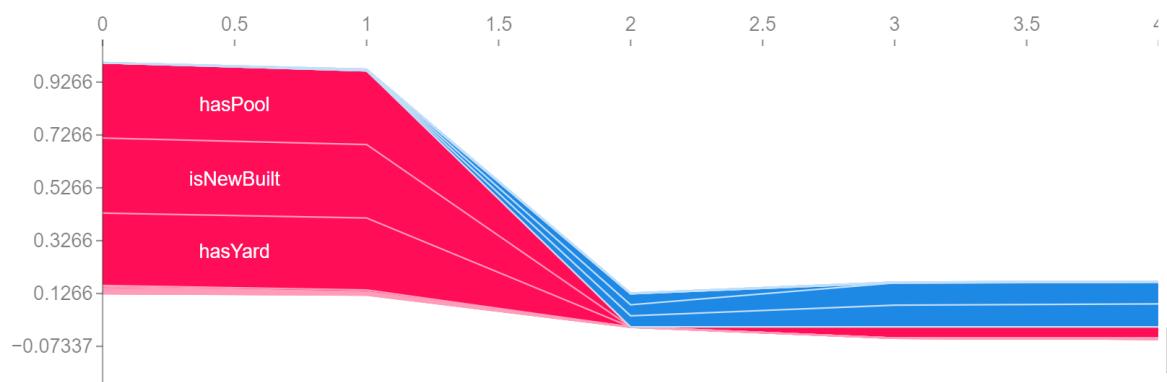


Figura 3.10 Explicación shap bosque aleatorio 1 clasificación binaria. En el gráfico se indican las contribuciones de cada atributo según cada instancia. En el eje x las instancias seleccionadas y en el eje y sus ponderaciones.

De nuevo, las explicaciones se vuelven a basar principalmente en los tres atributos mencionados en secciones anteriores.

Modelo de explicabilidad: permutación de atributos

El siguiente modelo de explicabilidad que se utilizará es la permutación de atributos. En la figura 3.11 se ve cómo de nuevo los atributos que modelan la explicación son los tres con mayor peso en las secciones anteriores. Al ser un modelo de explicabilidad global se obtiene una explicación para todas las instancias. Lo cual explica que no se le dé siquiera un peso bajo a otros atributos, pues ya se vio en modelos como LIME que incluso de una ejecución a otra los pesos más bajos de los atributos que modelaban la

explicación iban variando. Es por esto, que es coherente que como explicación global no se tengan en cuenta.

Weight	Feature
0.1274 ± 0.0035	isNewBuilt
0.1235 ± 0.0021	hasPool
0.1233 ± 0.0049	hasYard
0 ± 0.0000	price
0 ± 0.0000	numPrevOwners
0 ± 0.0000	numberOfRooms
0 ± 0.0000	floors
0 ± 0.0000	cityCode
0 ± 0.0000	cityPartRange
0 ± 0.0000	made
0 ± 0.0000	hasGuestRoom
0 ± 0.0000	hasStormProtector
0 ± 0.0000	basement
0 ± 0.0000	attic
0 ± 0.0000	garage
0 ± 0.0000	hasStorageRoom
0 ± 0.0000	squareMeters

Figura 3.11 Permutación de atributos bosque aleatorio clasificación binaria. Se muestra la lista de atributos con sus pesos y variaciones, estando cualquier atributo que no sean los tres lujoso ponderados a 0.

Modelo de explicabilidad: importancia de atributos

Por último, usaremos el método derivado del entrenamiento de un modelo de bosque aleatorio para continuar con el análisis de la explicabilidad. De nuevo es un método global.

En la figura 3.12 vemos una lista ordenada por peso de los atributos. Se repite la norma que se venía observando y se confirma la explicación que se dio a que en la permutación de atributos no tuviesen pesos el resto de atributos. Los valores que toman el resto de atributos están en torno a las milésimas todos, la importancia se ha dividido entre estos pues realmente el hecho de que se tome como importante o no cada uno de los restantes atributos depende de la instancia que se examine.

```
[('isNewBuilt', 0.333995961141522),
 ('hasYard', 0.30769662185933017),
 ('hasPool', 0.2756160018977986),
 ('basement', 0.0086819210210636),
 ('attic', 0.008268457621682853),
 ('cityCode', 0.008142885467968706),
 ('garage', 0.007872886395251778),
 ('floors', 0.00782569611904514),
 ('squareMeters', 0.007396994514268313),
 ('price', 0.007025439090581588),
 ('numberOfRooms', 0.006965754721661879),
 ('made', 0.006170098569909042),
 ('cityPartRange', 0.00436400658126414),
 ('hasGuestRoom', 0.004053891782507454),
 ('numPrevOwners', 0.003995926790548638),
 ('hasStorageRoom', 0.0010717779472981597),
 ('hasStormProtector', 0.0008556784782978866)]
```

Figura 3.12 Importancia de atributos para bosque aleatorio clasificación binaria. Se repite el patrón visto en secciones anteriores, en este caso la ponderación del resto de atributos no es 0

3.2.3. Modelo de caja negra: redes Neuronales

Detalles de la implementación de la red

En esta sección se detallará la arquitectura de la red. La red neuronal estará formada por una primera capa de normalización por lotes que aplica una transformación que mantiene la media de las salidas próximas a 0 y la desviación media cercana a 1. Después le sigue una capa densa de 10 neuronas con función de activación ReLU y como entrada 17 parámetros que serán los 17 atributos del conjunto de entrenamiento a partir de los cuales se hará la predicción. Por último, como salida tiene una sola neurona con función de activación sigmoide. Como optimizador se utilizará Adam y como función de pérdida la entropía cruzada binaria.

Rendimiento

Con esta implementación se consigue una precisión y un *recall* del 100 % sobre el conjunto total de datos.

Modelo de explicabilidad: modelo subrogado

Árbol de clasificación binaria

Del mismo modo que con el modelo de bosque aleatorio se utilizó un árbol de decisión como modelo subrogado. En la figura 3.13 podemos ver cómo el modelo

obtenido es el mismo árbol de decisión que se obtuvo para el bosque aleatorio. Por tanto, la explicación global del modelo de red neuronal coincide con la del anterior modelo expuesto, esto quiere decir, por tanto, que basan sus decisiones en los mismos atributos.

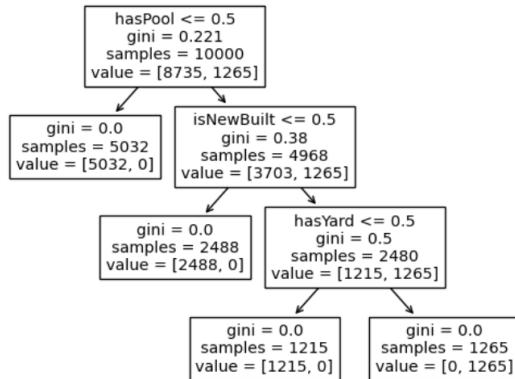


Figura 3.13 Modelo subrogado: árbol de decisión clasificación binaria con red neuronal. El árbol es idéntico al obtenido con el modelo de bosque aleatorio

Sistemas de Reglas

De nuevo en la figura 3.14 vemos cómo la regla con mayor ponderación es aquella que se corresponde a la rama del árbol que clasifica las viviendas lujosas (la regla 22). Se introduce una regla similar, pero en la que se tienen en cuenta los metros cuadrados, esto puede ser debido a que hay un conjunto de viviendas lujosas en las que además de cumplirse las otras tres condiciones se cumple el patrón de tener un número de metros cuadrados mayor a uno dado. No obstante, la ponderación que se le da es muy baja en comparación con la regla de mayor ponderación. Realmente, a través del árbol de decisión construido en la sección anterior se deduce que con una sola regla se podrían clasificar correctamente todas las casas lujosas. En la técnica de reglas bayesianas (que es en la que se basa la biblioteca que se está utilizando) se generan reglas para los ejemplos que tengan como salida 1, no se generarán reglas para ponderar casas básicas pues para ello la salida tendría que ser 0. De hecho, con la regla de mayor ponderación generada, seríamos capaces de clasificar todas las casas lujosas.

	rule	coef
8	made	-0.01
17	hasPool <= 0.5	-0.13
18	isNewBuilt <= 0.5	-0.09
20	hasYard > 0.5 and isNewBuilt > 0.5	0.19
19	hasPool > 0.5 and isNewBuilt > 0.5	0.01
22	hasYard > 0.5 and hasPool > 0.5 and isNewBuilt > 0.5	20.45
21	squareMeters <= 59967.5 and hasYard > 0.5 and hasPool > 0.5 and isNewBuilt > 0.5	0.24

Figura 3.14]

Explicación sistema basado en reglas red neuronal clasificación binaria. La regla que se pondera con mayor peso es aquella que contiene las condiciones para clasificar una vivienda lujosa.

Modelo de explicabilidad: LIME

La figura 3.15 muestra la explicabilidad local, por el modelo LIME, para una casa predicha por la red neuronal como casa básica. De nuevo se observa que la clasificación depende principalmente de los atributos ser de reciente construcción, tener piscina y tener patio, que son los de mayor peso. El resto de atributos que aparecen en la explicación tienen peso prácticamente nulo.

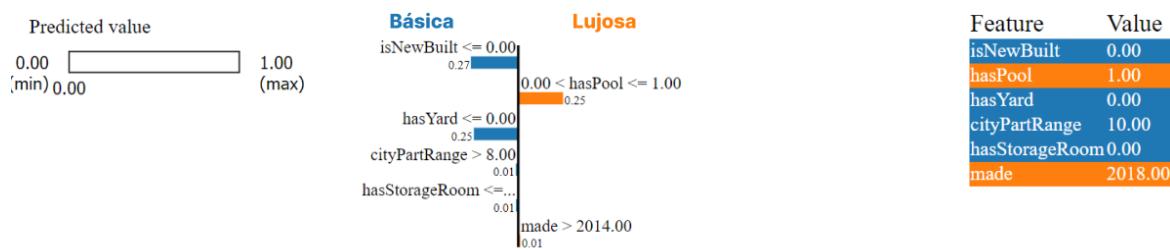


Figura 3.15 Explicación LIME clasificación binaria red neuronal 1. Esta instancia se pondera de forma similar a cuando se hizo con el modelo de bosque aleatorio. Se consideran atributos lujosos tanto tener piscina como el año de construcción, pero este último está tan bajo ponderado que no es determinante.

Pasando a la siguiente figura 3.16 en la que se clasifica una vivienda lujosa, es fácilmente clasificada pues contiene los tres atributos característicos.

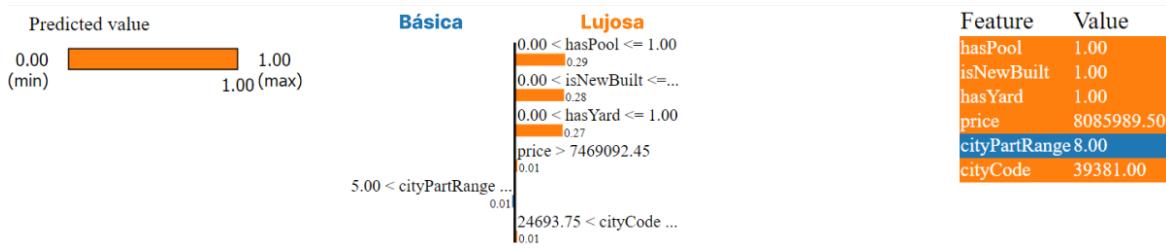


Figura 3.16 Explicación LIME clasificación binaria red neuronal 2. Esta vivienda está totalmente considerada como lujosa pues cumple todas las condiciones.

Se ha probado a continuación con un ejemplo en el que la vivienda tenía piscina y ha sido recientemente construida, si recordamos el árbol generado en la figura 3.13 era uno de los caminos para clasificar como vivienda básica, y este modelo, pese a tener dos atributos considerados como de vivienda lujosa sigue clasificando correctamente como vivienda básica 3.17.

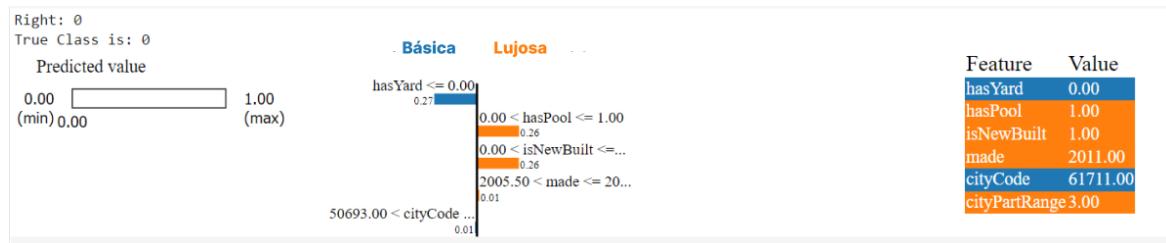


Figura 3.17 Explicación LIME clasificación binaria red neuronal 3. Pese a cumplir dos de las condiciones lujosas que además tienen una alta ponderación la vivienda sigue considerándose como básica.

También se probó para casos en los que se tuviesen otros dos atributos de los tres posibles atributos (tener piscina, ser recientemente construida y tener jardín) y en todas las viviendas eran básicas, en definitiva, sólo se considera una vivienda lujosa cuando tiene los tres atributos, y eso es lo que están aprendiendo los modelos.

Modelo de explicabilidad SHAP

En este caso, al tratarse de una red neuronal y no un modelo basado en un árbol de decisión se usará la variante de SHAP: **KernelSHAP**.

En esta figura 3.19 en la que se ordenan las instancias por similaridad la salida es casi idéntica a la obtenida del modelo de bosque aleatorio.

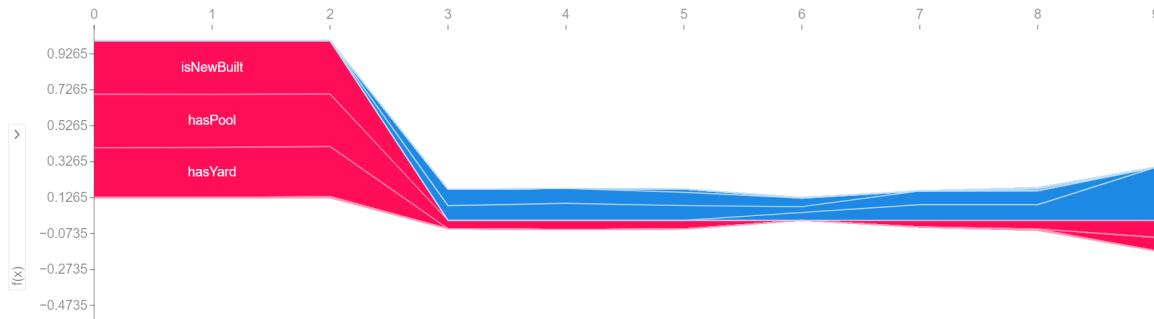


Figura 3.18 Explicación shap red neuronal clasificación binaria 1. Se disponen en el eje x las instancias para los que se calculará SHAP y en el eje y la ponderación de cada atributo.

Se prueba con distintos conjuntos de instancias y siempre son esos tres atributos los que determinan la predicción de la instancia.

Además, en la figura que se muestra a continuación se ha escogido ejemplos en los que parejas formadas a partir de los tres atributos que se usan para la clasificación tienen valor 1. En la figura, aparecen en primer lugar dos viviendas básicas, se ve como las viviendas tienen un mayor puntuaje que en la figura 3.19.

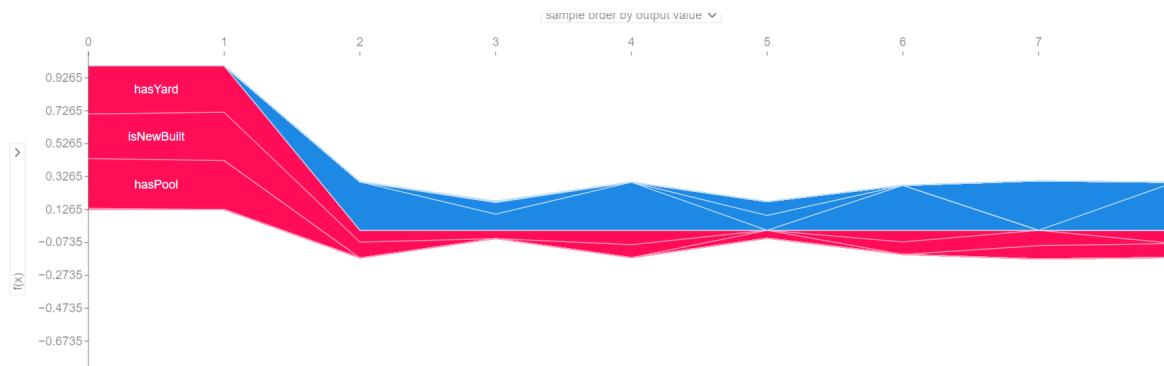


Figura 3.19 Explicación SHAP red neuronal clasificación binaria 2. Las primeras instancias son clasificadas como lujosas pero las siguientes son básicas y oscilan las ponderaciones de sus atributos pues algunas contienen algunos atributos lujosos y otras no

Modelo de explicabilidad: permutación de atributos

Los resultados obtenidos para esta técnica son casi idénticos a los obtenidos con el modelo de bosque aleatorio, varía en el peso que se le da a cada atributo, en general se pondera cuatro veces más alto en el modelo de red neuronal que en el modelo de

bosque aleatorio. Se hicieron varias ejecuciones para comprobar si las variaciones no dependían tanto del modelo si no de estas mismas, pero los resultados seguían siendo los mismos.

Weight	Feature
0.4953 ± 0.0284	isNewBuilt
0.4923 ± 0.0178	hasPool
0.4822 ± 0.0193	hasYard
0 ± 0.0000	price
0 ± 0.0000	numPrevOwners
0 ± 0.0000	numberOfRooms
0 ± 0.0000	floors
0 ± 0.0000	cityCode
0 ± 0.0000	cityPartRange
0 ± 0.0000	made
0 ± 0.0000	hasGuestRoom
0 ± 0.0000	hasStormProtector
0 ± 0.0000	basement
0 ± 0.0000	attic
0 ± 0.0000	garage
0 ± 0.0000	hasStorageRoom
0 ± 0.0000	squareMeters

Figura 3.20 Permutación de atributos clasificación binaria con red neuronal. Se obtienen resultados análogos al modelo de bosque aleatorio, ponderando a 0 los atributos no lujosos

3.3. Tarea: clasificación multiclas

3.3.1. Conjunto de datos utilizado

Una tarea de clasificación multiclas consiste en clasificar cada ejemplo en una clase seleccionada de entre más de dos clases posibles. Para la tarea de clasificación multiclas se ha escogido el conjunto de datos Clasificación de precios de móviles².

Este conjunto de datos contiene 20 características de distintos modelos de móviles y se encuentran clasificados en 4 rangos de precios (esta será la variable objetivo). Lo que se busca es a partir de estas características clasificar los móviles. No se tuvo que hacer ningún tipo de preprocesado pues los datos no contenían datos faltantes y todos eran de tipo numérico por lo que no hubo que transformarlos.

A continuación, se adjunta un ejemplo de registros de este conjunto de datos.

²<https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification>

index	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	...	price_range
0	842	0	2.2	0	1	0	7	...	1
1	1021	1	0.5	1	0	1	53	...	2
2	563	1	0.5	1	2	1	41	...	2
3	615	1	2.5	0	0	0	10	...	2
4	1821	1	1.2	0	13	1	44	...	1

Figura 3.21 Ejemplo de datos Clasificación Multiclase. Para este conjunto de datos no hubo que hacer ningún tratado de datos pues todos los datos eran numéricos. Las características presentes son relativas a propiedades de los dispositivos.

3.3.2. Modelo de caja negra: modelo de bosque aleatorio

Detalles de la implementación

En lo que se refiere a la implementación se utilizó el modelo con la configuración por defecto.

Rendimiento

Para esta tarea y este árbol se consigue un rendimiento del 100 % sobre el conjunto total de datos. Cómo la explicabilidad se va a hacer sobre todos los datos no se buscaba probar el modelo con datos nuevos.

Modelo de explicabilidad: modelo subrogado

A partir de las predicciones del modelo de bosque aleatorio se procedió a entrenar un árbol de decisión clásico. En esta ocasión debido a que si no se limitaba la profundidad del árbol este acababa con una profundidad demasiado alta como para considerar un modelo explicable, se limitó a tres. Con esta profundidad se clasificaban móviles en las cuatro categorías distintas y el rendimiento era aceptable de alrededor al 80 %.

Árbol de clasificación multiclase

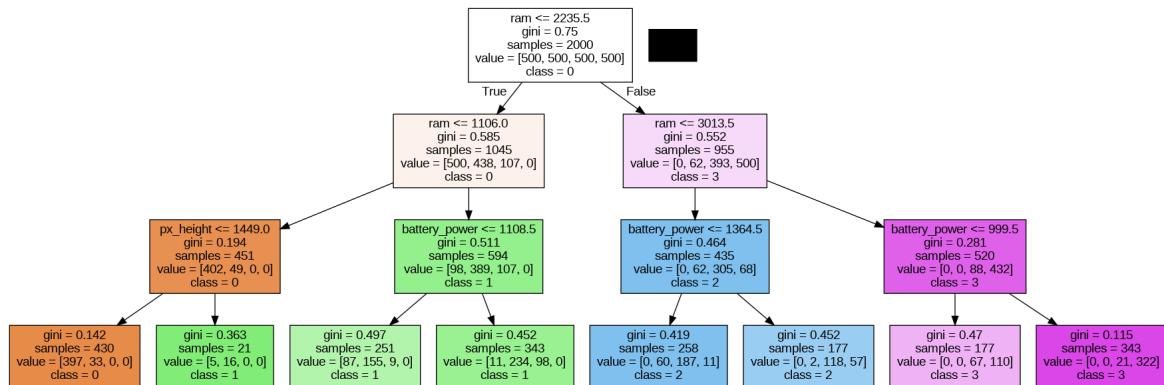


Figura 3.22 Modelo subrogado: árbol de decisión clasificación multiclas con bosque aleatorio. Se genera un diagrama del árbol, esta vez es más complejo pues se pasa de dos clases a 4. Los distintos colores indican la clase del nodo y a mayor saturación un índice de Gini más bajo. La clase con menos nodos finales será el rango 0, seguida de la clase dos y la clase 3 y por último la clase 1.

En la figura 3.22 vemos cómo los atributos que se tienen en cuenta son la memoria RAM, el tamaño de la pantalla y la capacidad de la batería. En las hojas finales se consigue en algunas ocasiones una impureza cercana al cincuenta por ciento, pero al no tratarse de una clasificación binaria no tienen un impacto tan grande sobre el rendimiento.

De este árbol de decisión podemos extraer que para que un dispositivo sea categorizado como de clase cero tendrá que tener una memoria RAM menor o igual a 1106 y la altura de pantalla ser menor o igual a 1449. Es coherente pues a mayor RAM y mayor tamaño de pantalla por lo general, se trata de móviles más caros. Serán considerados de rango uno los que superan la altura de 1449 como aquellos que tienen una memoria RAM mayor a 1106. Aquellos dispositivos que serán considerados de rango dos serán los de RAM menor o igual a 3013.5, en otro caso serán considerados de rango tres.

Se puede observar en el árbol de decisión construido que con una mayor profundidad la clasificación sería más precisa pues en muchos casos se introducen nuevos atributos como es el caso de la clasificación de los dispositivos de rango uno al introducir el atributo batería, pero al limitar la profundidad esa división no aporta nueva información.

Como última observación añadir que se colorea el árbol acorde a la clase a la que pertenece, es decir, color para cada rango y a mayor pureza más saturado.

Sistemas de Reglas

Al tratarse de una clasificación multiclas, no se puede entrenar directamente *RuleFit* como modelo subrogado, ya que este modelo solo es compatible con tareas de clasificación binaria. Es por esto que se utilizó la técnica «uno contra todos», con esta

técnica se va comparando cada categoría contra el resto de categorías de forma que las reglas que se obtienen son aquellas que se hacen uno para una categoría concreta contra el resto.

Otro apunte importante es la forma en la que se deberán interpretar las reglas debido a que aparecen con índice negativo las ponderaciones eso significa que es la opuesta de esa condición la que se tiene que cumplir para que pertenezca a una clase determinada. Esto en realidad no siempre se cumple, pero se comprueba que cuando se trata de reglas simples en las que sólo interviene un atributo tiende a hacerlo. Más adelante se comprobará como para reglas más complejas esta forma de interpretación no funciona.

Se describen a continuación los resultados obtenidos para cada rango de precio de los móviles.

Para la explicación de que un móvil tenga o no un precio de rango 0, el modelo subrogado proporciona 6 reglas con ponderaciones similares que en síntesis vienen a decir lo mismo. Recordamos que se tienen que interpretar de manera opuesta por lo que, si tomamos aproximadamente 1236 como valor, la ram tendrá que estar por debajo a dicho valor. Si miramos el árbol de decisión creado en la figura 3.22 vemos cómo en este modelo se clasifica como de ese rango si su ram es menor o igual a 1106 y la altura de su pantalla es menor o igual 1449. Esto último no se ha tenido en cuenta en el sistema subrogado de reglas.

rule	coef
25 ram > 1212.0	-0.07
21 ram > 1236.0	-0.10
20 ram > 1245.0	-0.08
22 ram > 1345.0	-0.07
23 ram > 1345.5	-0.07
24 ram > 1347.5	-0.04

Figura 3.23 Reglas para pertenecer al rango 0. Reglas con ponderación negativa, que al solo tener en cuenta un atributo se pueden interpretar al contrario. Por ejemplo, la regla 23 sería ram < 1347,5

Pasamos ahora a las reglas proporcionadas por el modelo subrogado para el rango 1 de precio. En este caso, se obtiene que la RAM debe de ser menor o igual a 2368.

Esto se cumple con lo que indica el árbol de decisión pues son reglas más generales que las que se indican en el árbol.

rule	coef
8	mobile_wt -0.00
22	ram > 2303.0 -0.18
20	ram > 2368.0 -0.30
21	ram > 2367.0 -0.17

Figura 3.24 Reglas para pertenecer al rango 1. Similar ocurre en este conjunto de reglas, se generan menos que en el ejemplo anterior.

Pasamos ahora a las reglas que clasificarán los móviles con precio de rango 2. En este caso se ha generado una única regla que indica que pertenecerán a esta categoría aquellos móviles con RAM mayor a 1952 y batería mayor a 1985 (figura 3.25). En el caso de la ram si se cumple con lo estipulado por el árbol, pero con esa restricción en la batería no tiene en cuenta los móviles con batería menor a 1364.5 que el árbol también consideraría como de rango 2. Este caso, por ejemplo, no sería correcto considerar la opuesta de la regla, en realidad lo que quiere decir la regla es que si se cumple esa condición los móviles que obtendremos serán cualesquiera menos los de rango 2.

rule	coef
8	mobile_wt -0.00
20	battery_power <= 1985.0 and ram <= 1952.0 -0.17

Figura 3.25 Reglas para pertenecer al rango 2. Esta regla generada es más compleja pues se tienen en cuenta más condiciones y debido a que cada atributo puede tomar un amplio rango de valores tampoco se puede invertir.

En el caso de los móviles con precio de rango 3, todas las reglas proporcionadas por el modelo subrogado tienen en torno a la misma ponderación. En este caso el sistema de reglas ha obtenido el patrón de que los móviles pertenecientes a esta categoría tienen RAM mayor a 2856 para la regla de mayor ponderación y en torno a 3014 para el resto, lo cual se acerca a lo indicado en el árbol de decisión. No obstante, se deja a un lado el atributo batería (figura 3.26).

rule	coef
21	ram <= 3012.0 -0.41
22	ram <= 3014.0 -0.19
23	ram <= 3013.5 -0.21
20	ram <= 2856.5 -0.35

Figura 3.26 Reglas para pertenecer al rango 3. De nuevo se generan reglas simples con ponderación negativa. Se pueden resumir en una única regla pues todas vienen a decir lo mismo variando ligeramente el valor del atributo ram.

Modelo de explicabilidad: LIME

De nuevo al utilizar LIME con más de dos clases el propio modelo crea la estructura de «uno contra todos». En las figuras que se mostrarán a continuación, veremos una comparativa con las distintas categorías.

En la figura 3.27 se muestra la explicabilidad local de un móvil que el bosque aleatorio predice que tiene un precio de rango 0. Se observa cómo en todos los casos se tienen en cuenta la RAM para predecir una clase u otra. Esto se puede intuir del árbol de decisión que se construyó en el que se encuentra este atributo en la raíz. De nuevo, a parecen atributos como el alto de la pantalla o la capacidad de la batería, pero sin duda el atributo más determinante en la clasificación es la memoria RAM. Veamos si para otras clases y otros ejemplos es este el atributo a tener en cuenta o si hay más que ayuden a la predicción.

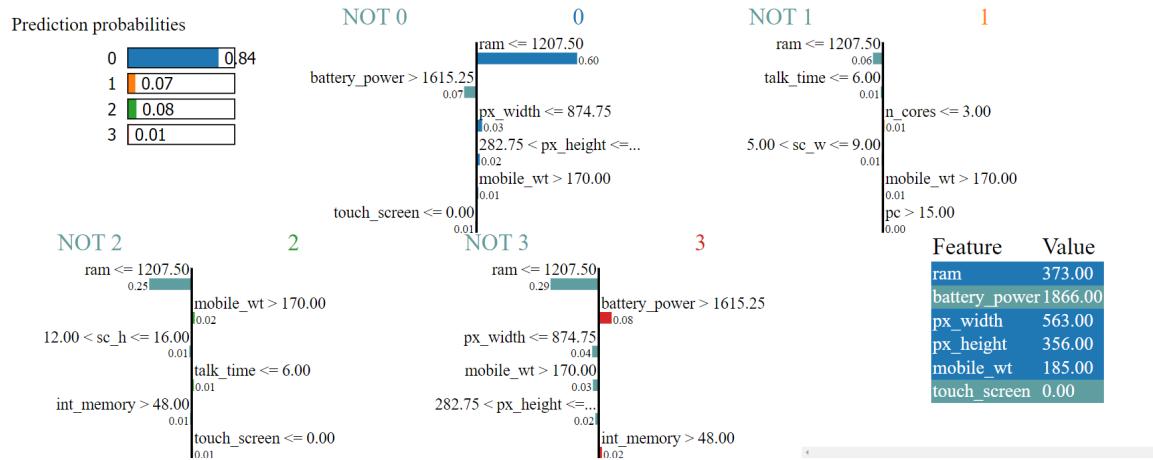


Figura 3.27 Ejemplo LIME para bosque aleatorio clasificación multiclas 1. Se genera una gráfica por cada clase a la que puede pertenecer. En este caso el predictor indica que pertenece a la clase 0 y la ram está mejor ponderada en la gráfica de esta clase. La ram es el atributo que más está ponderada en todas las gráficas.

Del mismo modo en la figura 3.28 tenemos un caso en el que la explicación aportada por LIME es compatible con la predicción del modelo.

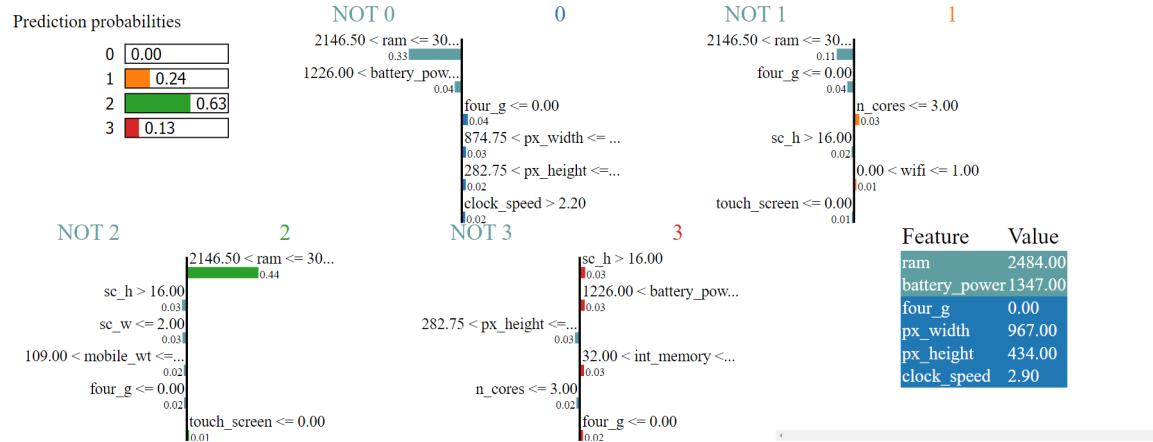


Figura 3.28 Ejemplo LIME para bosque aleatorio clasificación multiclas 2. Algo a destacar es que pese a que el predictor de una probabilidad moderada al rango 1 esto no se ve tan reflejado en la gráfica para este rango.

En la figura 3.29 la clase real es dos, los atributos que presentan tienen ponderaciones bastante altas para las otras clases, pero finalmente acaba siendo predicha como dos. Se podría esperar que se hubiese predicho la clase dos (pues esas eran las predicciones del modelo) pero según LIME lo que se esperaba es que perteneciese a la clase uno.

Con esto tenemos un ejemplo en el que LIME da una explicación que no se cumple con la realidad.

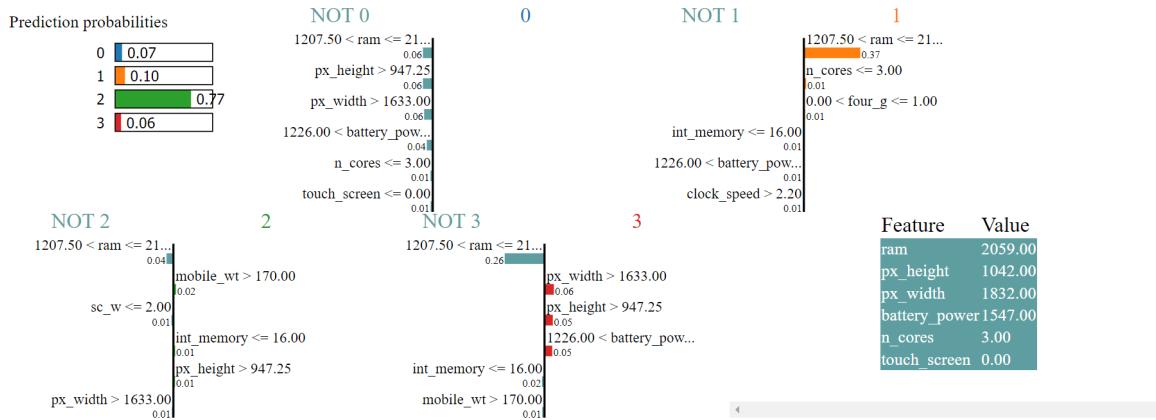


Figura 3.29 Ejemplo LIME para bosque aleatorio clasificación multiclas 3. Este ejemplo se expone para demostrar que en ocasiones LIME no encuentra ponderaciones que puedan ser acorde con lo que predice el predictor. En este caso se podría esperar que fuese la clase 1 la predicha en vez de la dos.

Se ha expuesto un ejemplo en el que LIME predice correctamente y otro para el que no lo hace. Realmente no es que LIME haga una mala predicción simplemente explica la instancia a partir de las ponderaciones de los atributos a partir del conjunto de datos proporcionados. Eso puede dar pie a que esa ponderación explique bien una instancia y otra no pues se salga de la norma. Esto también es importante tenerlo en cuenta pues cuando busquemos esa explicabilidad esta estará condicionada por la regularidad de los datos. No obstante, un caso aislado o valor anómalo siempre va a ser más difícil de justificar.

Por tanto, para este conjunto de datos y este modelo bosque aleatorio podemos afirmar que LIME no proporciona explicaciones que se adapten correctamente a las distintas predicciones. Podemos afirmar que LIME obtiene mejores resultados en conjuntos de datos en los que sea más sencillo encontrar patrones que sea regulares para las distintas clases. Esto último hace que se aleje algo de buscar la explicabilidad en modelos complejos, pues si la tarea se puede resolver como un modelo interpretable como los árboles de decisión posiblemente no haga falta usar métodos como LIME. pero cuando se complica la búsqueda del patrón y tenemos que pasar a usar modelos de caja negra, también se complica para LIME.

Modelo de explicabilidad: SHAP

El siguiente método de explicabilidad para este modelo será SHAP. En este caso pese a tratarse de una tarea de clasificación multiclas la librería que lo implementa cuenta con multitud de representaciones que hace más fácil la interpretabilidad. Es por esto que para este tipo de clasificación se analizará también el gráfico resumido que muestra las contribuciones de cada valor de cada atributo donde podremos ver cierto paralelismo con los resultados de LIME.

En primer lugar, como punto de partida antes de tratar de explicar predicciones concretas, se resumen en el gráfico de la figura 3.30 las medias de las ponderaciones de cada atributo proporcionadas por el modelo de explicabilidad local para cada rango de precios. Vemos que como era de esperar el atributo más influyente es la memoria RAM pero no para todos influye de igual forma, especialmente lo hace para la clase dos. Por otro lado, la batería es más importante para la clase tres, por ejemplo. Clases como la tres o la uno no destaca demasiado en ningún atributo en concreto.

Se repite el mismo patrón que en el árbol de decisión los atributos más significativos y por ende los mayor ponderados son la memoria RAM, la batería y la altura de la pantalla.

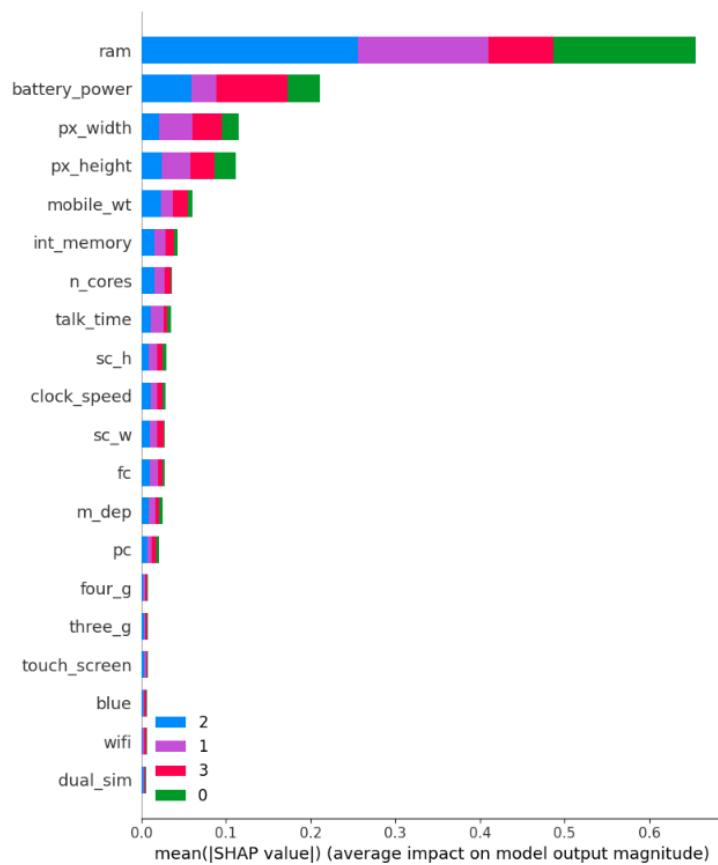


Figura 3.30 Gráfica resumen de SHAP en clasificación multiclase bosque aleatorio. Este gráfico muestra un resumen de las aportaciones de los valores *Shapley*, se encuentran ordenados de mayor a menos aportación. Además, se utiliza un código de colores para indicar la aportación de cada clase sobre el atributo en cuestión.

En la figura 3.31 se han representado 5 instancias, una para las clases dos, tres y cero y dos para otra la clase uno. Se puede observar cómo para las instancias de la clase uno las explicaciones son similares

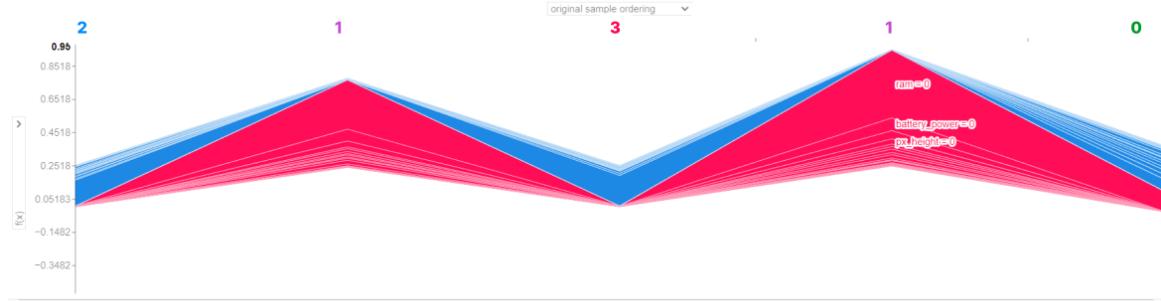


Figura 3.31 Ejemplo SHAP para bosque aleatorio clasificación multiclas 1. En el eje x se representan distintas instancias del conjunto de datos, en la parte superior se ha indicado la clase a la que pertenecen. En el eje y la aportación de cada atributo. Este gráfico no es representativo para tareas de clasificación multiclas.

Se han calculado los valores SHAP para la misma instancia que en el ejemplo 3.28, en la figura 3.32 vemos como la clase predicha (que es la dos) es la que obtiene una mayor puntuación, además se ve reflejada aquello que se veía en la explicación de LIME de que el ancho de la pantalla contribuía a ser de la clase tres, no tienen tampoco demasiada ponderación en SHAP pero es algo que este algoritmo también ha considerado como de la clase tres.



Figura 3.32 Ejemplo SHAP para bosque aleatorio clasificación multiclas 2. Se representan en cada gráfico la aportación de cada atributo para predecir esa instancia como cada uno de los rangos. Para el rango dos es para el que se obtiene un valor más próximo a 1, por tanto, SHAP lo predice como de esa clase.

En esta representación aparece de nuevo el concepto de valor base que para cada una de las representaciones de fuerza sería el valor medio de las instancias de cada clase. A este valor se le sumaría el valor de las fuerzas y así obtendríamos el valor de la predicción para cada clase. Por tanto, el mayor valor obtenido será a la clase a la que pertenecerá.

3.3.3. Modelo de explicabilidad: permutación de atributos

Lo que se podría esperar es que los tres atributos con más peso sean la memoria RAM, la batería y la altura de la pantalla. En la figura 3.33 vemos cómo se cumple lo comentado y cómo le da bastante más peso al atributo de la memoria RAM.

Weight	Feature
0.6314 ± 0.0262	ram
0.0996 ± 0.0074	battery_power
0.0311 ± 0.0076	px_width
0.0231 ± 0.0034	px_height
0.0020 ± 0.0009	mobile_wt
0.0005 ± 0.0006	m_dep
0.0005 ± 0.0000	clock_speed
0.0003 ± 0.0008	int_memory
0.0003 ± 0.0008	n_cores
0.0003 ± 0.0008	talk_time
0.0001 ± 0.0004	sc_h
0 ± 0.0000	pc
0 ± 0.0000	touch_screen
0 ± 0.0000	sc_w
0 ± 0.0000	four_g
0 ± 0.0000	fc
0 ± 0.0000	dual_sim
0 ± 0.0000	three_g
0 ± 0.0000	blue
0 ± 0.0000	wifi

Figura 3.33 Permutación de atributos bosque aleatorio clasificación multiclas. En este caso la ponderación más alta es para ram, con diferencia. El resto de atributos o tienen ponderaciones muy próximas a cero o directamente llegan a 0.

3.3.4. Modelo de explicabilidad: importancia de atributos

Por último, usando el modelo de bosque aleatorio extraeremos de él las importancias de los atributos, cuando comparamos los resultados en la tarea de clasificación binaria para las técnicas de permutación de atributos e importancia de atributos se obtuvieron resultados similares, veamos si ocurre lo mismo para la tarea de clasificación multiclas.

```
[('ram', 0.4778614162672778),
 ('battery_power', 0.07442786695601604),
 ('px_width', 0.05814008955926708),
 ('px_height', 0.056584395413913406),
 ('mobile_wt', 0.041449982885930305),
 ('int_memory', 0.03467041473399479),
 ('talk_time', 0.030495371460679613),
 ('pc', 0.029081415141312215),
 ('sc_w', 0.028651689773378248),
 ('sc_h', 0.028024848915969732),
 ('clock_speed', 0.02781914250579946),
 ('n_cores', 0.025109492121405466),
 ('m_dep', 0.02480884211472595),
 ('fc', 0.02447982030687356),
 ('wifi', 0.006991777511969236),
 ('dual_sim', 0.006865814809015037),
 ('four_g', 0.006535472405214557),
 ('touch_screen', 0.006462315070667533),
 ('blue', 0.006220653710847024),
 ('three_g', 0.0053191783357428015)]
```

Figura 3.34 Importancia de atributos bosque aleatorio. clasificación multiclas. Se representan las ponderaciones de cada atributo, de nuevo la superior ram y el resto oscilan los mismos valores.

De nuevo los valores para las dos últimas técnicas vistas son similares.

3.3.5. Modelo de caja negra: redes neuronales

Detalles de la implementación de la red

Se ha creado un primer modelo que tendrá una primera capa de normalización por lotes que acelerará el entrenamiento, posteriormente una capa densa de 10 neuronas y como entrada 17 parámetros. El primer cambio con respecto a la red neuronal de la sección anterior es la salida que se tratará de 4 neuronas con función de activación *softmax*, de modo que se devolverá la salida como una distribución de probabilidades entre las distintas clases. Compilamos el modelo usando con función de perdida la entropía cruzada categórica y como optimizador Adam. Algunos de los modelos de explicabilidad utilizados necesitan que la salida sea un valor entero y no decimal, es por esto, que tras compilar el modelo y entrenarlo, se crea un modelo se toma como entrada el modelo anterior y como salida una capa lambda con una función que redondea las salidas.

Rendimiento

Con esta implementación se consigue una precisión y un *recall* de casi el 80 % sobre el conjunto total de datos.

Modelo de explicabilidad: modelo subrogado

Árbol de clasificación binaria

De nuevo se harán las predicciones sobre el conjunto total de datos usando el modelo de red neuronal. Al construir el árbol de decisión se obtienen resultados ligeramente distintos a los obtenidos en el caso de la clasificación con el modelo de bosque aleatorio.

Los límites varían a partir del tercer nivel, en el primer caso que se tienen en cuenta la batería del dispositivo pasa de 1106 a 1466, es decir aumenta, pero la clase final predicha sigue siendo la misma. De nuevo en el tercer nivel al tener en cuenta la batería se sube el límite pasando de 1364.5 a 1701, pero se sigue clasificando en mismo rango el dos. En el último caso baja el límite de la batería, pero en este caso hay un cambio con respecto al modelo de bosque aleatorio pues de ser menor a 646.5 se considera de rango dos y en otro caso de rango tres cuando en el modelo anterior de clasificación para ambos casos era considerado de clase tres. Tiene sentido este cambio pues se ha bajado umbral de la batería, y demuestra que pese a tener más memoria RAM, si tiene menos batería puede considerarse de rango dos. Todo esto se puede observar en la figura 3.35.

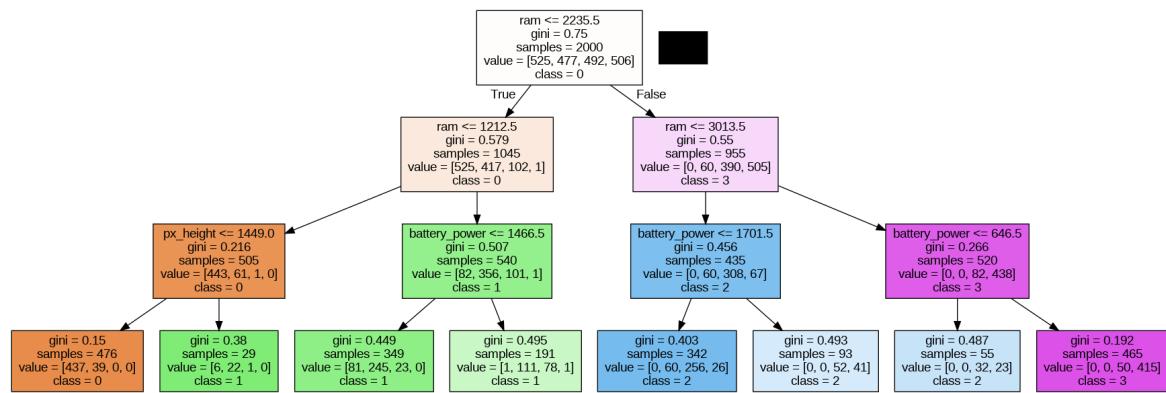


Figura 3.35 Modelo subrogado: árbol de decisión clasificación multiclas con red neuronal. El diagrama generado a partir del árbol de decisión generado varía con respecto al de bosque aleatorio. El número de nodos finales por clase ha variado acabando siendo la clase 3 y la clase 0 las que en menos nodos terminales se encuentran. La impureza es bastante alta en la mayoría de nodos menos en el nodo de categoría 0 (el primero por la izquierda) o el nodo de categoría 3 (el primero por la derecha). Esto puede ser debido a que no se encuentran tan distribuidas las instancias.

Sistemas de Reglas

Del mismo modo que con bosque aleatorio se plantearían modelos uno contra todos para poder abordar la tarea multiclas.

Se observa lo siguiente, en el caso de bosque aleatorio una de las veces que se entrenó el modelo para algunas categorías no generaba ninguna regla, pero al volver a entrenarlo ya creó reglas para todas las clases, al menos una. En el caso de las predicciones de la red neuronal se observa que siempre no genera reglas para los dispositivos pertenecientes al rango 0. En la figura 3.36 vemos como la única regla que se genera es relativa al peso del móvil, pero ni siquiera se le asigna un valor. Esto ocurrió en todas las ejecuciones del entrenamiento.

rule	coef
8	mobile_wt -0.0

Figura 3.36 Reglas para pertenecer al rango 0 para el modelo red neuronal. No se ha generado ninguna regla.

Para el resto de rangos sí se creaban reglas. Para el rango 1(figura 3.37) las reglas generadas son similares a las que se generan para bosque aleatorio, incluso las ponderaciones son similares. Esto hace que sea anómalo lo que ocurre con las reglas para el rango 0 de los datos predichos por la red neuronal.

rule	coef
8	mobile_wt -0.00
22	ram > 2303.0 -0.19
20	ram > 2368.0 -0.17
21	ram > 2367.0 -0.27
23	ram > 2371.5 -0.03

Figura 3.37 Reglas para pertenecer al rango 1 para el modelo red neuronal. De nuevo se generan reglas con ponderación negativa.

En el caso de las reglas generadas para el rango dos (figura 3.38) no se tiene en cuenta la batería, así como sí se hacía para a las reglas generadas con el modelo de bosque aleatorio y en el árbol de decisión construido a partir de las predicciones de la red neuronal. Como se comentó en la sección anterior hubo un cambio en el árbol

de modo que ahora dispositivos con menor batería a 645,5 se consideraban de la clase dos, pero si nos fijamos en el número de instancias de cada tipo en el nodo final no dista de demasiados valores, al igual que pasa en el nodo final azul de los dispositivos que cumplen que su batería es mayor a 1701,5, se acaba considerando de clase dos, pero ni la cantidad de dispositivos en ese nodo es alta ni hay una clara diferencia. Esta clasificación tan inexacta puede provenir de la limitación en profundidad que se le puso al árbol generado para así poder usarlo como modelo explicable. De todos modos, como lo que se busca en este estudio es explicabilidad y no rendimiento, es congruente lo que muestra con lo que sucede.

	rule	coef
8	mobile_wt	-0.00
20	ram <= 1953.0	-0.03
22	ram <= 1952.0	-0.06
21	ram <= 1904.5	-0.06
23	ram <= 1904.0	-0.01

Figura 3.38 Reglas para pertenecer al rango 2 para el modelo red neuronal. Reglas con ponderación negativa que en su defecto se pueden interpretar al contrario por su simplicidad.

Por último, se incluyen las reglas generadas para los dispositivos de rango tres, son similares a las que se generaron en el caso del árbol de decisión, de nuevo no se tiene en cuenta el atributo de batería para modelar las reglas.

	rule	coef
21	ram <= 3012.0	-0.18
23	ram <= 3013.0	-0.23
24	ram <= 3014.0	-0.25
25	ram <= 3013.5	-0.15
22	ram <= 3011.0	-0.01
20	ram <= 2856.5	-0.34

Figura 3.39 Reglas para pertenecer al rango 3 para el modelo red neuronal. Se muestran reglas con ponderación negativa que en síntesis indican lo mismo pues el umbral de cada regla es muy similar.

Modelo de explicabilidad: LIME

Puesto que en el modelo global ya se han observado diferencias en el comportamiento de la red neuronal con respecto a bosque aleatorio, se va a probar a ver la explicabilidad de LIME para las mismas instancias y posteriormente se analizarán otras para ver cómo se justifica esa diferencia captada en el árbol de decisión.

En la figura 3.40 vemos cómo el modelo de red neuronal es más preciso pues predice en un 95% el rango cero y en el caso de la figura 3.27 es del 84%, no obstante, la explicación de LIME es prácticamente idéntica para ambos casos pues pese a que varía el modelo de caja negra el conjunto de datos es idéntico y además no hay una gran variación en las predicciones.

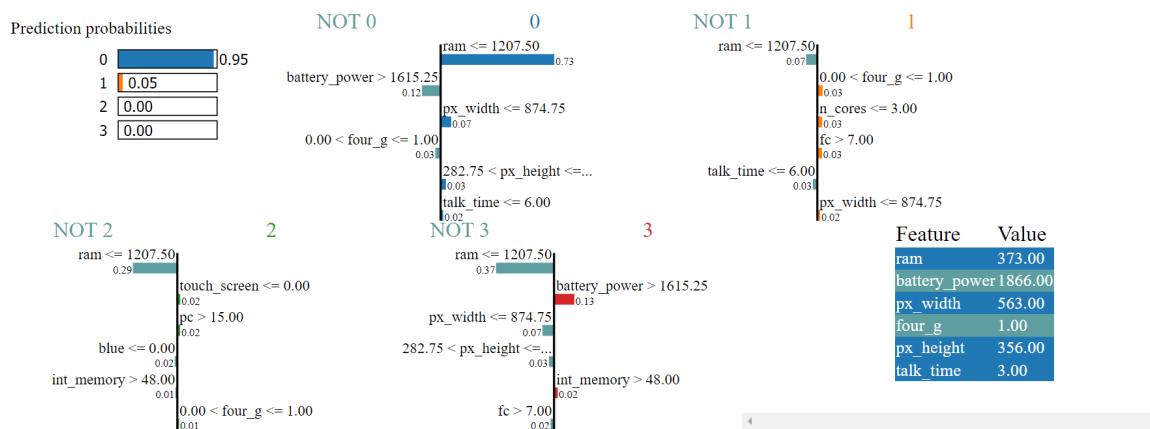


Figura 3.40 Ejemplo LIME para red neuronal clasificación multiclas 1. La variación que se pueda apreciar muy probablemente también estaría presente si volviésemos a ejecutar el modelo, por tanto, para esta instancia ambos modelos se comportan de manera similar.

Para el ejemplo expuesto en la figura 3.28 los resultados son idénticos, podemos verlo en la figura 3.41.

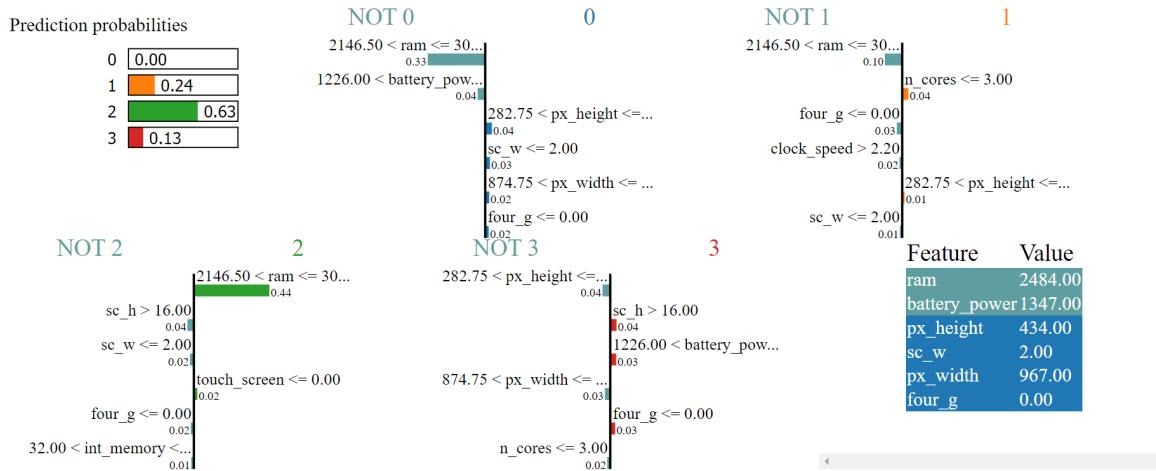


Figura 3.41 Ejemplo LIME para red neuronal clasificación multiclas 2. De nuevo un rendimiento muy similar, y además coherente la ponderación con respecto a la predicción.

Para ver cómo explica LIME el hecho de que un dispositivo considerado de rango tres pueda tener una batería asociada a rango dos vamos a buscar una instancia que tenga una RAM mayor a 3013,5 y una capacidad de la batería menor a 646,5. Se podrá comprobar si el hecho de que tenga una capacidad de la batería menor a 646,5 es considerado como algo no propio de un dispositivo de rango tres.

En la figura 3.42 se observa cómo el hecho de que la batería esté por debajo de ese umbral no es algo propio de un dispositivo de rango tres, no obstante, al tener una RAM de gran tamaño acaba considerándose de la categoría tres.

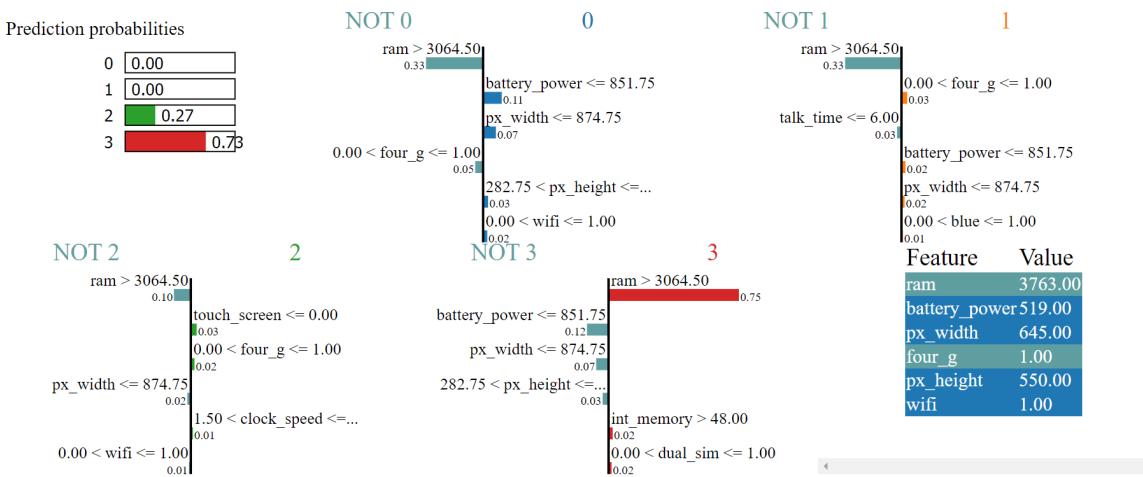


Figura 3.42 Ejemplo LIME para red neuronal clasificación multiclas 3. En este ejemplo se busca mostrar si detecta el hecho de que la batería de la instancia no sea propia del rango (según lo observado en el modelo subrogado de árbol de decisión).

LIME simula bastante bien el comportamiento que indicaba el modelo subrogado del árbol de decisión, no tiene exactamente los mismos umbrales y esto es lo que hace que en ocasiones las explicaciones aportadas no coincidan con las predicciones del modelo de caja negra.

Modelo de explicabilidad: SHAP

De nuevo, en este caso al tratarse de un modelo no basado en árboles de decisión se utilizará la variante de SHAP, kernelSHAP. Lo primero que compararemos será el esquema resumido que podemos obtener con la librería SHAP.

Para este modelo SHAP le da más peso a la memoria RAM para los dispositivos de rango tres que para el caso de bosque aleatorio, igual para el caso del atributo relacionado con la batería del dispositivo. Además, la memoria RAM deja de tener tanta importancia como era el caso en el modelo de caja negra bosque aleatorio. El resto de atributos tienen unas ponderaciones tan bajas que no se podría determinar si hay una diferencia perceptible. Ver figura 3.43.

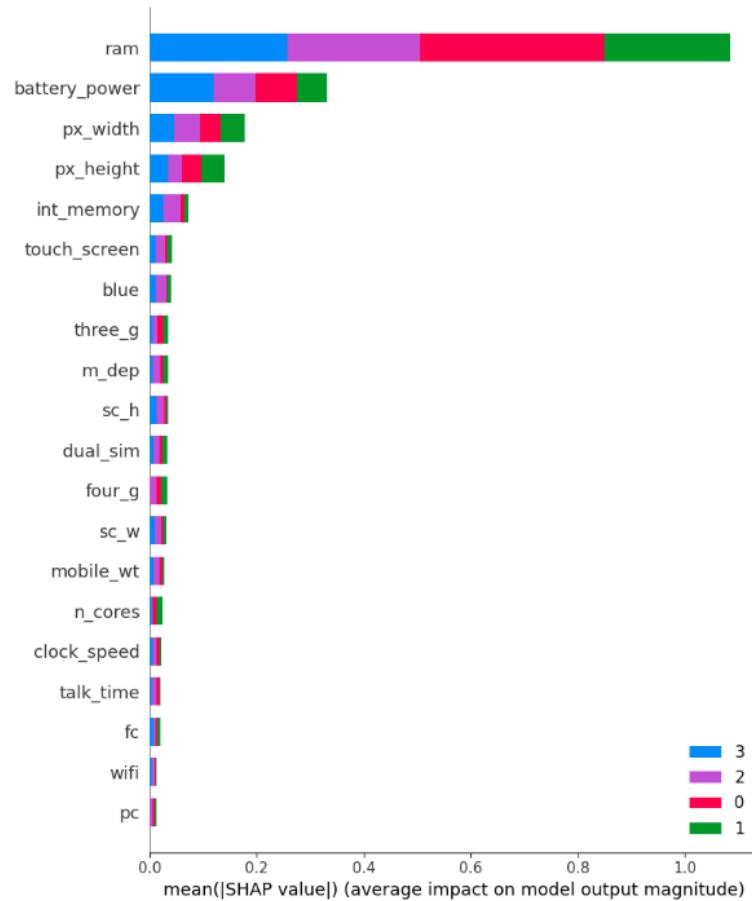


Figura 3.43 Gráfica resumen de SHAP en clasificación multiclas red neuronal. Esta gráfica es un resumen de las aportaciones de los distintos atributos a cada clase. Acompañada con una leyenda de coloreado de las distintas clases.

Para obtener una comparativa entre lo representado en la figura 3.31, al representarlo para el modelo de caja negra de red neuronal se obtienen grandes cambios lo que lleva a pensar que esas diferencias en el gráfico resumen 3.43 realmente son la prueba de que la explicabilidad de las instancias no se está haciendo de la misma forma para distintos modelos. En este tipo de figura se muestran los resultados que suman 1 y los que suman 0, esta representación no es del todo adecuada para clasificaciones multiclas, pero permite ver la importancia que se le da a los distintos atributos.

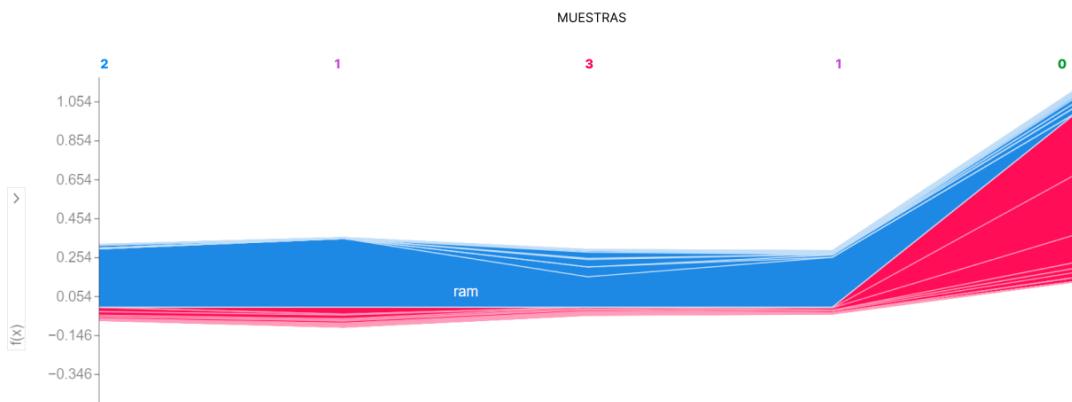


Figura 3.44 Ejemplo SHAP para red neuronal clasificación multiclas 1. En esta gráfica se muestran en el eje x las instancias seleccionadas para las que se calculará SHAP y en el eje y , las ponderaciones de sus atributos. Pese a que ya se vio en el caso del modelo bosque aleatorio que esta representación no eran posible en modelos multiclas, se quiso poner la comparativa.

Tal y como se hizo para el modelo de bosque aleatorio se quiso incluir una comparativa para la misma instancia analizada en la figura 3.32. Se observa que pese a que se ha entrenado SHAP con el mismo conjunto de datos los resultados difieren. Esto es debido a que como el modelo bosque aleatorio está basado en árboles de decisión se utilizó la variante de SHAP *TreeShap*, en cambio, para el modelo de red neuronal la variante que se utilizó fue *KernelShap*.

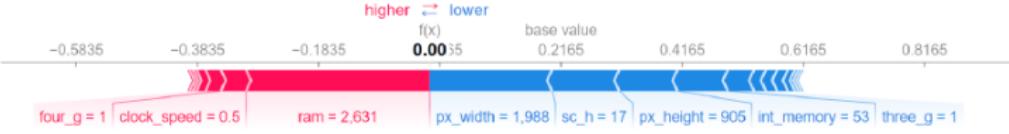
Valores SHAP para la clase 0



Valores SHAP para la clase 1



Valores SHAP para la clase 2



Valores SHAP para la clase 3

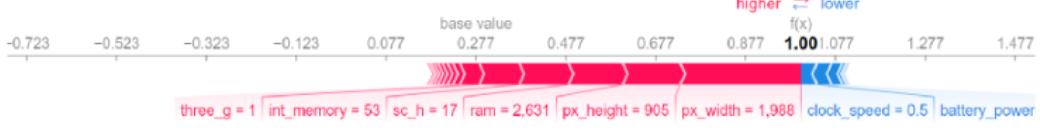


Figura 3.45 Ejemplo SHAP para red neuronal clasificación multiclas 2. SE muestran las gráficas para cada clase y las aportaciones de los valores *Shapley*. Para este caso pese a que se obtiene un valor de 1 para la clase 3 (siendo un valor más alto que cuando se calculó para en modelo de bosque aleatorio, por tanto más certero) la predicción no es correcta pues es de clase 0.

En este caso la clase predicha no sería la 2 (que es la real) sino la 3.

3.3.6. Modelo de explicabilidad: permutación de atributos

Los resultados obtenidos son casi idénticos a los de bosque aleatorio y también coinciden con los expuesto en el árbol de decisión.

Weight	Feature
0.6538 ± 0.0182	ram
0.1880 ± 0.0223	battery_power
0.0887 ± 0.0129	px_width
0.0835 ± 0.0108	px_height
0.0087 ± 0.0022	mobile_wt
0.0053 ± 0.0032	three_g
0.0034 ± 0.0028	sc_h
0.0031 ± 0.0027	four_g
0.0012 ± 0.0030	int_memory
0.0005 ± 0.0041	fc
0.0003 ± 0.0032	dual_sim
0.0002 ± 0.0024	pc
-0.0000 ± 0.0034	touch_screen
-0.0008 ± 0.0044	clock_speed
-0.0012 ± 0.0024	n_cores
-0.0013 ± 0.0082	talk_time
-0.0013 ± 0.0035	m_dep
-0.0015 ± 0.0037	sc_w
-0.0015 ± 0.0027	wifi
-0.0020 ± 0.0028	blue

Figura 3.46 Permutación de atributos red neuronal clasificación multiclas. Para este modelo de caja negra se obtienen incluso ponderaciones negativas para los últimos atributos de la lista. La ram sigue a la cabeza de la lista, así como los siguientes atributos. Esta vez se da ligeramente más ponderación a la batería.

3.4. Tarea: regresión

3.4.1. Conjunto de datos utilizado

Una tarea de regresión consiste en la predicción de una variable continua a partir de un conjunto de atributos. El conjunto de datos que se ha seleccionado es Calidad del vino. Se trata de un conjunto de datos en el que se encuentran recogidos valores para distintos compuestos químicos y su cantidad.

Este conjunto de datos contiene varios ficheros uno por cada variedad de vino, para este estudio se decide tratar con vinos tintos tan sólo. Este consta de un total de 1599 registros distintos. Se adjunta un ejemplo de algunos registros de este conjunto de datos.

index	fixed acidity	volatile acidity	citric acid	...	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.7	0.0	...	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.0	...	0.098	25.0	67.0	0.9968	3.2	0.68	9.8	5
2	7.8	0.76	0.04	...	0.092	15.0	54.0	0.997	3.26	0.65	9.8	5
3	11.2	0.28	0.56	...	0.075	17.0	60.0	0.998	3.16	0.58	9.8	6
4	7.4	0.7	0.0	...	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
5	7.4	0.66	0.0	...	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5

Figura 3.47 Ejemplo de datos Regresión. Como variable objetivo ya no hay clases marcadas si no puede tomar valores continuos del 0 al 10.

3.4.2. Modelo de caja negra: modelo de bosque aleatorio

Detalles de la implementación

En lo que se refiere a la implementación se utilizó el modelo con la configuración por defecto.

Rendimiento

Para esta tarea y este árbol se consigue un coeficiente de determinación 93,17% sobre el conjunto total de datos. Como la explicabilidad se va a hacer sobre todos los datos no se buscaba probar el modelo con datos nuevos.

Modelo de explicabilidad: modelo subrogado

Árbol de decisión

En este momento se genera el árbol de decisión a partir de las predicciones generadas por un modelo de bosque aleatorio. Las hojas finales toman valores desde 3,904 a 6,689. En este modelo los atributos que se tienen en cuenta son alcohol, sulfatos y acidez volátil principalmente (figura 3.48).

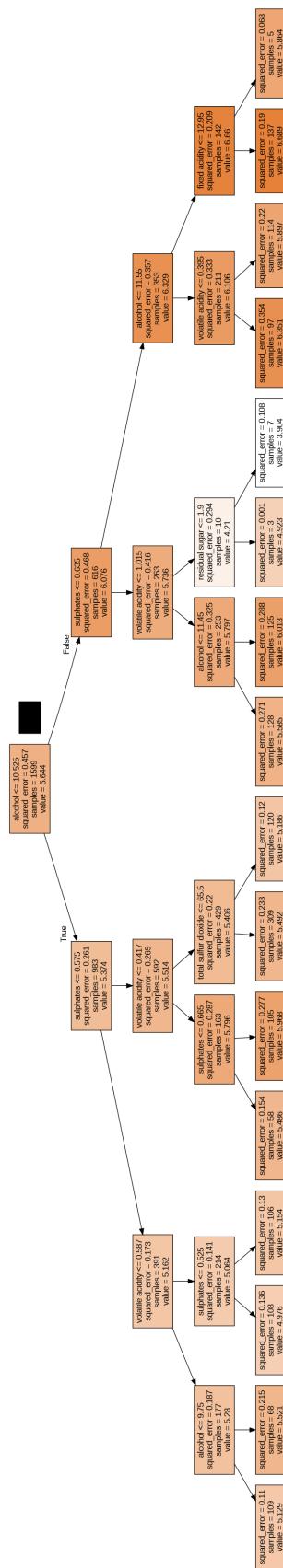


Figura 3.48 Modelo subrogado: árbol de decisión regresión con bosque aleatorio.

De nuevo se genera un diagrama del árbol construido. Los nodos con menor impureza se encuentran más saturados mientras que los de mayor menos.

A continuación, se incluye un histograma con los valores que toma la variable objetivo, calidad para los datos predichos. Los valores de calidad comienzan a aparecer desde 3,49, en mayor medida podemos encontrar valores entre 5 y 6. Las predicciones llegan hasta valores de casi 8 pero es cierto que en el árbol de decisión generado el nodo final con valor mayor alcanza un valor de 6,689 lo cual está relacionado con la limitación de la profundidad del mismo. Como podemos observar en el árbol hay un mayor número de nodos finales cercanos al rango entre 5 y 6 que a valores por debajo o por encima de éste. Esto es debido a la limitación de profundidad, el árbol ha generado más variabilidad entre las hojas finales en ese rango pues de ese modo su rendimiento será mayor. Si hay más valores en ese rango será más probable que haya menos error si se generan distintos caminos para distintos valores en el mismo de forma que la predicción esté más cerca del valor real pues no haya tanta diferencia entre el valor directamente anterior y posterior.

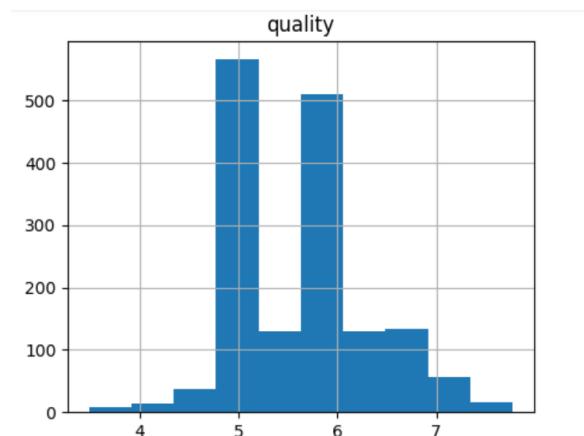


Figura 3.49 Histograma de la calidad a partir de los datos de predicción. En el eje x se representa la calidad y en el eje y se representa el número de instancias para cada calidad.

Sistema basado en reglas

Los sistemas basados en reglas no pueden utilizarse para resolver tareas de regresión sin un tratado de los datos previo. Para solucionarlo, se trata de transformar el problema de regresión en uno de clasificación. Esto se consigue dividiendo el rango de valores que puede tomar la salida en secciones de rangos más pequeños. Teniendo en cuenta como se distribuyen los datos se hará un rango para valores que vayan de 3 a 5, otro de 5 a 6, otro de 6 a 7, otro de 7 a 8 y otro de 8 en adelante. Realmente se asemeja al problema

de clasificación multiclase en el que el precio de los dispositivos estaba categorizado en rangos del 0 al 3. Una vez categorizado podemos pasar a aplicar la técnica de uno contra todos para la creación de las reglas de decisión. Antes de comenzar, añadir que la librería de generación de reglas de decisión suele introducir dos reglas que tan sólo son un atributo sin indicar límites por encima o por debajo lo cual no se puede interpretar de ninguna manera.

El primer conjunto de reglas se ha creado para el conjunto que va desde 3 a 4, en el árbol de decisión vimos cómo sólo hay un camino que lleva a un valor en ese rango que es 3,904. Con anterioridad se pensó que la forma de interpretar las reglas con ponderación negativa era interpretando lo contrario, pero debido a la casuística que hay entre los distintos valores que pueden tomar los diferentes atributos a los que hace referencia no tendría sentido. La forma de interpretar la regla es que si sucede eso es aquello que si se cumple es contraproducente para obtener un valor en ese rango, en este caso. Lo que se hará es obtener los registros que cumplen la regla.

	rule	coef
7		density -2.37
12	volatile acidity <= 0.7775 and alcohol > 9.35	-0.14
11	volatile acidity <= 0.7925 and alcohol > 8.75 and residual sugar <= 6.25 and sulphates > 0.515	-0.00

Figura 3.50 Reglas para pertenecer al rango 0. Se han generado dos reglas ambas bastante complejas por lo que no se podrán interpretar al contrario al estar ponderadas negativamente.

Se muestra a continuación, un histograma que recoge los valores de calidad de aquellas instancias que cumplen la regla 12. Puesto que se ha representado la regla tal y como venía pese a tener ponderación negativa los resultados obtenidos no son pertenecientes al rango para el que se ha creado la regla. Por tanto, estas reglas no nos permiten interpretar de ninguna forma este rango, puede que sea debido a la baja cantidad de ejemplos pertenecientes a este rango.

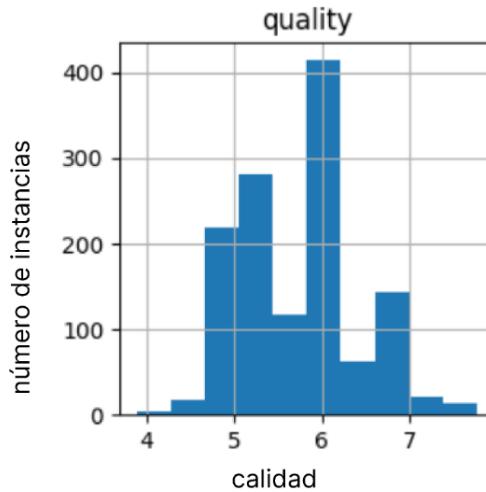


Figura 3.51 Histograma representación instancias que cumplen regla 12. Este es el histograma generado a partir de la regla 12, la mayoría de instancias son de calidades superiores a 5.

Pasemos ahora al caso en el que la calidad va desde 5 a 6, uno de los rangos con mayor número de instancias. Para este caso sí hay reglas con ponderación positiva y por ello que se intentará buscar en el árbol un camino que se ajuste a las mismas, pero debido a que no se tienen en cuenta exactamente los mismos atributos no se puede reconstruir el camino, es por esto que se decide generar el histograma correspondiente.

rule	coef
7	density 0.57
12	volatile acidity <= 0.8075 and alcohol <= 10.75 0.07
11	fixed acidity <= 11.4 and volatile acidity <= 0.985 and alcohol <= 10.75 0.07
13	volatile acidity <= 0.8625 and volatile acidity > 0.395 and alcohol <= 11.25 0.02

Figura 3.52 Reglas para pertenecer al rango 1. Las reglas generadas para este rango de calidad tienen ponderación negativa por lo que son válidas. La 12 y la 11 tienen la misma ponderación pues son muy similares.

El histograma se muestran a continuación los ejemplos que se encuentran en el mismo son en mayor medida de calidad entre 5 y 6 y ligeramente de calidad por encima de 6. Por lo que la regla estaría acotando bien.

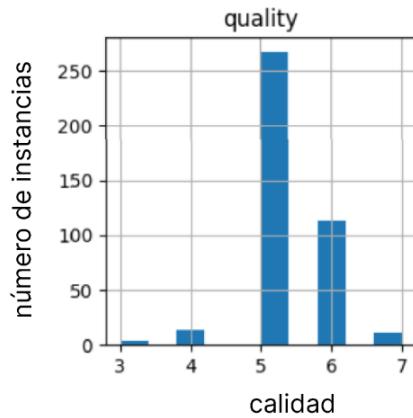


Figura 3.53 Histograma representación instancias que cumplen regla 1. En su mayoría son vinos del rango a estudiar.

Para las reglas del siguiente rango (entre 6 y 7) pasa algo similar a lo que ocurrió con las reglas del primer rango, hay tan pocas instancias que no ha sido capaz de generar reglas para estas instancias. Ver figura 3.54

	rule	coef
7	density	-1.12
13	fixed acidity <= 11.95 and volatile acidity > 0.315 and alcohol <= 11.08333	-0.02
12	alcohol <= 11.08333 and sulphates <= 0.725	-0.02
11	volatile acidity > 0.315 and alcohol <= 10.45	-0.03

Figura 3.54 Reglas para pertenecer al rango 2. De nuevo reglas con ponderaciones negativas demasiado complejas para interpretarlas al revés.

Al representar el histograma (para la regla 11) con las reglas de ponderación negativa se obtiene un histograma en el que el mayor número de instancias se distribuye para valores que no sea los del rango.

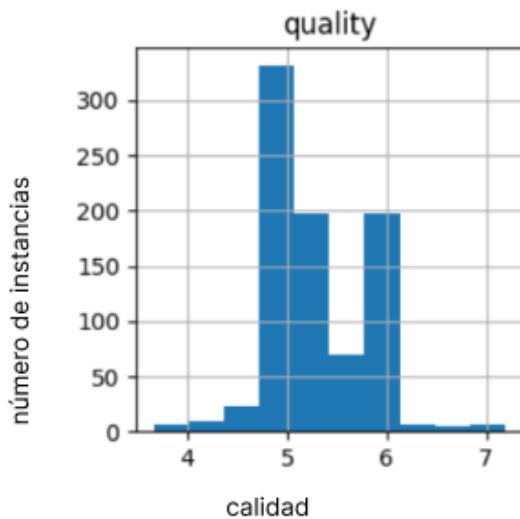


Figura 3.55 Histograma representación instancias que cumplen regla 12. En este histograma se representan las instancias que cumplen la regla 12 que en este caso serían las no pertenecientes al rango 6 – 7.

Ahora pasamos al rango que incluye los vinos con calidad entre 7 y 8. Se observa que una de las reglas está ponderada positivamente con un valor mayor a cualquier otro, se representará su histograma para comprobar si la mayoría de vinos se encuentran entre esos niveles de calidad.

	rule	coef
7		
11	volatile acidity > 0.375 and alcohol <= 12.55	-0.24
12	alcohol > 11.25 and free sulfur dioxide <= 25.5 and sulphates > 0.685	1.36

Figura 3.56 Reglas para pertenecer al rango 3. Se han generado dos reglas una con ponderación negativa y otra no sólo positiva si no con un valor muy alto, esto demuestra que el modelo de reglas de decisión ha encontrado un patrón muy marcado para este rango.

Se muestra a continuación el histograma:

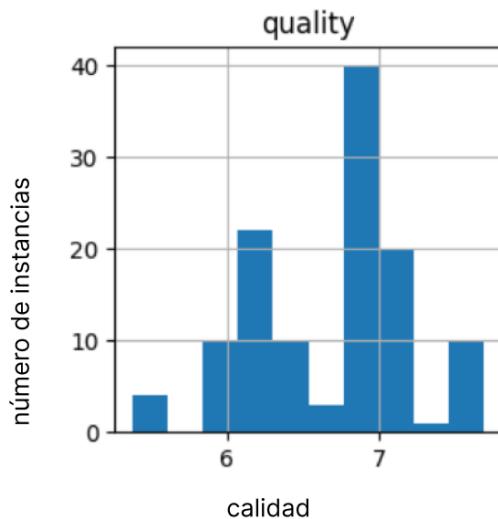


Figura 3.57 Histograma representación instancias que cumplen regla 12. Al representarlo vemos como también se considerarían instancias con valor de calidad menor a 7.

Del último rango no se generaron reglas pues no había vinos con calidad mayor o igual a 8 en el conjunto de datos predichos.

Modelo de explicabilidad: LIME

Ahora se utilizará LIME para la interpretación local. En una tarea de regresión el mínimo y el máximo valor que puede tomar es en función de las predicciones del modelo. Por ejemplo, en este primer caso (figura 3.58) nos pone que el valor mínimo es 4,23 y el valor máximo es 7,08 si recordamos los valores entre los que se encontraban las hojas finales del árbol de decisión eran 3,571 a 6,692, no tienen por qué coincidir pese a que se hiciesen con el mismo modelo de bosque aleatorio las predicciones pueden variar de una predicción a otra. Se ha estado mirando la implementación de LIME y lo que se hace es calcular las predicciones para todos los valores que se le pasan y después calcula el mínimo y el máximo. Para fijarnos en la predicción de LIME debemos mirar el texto *intercept*, para una instancia que se califica con una nota de 5 aproximadamente, se pondera negativamente un valor de alcohol por debajo de 9,5, en el árbol de decisión creado a partir de las predicciones del modelo podíamos encontrar hojas finales alrededor de 5 en ambos lados estando el atributo alcohol en la raíz. No obstante, el valor más cercano a la predicción de LIME se encontraba en el lado derecho, es decir para aquellas instancias con alcohol mayor a 11,083 esto significa, por tanto, es coherente que se considere negativo que el valor esté por debajo de 9,50 que a su vez

estarán por debajo de 10,525. Se probará a calcular la explicabilidad de otro ejemplo cercano 5 para comprobar si se sigue teniendo en cuenta de la misma forma el alcohol.

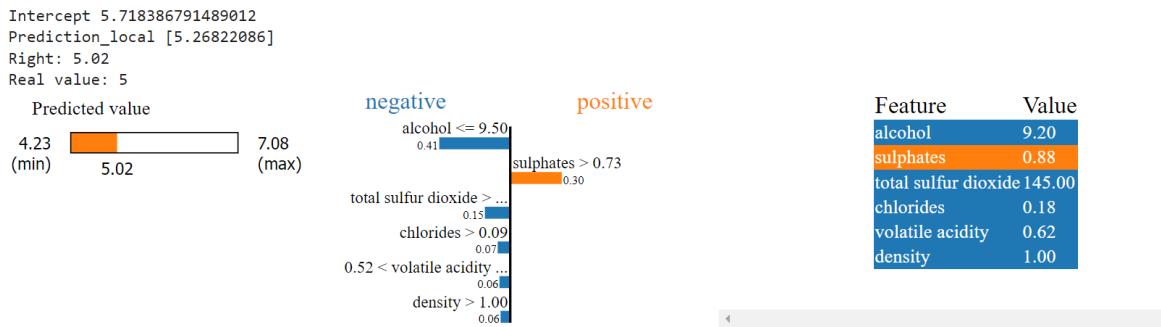


Figura 3.58 Ejemplo LIME para bosque aleatorio regresión 1. A la representación se le suma el resultado de cuatro parámetros *Intercept*, *Prediction_local*, *right* y *real value*. El gráfico central contiene los atributos que suman positivamente al valor de calidad y los que suman negativamente.

En este caso (figura 3.59) pasa algo similar, la predicción se acerca al mismo valor que antes para LIME, por lo que sigue considerando como negativo esos niveles de alcohol. Además, en atributos como *sulphates* toma un valor entre 0,55 y 0,62 según el árbol construido se espera que el atributo tome un valor menor o igual a 0,645 (siguiendo el camino que nos lleva a la hoja 5,575 que sería la predicción de LIME, aquí tenemos un caso en el que la explicabilidad global difiere de la explicabilidad local). Por otro lado, se valora positivamente el valor de *total sulfur dioxide*. Por tanto, lo que estaría haciendo LIME es calcular el valor de su predicción en base a cómo ha ponderado las instancias según la salida. Vemos cómo la explicación no es del todo acertada pues la explicación que da se aleja de la explicación global. Es cierto que no sabemos qué explicación es la más correcta pues la global parte de un árbol de decisión al que se le ha limitado la profundidad y por otro lado LIME hace explicaciones locales que además varían en distintas ejecuciones. La variación no es demasiado grande pero esas oscilaciones hacen que realmente no se pueda confiar del todo en caso como es la regresión.

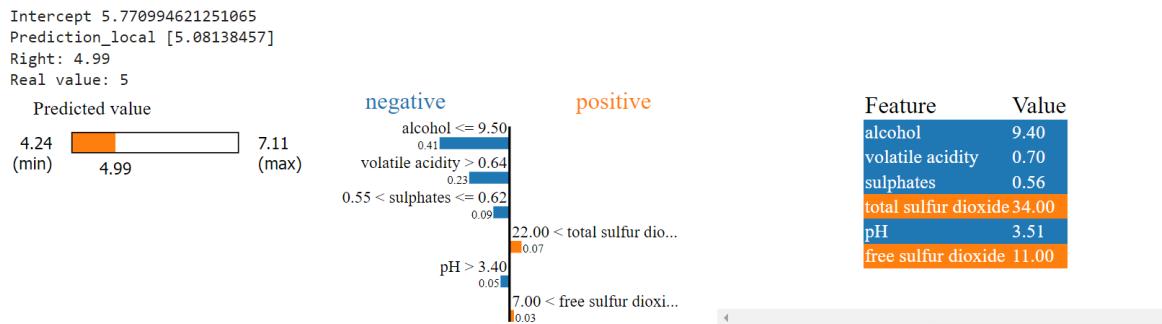


Figura 3.59 Ejemplo LIME para bosque aleatorio regresión 2. Para este caso la ponderación de la mayoría de atributos hacen que no sume.

Se prueba a continuación (figura 3.60) con una instancia que tiene una calidad de 7, de nuevo LIME, acorde a la explicabilidad que da predice un valor cercano al anterior. Además en este caso, cambia los límites superiores e inferiores del atributo alcohol por otros para que siga ponderando negativamente. En este caso, tener un valor de alcohol igual a 10 es considerado como negativo. Igualmente, en *Right* se indica el valor que da el algoritmo de predicción, que está más cercano del valor dado por LIME que del real por lo que la explicación es normal que no sea tan próxima a una explicación global.

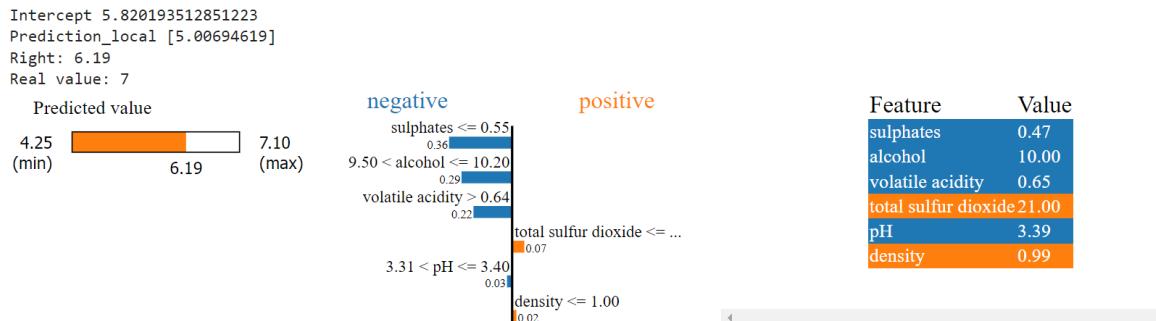


Figura 3.60 Ejemplo LIME para *Random Forest* regresión 3. De nuevo la mayoría de valores de los atributos ponderan negativamente.

Además, se comprueba que sumando los pesos de las ponderaciones de los distintos atributos más el valor interceptado de LIME (ver figura 3.61) se consigue el valor del predictor local. Los pesos vienen dados en un diccionario con dos entradas: 0 y 1. Esto es debido a que de entrada LIME está preparado para clasificaciones binarias, por ello cuando se hizo la clasificación multiclase se utilizó un modelo de uno contra todos. Tomaremos los valores que se encuentran en la parte positiva para la tarea de regresión.

```
{0: [(9, 0.36369904733214514),
      (10, 0.2919618357964289),
      (1, 0.223083328257263),
      (6, -0.07038241131680663),
      (8, 0.02949700848749578),
      (7, -0.024611487549753796)],
 1: [(9, -0.36369904733214514),
      (10, -0.2919618357964289),
      (1, -0.223083328257263),
      (6, 0.07038241131680663),
      (8, -0.02949700848749578),
      (7, 0.024611487549753796)]}
```

Figura 3.61 Pesos devueltos por LIME. Se devuelve un diccionario con dos claves uno si se toman los atributos positivamente y otro negativamente.

A estos pesos habrá que sumarle el valor calculado por el modelo, es decir, la salida de *intercept*.

Modelo de explicabilidad: SHAP

Se pasará a las explicaciones con SHAP usando la variante de *Treeshap*. En primer lugar, se hará una comparativa del gráfico de fuerzas para la misma instancia que en la figura 3.58. Lo primero destacable es que el propio modelo devuelve un valor más cercano al real que en el caso de LIME, lo segundo es que los atributos considerados como positivos en este caso los sulfatos son los que hacen que el valor se acerque al real, ya que los otros atributos estaban decrementando demasiado el valor de predicción. Como en ejemplos anteriores a mayor tamaño de flecha mayor será el impacto sobre la predicción.



Figura 3.62 Ejemplo SHAP para bosque aleatorio regresión 1. Se utiliza el gráfico individualizado pues ya no se habla de distintas categorías si no de una regresión, es similar al caso de la clasificación binaria, en vez de clasificar entre 0 y 1 es entre 0 y 10.

Veamos ahora un ejemplo en el que el valor real de la calidad del vino es de 7 (figura 3.63). Se observa que el valor de la predicción de SHAP es exactamente el que se muestra en el gráfico. Las explicaciones de SHAP se ajustan de mejor manera a los

ejemplos que las de LIME, por tanto. Pues con los modelos de explicabilidad local no se busca que la predicción de estos sea lo más cercana a la real sino lo más cercana a la predicción del modelo.



Figura 3.63 Ejemplo SHAP para bosque aleatorio regresión 2

También se ha generado el gráfico resumen donde se puede ver la contribución de cada atributo. El atributo que más contribuye es el alcohol, seguido de los sulfatos y la acidez volátil. Esto es coherente con lo visto en el árbol generado a partir de las predicciones (figura 3.48), por lo que podemos afirmar el comportamiento global del modelo es el mismo que cuando se observa una instancia concreta. Ver figura 3.64

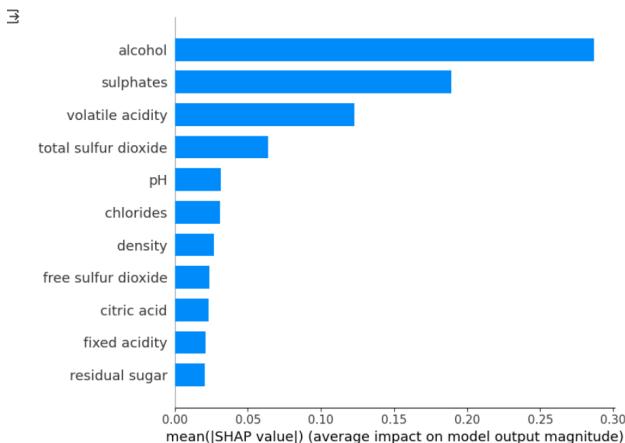


Figura 3.64 Ejemplo SHAP para bosque aleatorio regresión 3. Esta gráfica recoge el impacto de cada atributo sobre la predicción de cada instancia.

3.4.3. Modelo de explicabilidad: permutación de atributos

Pasamos al modelo de explicabilidad de permutación de atributos. Los resultados obtenidos en la gráfica resumida de la sección anterior podrían asemejarse con lo que se obtendrán a continuación.

De nuevo los atributos con mayor ponderación son alcohol, sulfatos y acidez volátil. Los que le siguen son también tenidos en cuenta en el árbol de decisión generado y en el gráfico 3.64.

Weight	Feature
0.6306 ± 0.0248	alcohol
0.4179 ± 0.0207	sulphates
0.2758 ± 0.0138	volatile acidity
0.1415 ± 0.0061	total sulfur dioxide
0.0827 ± 0.0050	chlorides
0.0760 ± 0.0020	pH
0.0663 ± 0.0055	density
0.0619 ± 0.0014	free sulfur dioxide
0.0585 ± 0.0036	citric acid
0.0582 ± 0.0033	residual sugar
0.0534 ± 0.0021	fixed acidity

Figura 3.65 Permutación de atributos bosque aleatorio regresión. Los atributos tienen todos ponderación mayor a cero, teniendo los tres primeros considerablemente mayor ponderación.

3.4.4. Modelo de explicabilidad: importancia de atributo

Por último, pasaremos a utilizar el método derivado de la librería del modelo de bosque aleatorio que permitirá obtener la importancia de atributos a partir de generar el modelo. Los resultados obtenidos son similares, hay bastante uniformidad entre las distintas técnicas de explicabilidad.

```
[('alcohol', 0.280166639182847),
 ('sulphates', 0.1391559267066379),
 ('volatile acidity', 0.13048797492778313),
 ('total sulfur dioxide', 0.07917313875780917),
 ('chlorides', 0.06219671991733558),
 ('pH', 0.05664768598264135),
 ('density', 0.05443672371700484),
 ('residual sugar', 0.05341775045772468),
 ('fixed acidity', 0.04975743821652867),
 ('free sulfur dioxide', 0.04790026811967076),
 ('citric acid', 0.04665973401401691)]
```

Figura 3.66 Importancia de atributos bosque aleatorio regresión. Los resultados se asemejan a los obtenidos en la sección de permutación de atributos.

3.4.5. Modelo de caja negra: redes neuronales

Detalles de la implementación de la red

En esta sección se detallará la arquitectura de la red. Esta red tendrá de nuevo una capa de normalización por lotes, le sigue una capa densa de 20 neuronas con entrada de

12 parámetros, otra capa densa de 10 neuronas con 12 parámetros de entrada de nuevo, por último una capa densa de 1 neurona con función de activación ReLU. Al compilar se utilizará como optimizador Adam y como función de pérdida el error cuadrático medio.

Rendimiento

Con esta implementación se consigue un error absoluto medio de 0,57. No es un rendimiento demasiado bajo, debemos tener en cuenta que los valores que se dan como salida irán desde 0 a 10, no se consigue la máxima exactitud, pero es un error asumible.

Modelo de explicabilidad: modelo subrogado

Árbol de decisión

Del mismo modo que se hizo para las otras tareas a partir de las predicciones de un modelo red neuronal se creó un árbol de decisión limitando de nuevo la profundidad de este a 4. Se intentó con una profundidad menor, pero al tratarse de una tarea de regresión en la que se limitarían los posibles resultados a tantas hojas finales como hubiera, se optó por intentar la máxima profundidad posible que siguiese permitiendo la explicabilidad. Los atributos que se han tenido en cuenta son la acidez volátil, los sulfatos, el cloruro y el dióxido de sulfuro total. Al final los valores que pueden tomar las instancias van desde 5,106 a 9,9799. En general el árbol es muy similar al generado con los datos de predicción de bosque aleatorio. Ver figura 3.67.

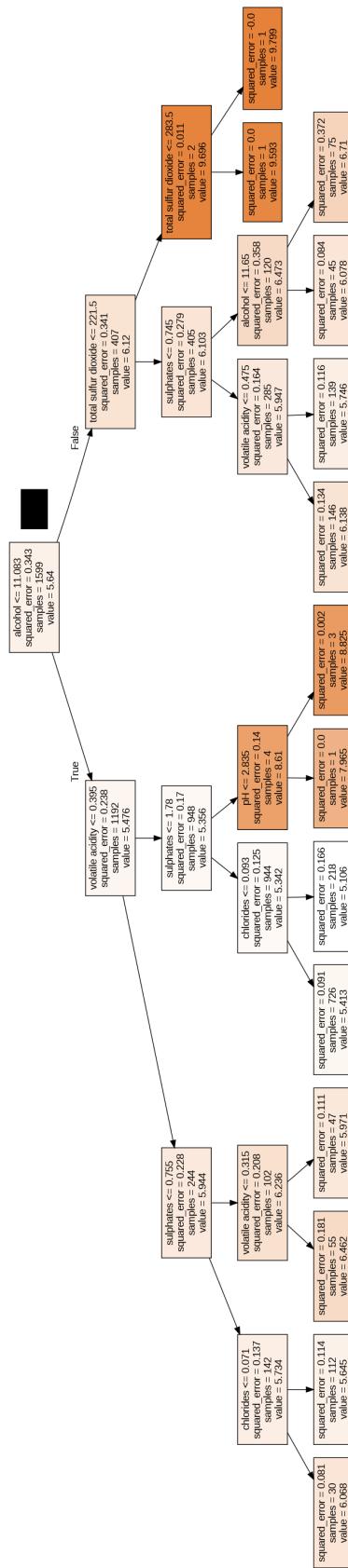


Figura 3.67 Modelo subrogado: árbol de decisión regresión con redes neuronales.

El modelo ha cambiado ligeramente con respecto al modelo generado con el bosque aleatorio. A simple vista se observan nodos menos saturados lo cual significa que la impureza de los mismos es mayor.

A continuación, se incluye un histograma con los valores que toma la variable objetivo, calidad. De los dos valores que hay menos cantidad de instancias es de 3 y 4, lo que explica que al limitar la profundidad del árbol sea para estos dos valores que no se crean nodos finales cercanos. Ver figura 3.68.

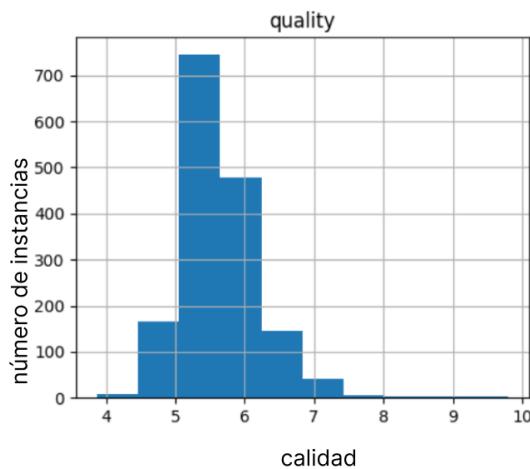


Figura 3.68 Histograma de la calidad a partir de los datos de predicción. Se observa como para las predicciones de la red neuronal se predice en mayor medida vinos de calidad entre 5 y 6 que de otras calidades. Sin embargo, en el caso del bosque aleatorio estaba mucho más repartido.

Resulta interesante que se hay creado un nodo final para valores alrededor de 9 habiendo tan pocos vinos que tengan esa calidad.

Sistema basado en reglas

Del mismo modo que se hizo en el caso de los datos de predicción de bosque aleatorio habrá que dividir el conjunto de datos en varios rangos, se seguirán creando rangos del 0 al 4 y más teniendo en cuenta que el árbol de decisión generado contempla un nodo final con valor mayor a 8.

Para el primer de rango de 3 a 5 no se ha creado ninguna regla, viendo que en el árbol de decisión generado no hay ningún nodo final (figura 3.67) que esté entre estos valores y que en el histograma 3.68 el volumen de elementos de este rango es bastante bajo se concluye que no había suficientes instancias para poder extraer un patrón. Lo único que se generó fueron un par de reglas en las que sólo se indicaba un atributo

pero sin aportar un valor por el que debería estar por encima o por abajo. Se adjuntan las reglas a continuación:

rule	coef
4	chlorides 3.90
7	density -2.68

Figura 3.69 Reglas para pertenecer al rango 0. No se han generado reglas tan sólo dos entradas que no indican umbrales de ningún tipo.

Para el siguiente rango, es decir entre 5 y 6 sí se generaron reglas y además con ponderación positiva, viendo en el histograma 3.68 la gran cantidad de vinos entre esa calidad, es coherente que se haya podido encontrar un patrón que se cumpla para el rango y no uno que se cumpla para cualquier vino que no sea del rango. Si seguimos las condiciones indicadas por la regla en el árbol de decisión subrogado llegamos a la hoja final 5,413 que está en el rango entre 5 y 6 Como algunos límites no son exactamente los mismos que los establecidos por el árbol también se podría llegar al nodo final con valor 5,106 y al nodo final 5,971, que también estarían en el rango.

rule	coef
7	density 0.69
11 fixed acidity <= 13.0 and volatile acidity > 0.385 and alcohol <= 11.925 and chlorides <= 0.1065 and sulphates <= 0.86 0.17	

Figura 3.70 Reglas para pertenecer al rango 1. Se genera una regla bastante compleja, además no está ponderada negativamente, por lo tanto, podrá utilizarse para acotar vinos de esa calidad.

Se muestra seguidamente un histograma que es generado a partir del filtrado de aquellos vinos que cumplen la regla devuelta. Se observa en la figura 3.71 que las instancias oscilan entre el límite al que pertenece por lo que la regla perfectamente recoge los vinos de esa calidad.

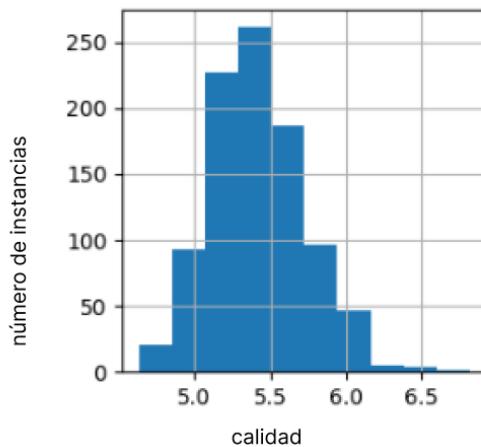


Figura 3.71 Histograma representación instancias que cumplen regla 1. Se presenta un diagrama que concentra los vinos de las calidades pertenecientes al rango.

Para el caso de del rango entre 6 y 7 de nuevo las reglas generadas tienen ponderación negativa, esto es contrario a lo que se observa en el árbol en el que hasta 5 nodos pertenecían a este rango, posiblemente el modelo *RuleFit* necesite más instancias para poder identificar patrones. Las reglas generadas se incluyen a continuación.

rule	coef
7	density -1.34
11	volatile acidity > 0.345 and alcohol <= 11.95 and citric acid <= 0.655 and sulphates <= 0.815 -0.04

Figura 3.72 Reglas para pertenecer al rango 2. Se devuelve una regla con ponderación negativa bastante compleja que no permitiría acotar instancias del rango.

Al generar el histograma acorde a la regla se puede ver que la regla identifica en su mayoría vinos con calidad no entre 6 y 7, esto indica que al buscar interpretabilidad usando este modelo debemos tener encuentro con clases desequilibradas en este caso rangos. En los datos reales que no eran los de la predicción ya había ese desequilibrio y ello se ha visto reflejado en las predicciones y por tanto en la explicabilidad.

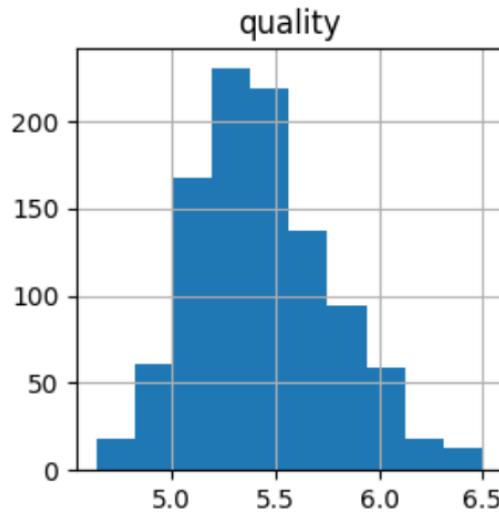


Figura 3.73 Histograma representación instancias que cumplen regla 2. El histograma generado no acota los vinos con calidad en ese rango.

Para los dos últimos rangos, de 7 a 8 y de 8 en adelante tan sólo se generan reglas con ponderación negativa, esto pasa por la poca cantidad de instancias para esta calidad. No obstante, que pasase eso en el caso de los datos de predicción de bosque aleatorio casaba más con la realidad pues en el árbol de decisión no se llegó a valores de 9 ni siquiera, en este caso hay variedad a partir de calidad 5 hasta más de 9, pero las reglas de decisión se centran entre 5 y 6.

rule	coef
7	density -3.30
11	alcohol <= 13.2 and citric acid <= 0.7 and chlorides > 0.0495 and sulphates <= 0.98 -0.19
12	alcohol <= 12.95 and citric acid <= 0.645 and density > 0.99196 and sulphates <= 0.855 -0.86

Figura 3.74 Reglas para pertenecer al rango 3. Todas las reglas generadas tienen ponderación negativa.

rule	coef
7	density -4.12
12	alcohol <= 13.8 and total sulfur dioxide <= 219.0 and density <= 1.00125 and sulphates <= 1.105 -0.68
11	alcohol <= 13.7 and residual sugar <= 8.45 and free sulfur dioxide <= 60.0 and total sulfur dioxide <= 214.5 and density > 0.99191 and sulphates <= 1.115 -0.42
13	alcohol <= 13.7 and total sulfur dioxide <= 132.0 and density <= 1.00295 and sulphates <= 1.1 -0.19

Figura 3.75 Reglas para pertenecer al rango 4. Todas las reglas generadas tienen ponderación negativa

Modelo de explicabilidad: LIME

Se quiere llevar a cabo la misma dinámica que en el apartado anterior, se expondrán los mismos ejemplos, pero para la red neuronal y se comparará lo que espera recibir el modelo con lo que predice LIME.

Se ha pasado el mismo vino que se pasó en el ejemplo 3.58.

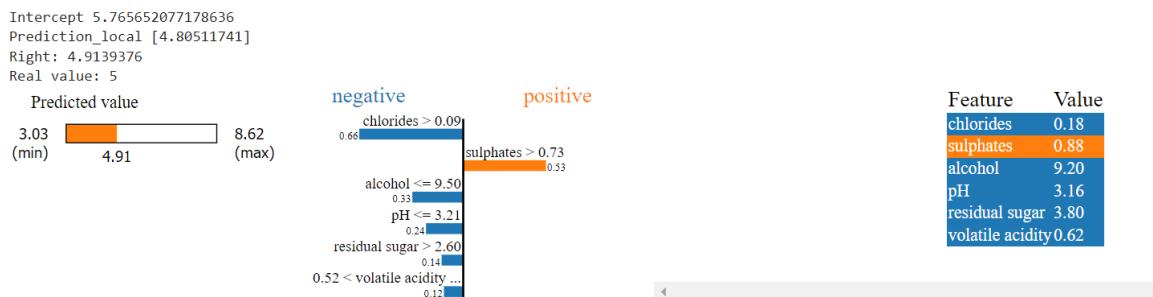


Figura 3.76 Ejemplo LIME para red neuronal regresión 1. los límites inferiores y superiores difieren bastante pese a que se está utilizando el mismo conjunto de datos de entrada. Por otro lado, el valor de predicción de LIME que es *Intercept* es muy similar, nada que no suele alejarse entre una ejecución y otra. Como ya se comprobó en el árbol de decisión subrogado el alcohol y los sulfatos toman gran importancia, pero lo que sorprende es que para el modelo de red neuronal se le da más peso a los cloruros lo que no se reflejaba en el árbol. Para esta predicción el único atributo que ponderará positivamente serán los sulfatos.

Como ya se explicó en la sección anterior el cálculo de la predicción local se hace a partir de la predicción de LIME más las ponderaciones, y si su valor es cercano a la predicción del modelo eso quiere decir que se han ponderado correctamente los atributos. Aunque la acidez volátil tenga mayor ponderación en valor absoluto pues pondera negativamente, es el pH y los cloruros los que afectan positivamente. Con LIME se obtiene un mayor valor que con la predicción local, pero como la mayoría de atributos ponderan negativamente al final el resultado local no acaba superándolo.

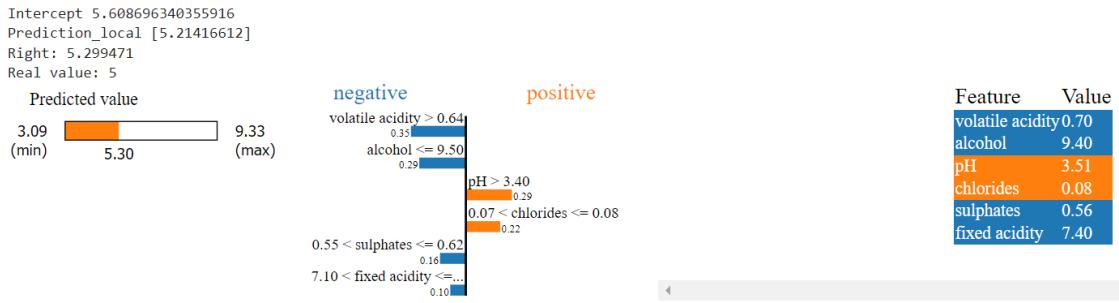


Figura 3.77 Ejemplo LIME para red neuronal regresión 2. Para el siguiente ejemplo, LIME sigue obteniendo algo similar, pero al predecir algo distinto el modelo de red neuronal la explicación difiere con respecto a la del bosque aleatorio, figura 3.59. Ahora el atributo con mayor ponderación oscila entre la acidez volátil y el alcohol dependiendo de la ejecución

Siendo esto los pesos:

```
{0: [(1, 0.34762435869958525),
      (10, 0.2948738727072011),
      (8, -0.292056145577651),
      (4, -0.21794885756619864),
      (9, 0.16356596571732127),
      (0, 0.09847102746286732)],
 1: [(1, -0.34762435869958525),
      (10, -0.2948738727072011),
      (8, 0.292056145577651),
      (4, 0.21794885756619864),
      (9, -0.16356596571732127),
      (0, -0.09847102746286732)]}
```

Figura 3.78 Pesos devueltos por LIME. Si sumamos el valor de *Intercept* a cada uno de los valores que se encuentran en la clave 1, obtenemos el valor de la predicción local.

Por último, en vez de estudiar el caso para el que el vino tiene una calidad de 7 lo haremos para un vino con calidad de 8 ya que vimos anteriormente que este modelo tenía dificultades para predecir valores fuera del rango entre 5 y 6.

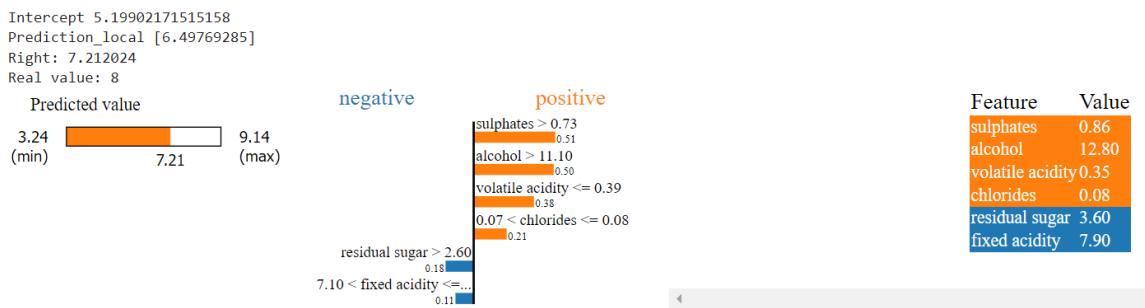


Figura 3.79 Ejemplo LIME para red neuronal regresión 3. En este caso son varios los atributos que ponderan positivamente, por ello se consigue un valor bastante alto de predicción local. No obstante, no llega al de la predicción de la red neuronal. Como hay menos vinos con valores tan altos, LIME no se adapta tan bien, pero ha sabido identificar mejor que se trataba de un vino de mayor calidad a la hora de indicar la positividad y la negatividad de los atributos.

Modelo de explicabilidad: SHAP

La variante que se utilizará será *Kernelshap* al no ser un modelo basado en árboles de decisión.



Figura 3.80 Ejemplo SHAP para red neuronal regresión 1. Se ha representado el ejemplo de 3.77, se obtiene un valor idéntico al del modelo, en este caso los sulfatos no tienen ningún peso sobre la predicción pese a ser este el único atributo con peso positivo en LIME. Los dos atributos con mayor impacto son alcohol y cloruros, similar a lo que se podría ver en el árbol de decisión subrogado

Veremos ahora un ejemplo para el que la calidad del vino es de 8.



Figura 3.81 Ejemplo SHAP para red neuronal regresión 2. Para este ejemplo si hay mayor semejanza con lo que se observó en LIME 3.79, por lo general son los mismos atributos lo que se tienen en cuenta. Tal y como ocurrió en el modelo bosque aleatorio con SHAP se consiguen resultados más fieles a la predicción del modelo que con LIME.

Por último, se incluye el gráfico de las contribuciones, difiere bastante de lo visto en el árbol de decisión.

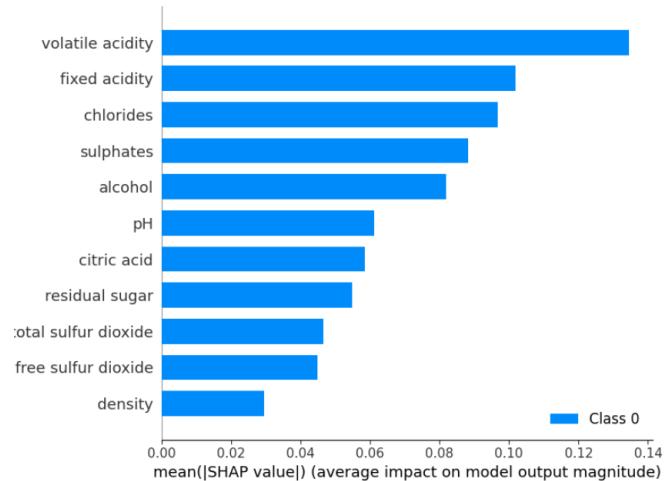


Figura 3.82 Ejemplo SHAP para red neuronal regresión 3. En este modelo todos los atributos contribuyen en mayor medida a la predicción que en el caso del modelo de bosque aleatorio. El alcohol pasa de estas en primera posición de importancia a quinta posición.

3.4.6. Modelo de explicabilidad: permutación de atributos

Se pasará a ver los resultados obtenidos en la permutación de atributos. Pese a que en orden de importancia la clasificación se encuentra muy similar a lo obtenido en el modelo de bosque aleatorio la importancia por atributo es mucho menor, del orden de la centésima.

Weight	Feature
0.0718 ± 0.0077	alcohol
0.0458 ± 0.0074	volatile acidity
0.0453 ± 0.0066	sulphates
0.0377 ± 0.0072	chlorides
0.0223 ± 0.0080	fixed acidity
0.0175 ± 0.0077	pH
0.0012 ± 0.0038	free sulfur dioxide
-0.0017 ± 0.0042	residual sugar
-0.0021 ± 0.0053	citric acid
-0.0030 ± 0.0056	total sulfur dioxide
-0.0034 ± 0.0014	density

Figura 3.83 Permutación de atributos red neuronal regresión. Para este modelo hay ponderaciones negativas en los atributos menor ponderados, además el alcohol vuelve a tener mayor importancia pese a la menor ponderación en SHAP 3.82 .

Capítulo 4

Conclusiones

El principal objetivo de este estudio es poder conocer las distintas técnicas y la forma de aplicarlas para distintas tareas y modelos. No obstante, tras el desarrollo de los distintos casos de uso se hará una vista general de los resultados en la que se destaque las técnicas que han dado explicaciones más coherentes para las distintas tareas y modelos.

La tarea de clasificación binaria parecía la más simple de abordar y para la que más técnicas de explicabilidad estaban adaptadas. Es la única para la que no hubo que adaptar ningún modelo de explicabilidad para obtener los resultados. Para ambos modelos (redes neuronales y bosques aleatorios), los resultados fueron similares. En esta tarea se ve una relación directa entre los patrones identificados en el árbol de decisión subrogado y el sistema basado en reglas. Estos mismos patrones se repetirán también en LIME, SHAP, permutación de atributos e importancia de atributos.

Al pasar a la tarea de clasificación multiclase, los árboles subrogados no presentan la impureza presente en los árboles de la tarea anterior. No obstante, esto se debe al conjunto de datos y no tanto a la tarea. Viendo estos resultados, se podía prever que los sistemas de reglas no obtendrían algo mejor. Además, la implementación de sistemas de reglas utilizada no estaba adaptada para tareas que no fueran binarias, por lo que hubo que hacer una adaptación en la implementación utilizando la técnica uno contra todos. Se esperaba que la mayoría del *software* a utilizar tuviese implementaciones para los distintos tipos de tareas que se quería abordar, pero lo cierto es que en su mayoría tan sólo eran compatibles con clasificaciones binarias.

Además, se tuvo que lidiar con ponderaciones negativas que en primera instancia no se supo interpretar. Cuando se generaban las reglas para pertenecer a la clase 1, por ejemplo, estas tenían ponderaciones negativas, por lo que en realidad clasificaban

las instancias que no pertenecían a esa clase, es decir, el resto. Esto quiere decir que esta técnica no estaría encontrando patrones para la clase especificada.

Además, en el caso de LIME, las predicciones de probabilidad del modelo no siempre coincidían con las explicaciones aportadas por LIME. Con SHAP pasa algo similar a los sistemas basados en reglas, ya que para algunas de sus representaciones no era compatible con una representación multiclas, es por esto que se obtienen los gráficos de fuerza de forma individualizada (de nuevo, «uno contra todos»). Por último, comparando los dos modelos de caja negra utilizados, las explicaciones de SHAP son más congruentes en el caso de los modelos de bosque aleatorio que en el caso de las redes neuronales.

Finalizando con la tarea de regresión, es para la que hubo que hacer una adaptación que se alejaba más del objetivo de su tarea. Tal y como hace un árbol de decisión con tareas de regresión, los posibles resultados de salida quedan resumidos en x valores discretos, lo que limita la variedad de salidas. Para el sistema basado en reglas, se dividió en rangos. Pese a la discretización de las salidas, los resultados de los sistemas subrogados capturaban mejor de lo que se esperaba los patrones de las predicciones, y tanto la implementación de LIME como SHAP daban una explicación bastante más exacta que para el resto de tareas vistas. A destacar, no creo que la forma de abordar la incompatibilidad de realizar tareas de regresión de los sistemas basados en reglas sea la más idónea, pero viendo que los árboles de decisión discretizaban la salida se quiso seguir el mismo enfoque.

Respecto a consistencia, los dos métodos que han obtenido mejores resultados para cualquiera de las tareas llevadas a cabo y modelos utilizados son la importancia de atributos derivado de bosques aleatorios y la permutación de atributos.

En lo que se refiere a lo aprendido, este trabajo de fin de máster me ha permitido acercarme a este campo de la inteligencia artificial en el que tantas esperanzas hay depositadas. De todas las técnicas que he expuesto tan sólo conocía la importancia de atributos derivada del modelo de bosque aleatorio, y sin duda ha sido muy enriquecedor conocer es qué se basa cada técnica para conferir la explicabilidad. Además me ha permitido profundizar en los modelos de caja negra que ya había utilizado para predecir, pero sin configurar los modelos con conocimiento de causa para mejorar su rendimiento. En general, este trabajo me ha permitido no sólo aprender sobre explicabilidad sino comprender el porqué de las configuraciones de hiperparámetros que también nos puede acercar a la explicabilidad de los resultados.

Como trabajo futuro, propongo abordar tareas de clasificación de imágenes y de análisis de sentimientos de textos, la biblioteca utilizada para la técnica LIME

tiene módulos para ello. Además, investigando en la documentación de SHAP se han visto aplicaciones para explicabilidad en resumen de textos, respuesta a preguntas explicabilidad para modelos de textos como los LLM.

Bibliografía

- [1] 2024. *Qué es el aprendizaje automático*. https://en.wikipedia.org/wiki/Machine_learning
- [2] AlibiExplain. 2024. *SHAP*. <https://docs.seldon.io/projects/alibi/en/latest/>
- [3] ELI5. 2021. *Permutation Importance*. https://eli5.readthedocs.io/en/latest/blackbox/permuation_importance.html
- [4] imodels. 2024. *Rule Fit*. <https://csinva.io/imodels/>
- [5] Tensorflow Keras. 2024. *Model y Sequential*. https://www.tensorflow.org/api_docs/python/tf/keras/Model
- [6] Scikit Learn. 2024. *Decision Tree*. <https://scikit-learn.org/stable/api/sklearn.tree.html>
- [7] Scikit Learn. 2024. *Random Forest Classifier*. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html#sklearn.ensemble.BaggingClassifier>
- [8] Scikit Learn. 2024. *Random Forest Regressor*. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>
- [9] LIME. 2021. *Lime tabular*. https://lime-ml.readthedocs.io/en/latest/lime.html#module-lime.lime_tabular
- [10] Christoph Molnar. 2024. *Interpretable Machine Learning*. <https://christophm.github.io/interpretable-ml-book/>
- [11] Nadia Burkart y Marco F. Huber. 2021. A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research (JAIR)* 70 (2021), 245–317.
- [12] Riccardo Guidotti y otros. 1984. A Survey of Methods for Explaining Black Box Models. *ACM Digital Library* 93 (1984), 1–42.