46–59 minutes

Proyecto SHAP

Autores:

- Mateo Alejandro Diaz Munoz - maadiazmu@unal.edu.co
- Fredy Andres Rosero Cristancho - faroseroc@unal.edu.co
- Samuel Camilo Burgos Rojas - sburgos@unal.edu.co

Fecha: November 2023

Introducción

En el contexto de inteligencia artificial, la falta de transparencia en los modelos de predicción presenta un desafío significativo. En este documento se explora la libreria de SHAP, sus aplicaciones y una pequeña explicación a cómo funciona por dentro.

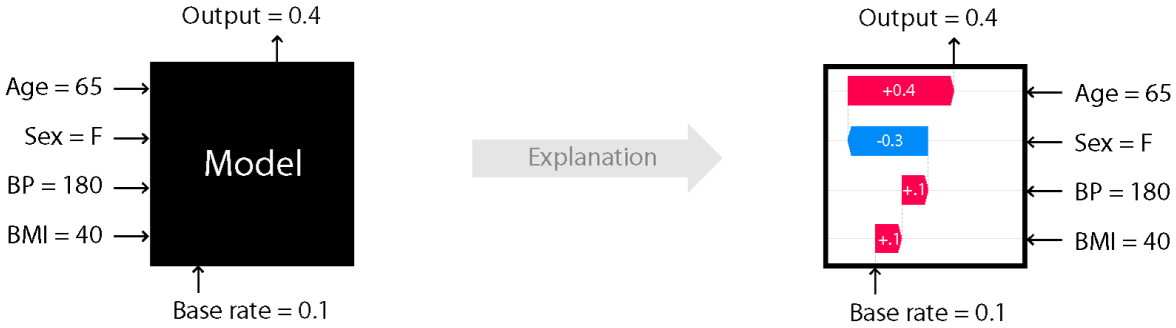
Abordaremos las matemáticas detras de los valores de Shapley con una implementación *Sympy* para hacer el cáculo paso a paso de los valores de Shapley del juego de guantes. Identificaremos los origenes de los datos de las gráficas que nos arroja SHAP con un modelo de regresión del Avaluo vivienda de california del dataset de StatLib. Finalmente haremos una aplicación para los modelos de clasificación con los conjuntos de datos de Iris y Diabetes de NCSU Statistics Department.

Identificación del problema

Los modelos de aprendizaje automático, a pesar de su alta precisión, enfrentan una barrera significativa en términos de interpretación. La dificultad para entender su funcionamiento y mejorar sus resultados se acentúa debido a la complejidad propia a estos modelos.

El problema se intensifica cuando consideramos que los modelos de inteligencia artificial suelen depender de grandes conjuntos de datos, lo que complica la identificación de las características individuales que ejercen una mayor influencia en las predicciones. La opacidad o falta de transparencia no solo afecta la comprensión del modelo, sino que también limita la capacidad para realizar mejoras específicas basadas en la relevancia de las características del modelo.

En este contexto, la falta de transparencia se convierte en un obstáculo central. Modelos como las redes neuronales y las máquinas de soporte vectorial, consideradas “cajas negras”, dificultan la comprensión de cómo cada característica de entrada contribuye a las decisiones del modelo. Esta falta de visibilidad no solo genera desconfianza en los resultados del modelo, sino que también complica la detección de posibles sesgos o problemas de equidad. Superar esta opacidad se vuelve esencial para fortalecer la confianza en los modelos y abordar cuestiones críticas de sesgo y equidad en los modelos de inteligencia artificial.



Caja negra vs. Caja blanca. Imagen tomada de shap.readthedocs.io

La librería SHAP (SHapley Additive exPlanations) en Python nos brinda una solución efectiva para abordar la opacidad en los modelos de aprendizaje automático. Proporciona una herramienta integral para la interpretación de modelos, permitiendo la asignación de contribuciones individuales de cada característica a las predicciones del modelo. Al utilizar los valores Shapley de la teoría de juegos cooperativos, SHAP facilita la comprensión de la importancia relativa de cada característica, ayudando así en la identificación de patrones, la validación del modelo y la detección de sesgo. En resumen, SHAP se presenta como una herramienta esencial para mejorar la transparencia y la interpretabilidad en el desarrollo y la aplicación de modelos de aprendizaje automático. [1]

Estado del arte

Para analizar el estado del arte utilizamos el siguiente *query string*

```
TITLE-ABS-KEY (
    ( "SHAP" AND "python" )
    AND
    ( "Shapley Additive exPlanation" OR "detection" OR "shap" OR "Explainability"
OR "Explainable" OR "exploitation" )
)
AND ( LIMIT-TO ( DOCTYPE , "ar" ) )
AND ( LIMIT-TO ( OA , "all" ) )
```

El cual nos arrojó 22 artículos consignados en SHAP.bib los cuales se analizaron con pyBibx en el notebook pyBibX_shap.ipynb de nuestro repositorio. La información a resaltar es la siguiente

Documentos

index	ID	Document
0	0	Li, Liangbo and Pu, Cheng and Jin, Nenghao and Zhu, Liang and Hu, Yanchun and Cascone, Piero and Tao, Ye and Zhang, Haizhong (2023). Prediction of 5-year overall survival of tongue cancer based machine learning. UNKNOW. doi:10.1186/s12903-023-03255-w.
1	1	Hang, Hoang Thi Thanh and Huy, Tran Trong and Phung, Nguyen Thi Kim and Uyen, Nguyen Thi Huynh (2022). IMPACT OF CAPITAL STRUCTURE ON BUSINESS PERFORMANCE OF ENERGY COMPANIES LISTED ON VIETNAM STOCK MARKET. UNKNOW. doi:10.24874/PES04.04.006.
2	2	Knitza, Johannes and Janousek, Lena and Kluge, Felix and von der Decken, Cay Benedikt and Kleinert, Stefan and Vorbrüggen, Wolfgang and Kleyer, Arnd and Simon, David and Hueber, Axel J. and Muehlensiepen, Felix and Vuillerme, Nicolas and Schett, Georg and Eskofier, Bjoern M. and Welcker, Martin and Bartz-Bazzanella, Peter (2022). Machine learning-based improvement of an online rheumatology referral and triage system. UNKNOW. doi:10.3389/fmed.2022.954056.

index	ID	Document
3	3	Scavuzzo, Carlos Matias and Scavuzzo, Juan Manuel and Campero, Micaela Natalia and Anegagrie, Melaku and Aramendia, Aranzazu Amor and Benito, Agustín and Periago, Victoria (2022). Feature importance: Opening a soil-transmitted helminth machine learning model via SHAP. UNKNOW. doi:10.1016/j.idm.2022.01.004.
4	4	Eder, Matthias and Moser, Emanuel and Holzinger, Andreas and Jean-Quartier, Claire and Jeanquartier, Fleur (2022). Interpretable Machine Learning with Brain Image and Survival Data. UNKNOW. doi:10.3390/biomedinformatics2030031.
5	5	Shi, Songchang and Pan, Xiaobin and Zhang, Lihui and Wang, Xincai and Zhuang, Yingfeng and Lin, Xingsheng and Shi, Songjing and Zheng, Jianzhang and Lin, Wei (2022). An application based on bioinformatics and machine learning for risk prediction of sepsis at first clinical presentation using transcriptomic data. UNKNOW. doi:10.3389/fgene.2022.979529.
6	6	Gashi, Milot and Vuković, Matej and Jekic, Nikolina and Thalmann, Stefan and Holzinger, Andreas and Jean-Quartier, Claire and Jeanquartier, Fleur (2022). State-of-the-Art Explainability Methods with Focus on Visual Analytics Showcased by Glioma Classification. UNKNOW. doi:10.3390/biomedinformatics2010009.
7	7	Li, Richard and Shinde, Ashwin and Liu, An and Glaser, Scott and Lyou, Yung and Yuh, Bertram and Wong, Jeffrey and Amini, Arya (2020). Machine learning–Based interpretation and visualization of nonlinear interactions in prostate cancer survival. UNKNOW. doi:10.1200/CCI.20.00002.
8	8	Kang, Danbee and Kim, Hyunsoo and Cho, Juhee and Kim, Zero and Chung, Myungjin and Lee, Jeong Eon and Nam, Seok Jin and Kim, Seok Won and Yu, Jonghan and Chae, Byung Joo and Ryu, Jai Min and Lee, Se Kyung (2023). Prediction Model for Postoperative Quality of Life Among Breast Cancer Survivors Along the Survivorship Trajectory From Pretreatment to 5 Years: Machine Learning-Based Analysis. UNKNOW. doi:10.2196/45212.
9	9	Assegie, Tsehay Admassu (2023). Evaluation of Local Interpretable Model-Agnostic Explanation and Shapley Additive Explanation for Chronic Heart Disease Detection. UNKNOW. doi:10.46604/peti.2023.10101.
10	10	Mc Cord—De Iaco, Kimberly A. and Gesualdo, Francesco and Pandolfi, Elisabetta and Croci, Ileana and Tozzi, Alberto Eugenio (2023). Machine learning clinical decision support systems for surveillance: a case study on pertussis and RSV in children. UNKNOW. doi:10.3389/fped.2023.1112074.
11	11	Talkhi, Nasrin and Nooghabi, Mehdi Jabbari and Esmaily, Habibollah and Maleki, Saba and Hajipoor, Mojtaba and Ferns, Gordon. A. and Ghayour-Mobarhan, Majid (2023). Prediction of serum anti-HSP27 antibody titers changes using a light gradient boosting machine (LightGBM) technique. UNKNOW. doi:10.1038/s41598-023-39724-z.
12	12	Amparore, Elvio and Perotti, Alan and Bajardi, Paolo (2021). To trust or not to trust an explanation: using LEAF to evaluate local linear XAI methods. UNKNOW. doi:10.7717/peerj-cs.479.
13	13	Guo, Le and Li, Xijun and Zhang, Chao and Xu, Yang and Han, Lujun and Zhang, Ling (2023). Radiomics Based on Dynamic Contrast-Enhanced Magnetic Resonance Imaging in Preoperative Differentiation of Combined Hepatocellular-Cholangiocarcinoma from Hepatocellular Carcinoma: A Multi-Center Study. UNKNOW. doi:10.2147/JHC.S406648.
14	14	Kiss, Szabolcs and Pintér, József and Molontay, Roland and Nagy, Marcell and Farkas, Nelli and Sipos, Zoltán and Fehérvári, Péter and Pecze, László and Földi, Mária and Vincze, Áron and Takács, Tamás and Czakó, László and Izbéki, Ferenc and Halász, Adrienn and Boros, Eszter and Hamvas, József and Varga, Márta and Mickevicius, Artautas and Faluhelyi, Nándor and Farkas, Orsolya and Váncsa, Szilárd and Nagy, Rita and Bunduc, Stefania and Hegyi, Péter Jenő and Márta, Katalin and Borka, Katalin and Doros, Attila and Hosszúfalusi, Nóra and Zubek, László and Erőss, Bálint and Molnár, Zsolt and Párniczky, Andrea and Hegyi, Péter and Szentesi, Andrea and Kiss, Szabolcs and Földi, Mária and Vincze, Áron and Takács, Tamás and Czakó, László and Izbéki, Ferenc and Halász, Adrienn and Boros, Eszter and Varga, Márta and Nagy, Rita and Márta, Katalin and Doros, Attila and Erőss, Bálint and Molnár, Zsolt and Párniczky, Andrea and Szentesi, Andrea and Bajor, Judit and Gódi, Szilárd and Sarlós, Patrícia and Czimmer, József and Szabó, Imre and Pár, Gabriella and Illés, Anita and Hágendorn, Roland and Németh, Balázs Csaba and Kui, Balázs and Illés, Dóra and Gajdán, László and Dunás-Varga, Veronika and Fejes, Roland and Papp, Mária and Vitális, Zsuzsanna and Novák, János and Török, Imola and Macarie, Melania and Ramírez-Maldonado, Elena and Sallinen, Ville and Galeev, Shamil and Bod, Barnabás and Ince, Ali Tüzün and Pécsi, Dániel and Varjú, Péter and Juhász, Márk Félix and Ocskay,

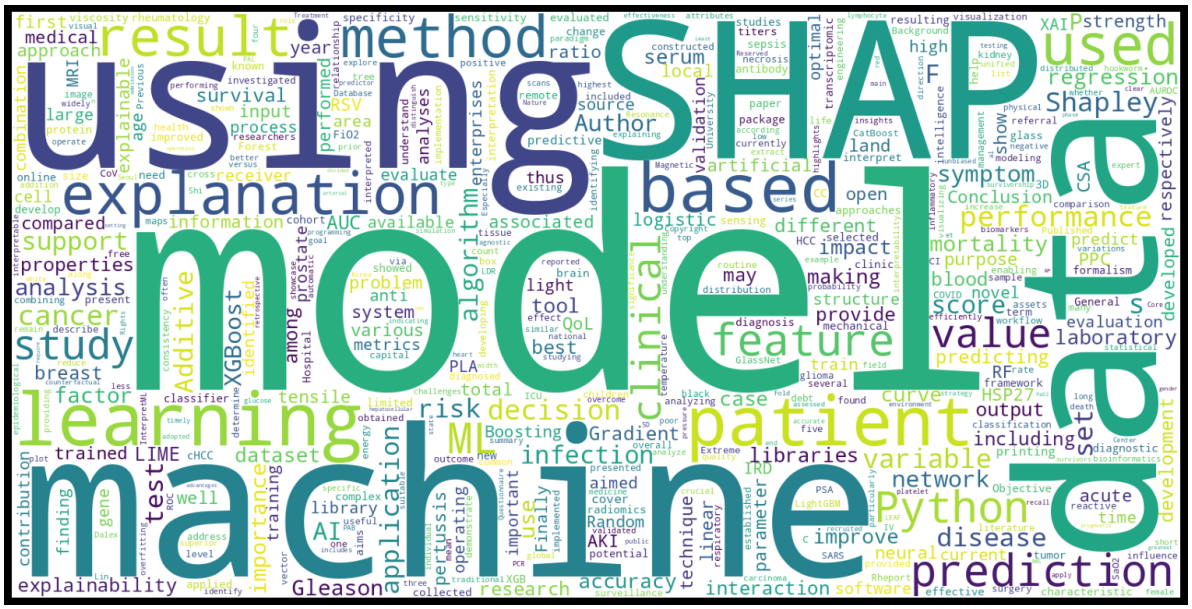
index	ID	Document
		Klementina and Mikó, Alexandra and Szakács, Zsolt (2022). Early prediction of acute necrotizing pancreatitis by artificial intelligence: a prospective cohort-analysis of 2387 cases. UNKNOW. doi:10.1038/s41598-022-11517-w.
15	15	Patel, Sharad and Singh, Gurkeerat and Zarbiv, Samson and Ghiassi, Kia and Rachoin, Jean-Sebastien (2021). Mortality Prediction Using SaO2/FiO2 Ratio Based on eICU Database Analysis. UNKNOW. doi:10.1155/2021/6672603.
16	16	Booth, Adam L. and Abels, Elizabeth and McCaffrey, Peter (2021). Development of a prognostic model for mortality in COVID-19 infection using machine learning. UNKNOW. doi:10.1038/s41379-020-00700-x.
17	17	Ege, Duygu and Sertturk, Seda and Acarkan, Berk and Ademoglu, Ahmet (2023). Machine learning models to predict the relationship between printing parameters and tensile strength of 3D Poly (lactic acid) scaffolds for tissue engineering applications. UNKNOW. doi:10.1088/2057-1976/acf581.
18	18	Shao, Jiakang and Liu, Feng and Ji, Shuaifei and Song, Chao and Ma, Yan and Shen, Ming and Sun, Yuntian and Zhu, Siming and Guo, Yilong and Liu, Bing and Wu, Yuanbin and Qin, Handai and Lai, Shengwei and Fan, Yunlong (2023). Development, External Validation, and Visualization of Machine Learning Models for Predicting Occurrence of Acute Kidney Injury after Cardiac Surgery. UNKNOW. doi:10.31083/j.rcm2408229.
19	19	Bachoc, François and Gamboa, Fabrice and Halford, Max and Loubes, Jean-Michel and Risser, Laurent (2023). Explaining machine learning models using entropic variable projection. UNKNOW. doi:10.1093/imaiai/iaad010.
20	20	Chen, Haifei and Yang, Liping and Wu, Qiusheng (2023). Enhancing Land Cover Mapping and Monitoring: An Interactive and Explainable Machine Learning Approach Using Google Earth Engine. UNKNOW. doi:10.3390/rs15184585.
21	21	Cassar, Daniel R. (2023). GlassNet: A multitask deep neural network for predicting many glass properties. UNKNOW. doi:10.1016/j.ceramint.2023.08.281.

Top 15 palaabras clave

index	ID	KWP
0	p_0	machine learning
1	p_1	human
2	p_2	article
3	p_3	support vector machine
4	p_4	male
5	p_5	female
6	p_6	controlled study
7	p_7	retrospective study
8	p_8	receiver operating characteristic
9	p_9	humans
10	p_10	artificial intelligence
11	p_11	sensitivity and specificity
12	p_12	risk assessment
13	p_13	random forest
14	p_14	prediction

index	ID	KWP
15	p_15	middle aged

Nube de palabras asociadas



Metodología

1. Contexto y definicion de los valores de shapley
2. Valores de shapley aplicados en situaciones cotidianas.
3. Analisis visual y matematico de los valores de shapley de un modelo de regresion lineal.
4. Planteamiento de casos de estudio, descripcion y analisis matematico en python.
5. Aplicación de SHAP, definicion de datasets, datos de entrada y herramientas.
6. Entrenamiento del Modelo: se aplica un tipo de modelo, en nuestros casos de estudio regresion lineal, random forest y KNN.
7. Visualización de Resultados y analisis, que describe las
8. Analisis de resultados
9.
10. Pruebas Exhaustivas

Marco teórico

Juegos cooperativos TU

Un juego se puede pensar como una secuencia de movimientos. En cada uno de ellos, uno de los jugadores decide su actuación entre distintas posibilidades. Al final del juego se asigna un pago a los jugadores que depende de las decisiones que todos ellos hayan tomado. Podemos, por tanto, describir un juego como una función que asigna un pago a cada "posición final" posible en el mismo. En el caso de los juegos **utilidad transferible** o **TU** por sus siglas en ingles, se pueden formar grupos de jugadores (que denominaremos **coaliciones**) que fuercen determinados repartos. Así pues, un juego TU puede ser definido mediante una función que asigna a cada coalición posible un número real que indica el pago asociado a dicha coalición[2].

(Un juego TU es un par donde es el conjunto finito de jugadores y es una función que cumple que .

Un juegos TU también es conocido como un **juego cooperativo** o **juego de coalición**.

Uno de los principales problemas que plantea la teoría de juegos cooperativos TU es cómo repartir la ganancia total entre

todos los jugadores de manera equitativa y acorde con la participación individual de cada jugador. Una de las técnicas más utilizadas para dar solución a este problema es el valor de Shapley[3].

Función característica

La función se llama función característica. La función tiene el siguiente significado: si es una coalición de jugadores, entonces , llamado el **valor de la coalición** , describe la suma total esperada de pagos que los miembros de pueden obtener mediante la cooperación.

(La función característica de un juego TU es una función que cumple que .

Su dominio es el conjunto de partes de , es decir, conjunto de todos los subconjuntos de , que denotamos por ó . Dada una coalición , representa el pago asegurado por los jugadores de , independientemente de las estrategias del resto de jugadores. Se denotará por el conjunto de todos los **juegos TU** con conjunto de jugadores , y por la cardinalidad de . Por simplicidad, en general identificaremos con su función característica .

Coalición

Se denomina Coalición a cualquier subconjunto no vacío de . Para cada coalición está asociado un número el cual representa el pago que se puede asegurar a los jugadores que forman parte de , independientemente de lo que hagan los demás jugadores. El **valor de una coalición** se puede considerar como la cantidad mínima que puede obtener una coalición si todos los jugadores que forman parte de ella se asocian y juegan en equipo.

Valor de Shapley

El valor de Shapley es una forma de distribuir las ganancias totales a los jugadores, suponiendo que todos colaboran. Es una distribución “justa” en el sentido de que es la única distribución con ciertas propiedades deseables que se enumeran a continuación.

Shapley, analizó durante mucho tiempo los juegos cooperativos y en 1953 propuso el concepto de valor de un juego dado para cada jugador a través de la siguiente expresión [4]:

donde es el número total de jugadores , es el numero de jugadores en la coalición y la suma se extiende sobre todos los subconjuntos de que no contienen al jugador .

También tenga en cuenta que es el coeficiente multinomial.

Aquí, los términos clave son:

- : Es el Shapley value del jugador en el juego representado por la función de valor .
- : Es el conjunto de todos los jugadores (en nuestro caso, todas las características).
- : es una coalición que no contiene al jugador [5].
- o : Representa el número de jugadores en la coalición .
- ó : Es el número total de jugadores (características).
- : Es el valor de la función cuando la coalición es ampliada para incluir al jugador .
- : Es el valor de la función cuando solo consideramos la coalición .

Muchas veces un juego TU denotado como se expresa tan solo como por practicidad. por lo que podemos tambien encontrar que seria equivalente a

Esta fórmula puede parecer compleja, pero es esencialmente una suma ponderada de las diferencias en la contribución de la característica cuando se agrega a diferentes coaliciones. El término representa la ponderación de las diferentes permutaciones posibles de las coaliciones.

Una fórmula equivalente alternativa para el valor de Shapley es:

donde la suma recorre todos los órdenes posibles de jugadores y es el conjunto de jugadores en que preceden (estan antes de) a en el orden .

Finalmente, también se puede expresar como

que se puede interpretar como

Contribución marginal

El valor de Shapley, puede interpretarse como la contribución marginal esperada del jugador o como un promedio de las contribuciones marginales de dicho jugador a todas las coaliciones no vacías , considerando que la coalición del jugador sea equiprobable en tamaño y que todas las coaliciones de tamaño tienen la misma probabilidad.

Propiedades del valor de Shapley

El valor de Shapley tiene las siguientes propiedades deseables:

1. Eficiencia: La ganancia total se distribuye:

2. Simetría: si i y j son dos actores que son equivalentes en el sentido de que: para cada subconjunto de que no contiene ni , entonces .

3. Linealidad: Si dos juegos cooperativos descritos por las funciones de ganancia y son combinados, entonces la ganancia distribuida debería corresponder a la ganancia derivada de y :

por cada en .

También, por cada número real :

por cada en .

4. Zero Player (Jugador Nulo): El valor de Shapley de un jugador nulo en un juego v es cero. Un jugador es nulo en v si if para todas las coaliciones .

De hecho, dado un conjunto de jugadores, el valor de Shapley es el único mapa a partir del conjunto de todos los juegos de vectores de ganancias que satisface todas las cuatro propiedades aquí mencionadas.

Dividendos Harsanyi

La fórmula del valor de Shapley utiliza los Dividendos Harsanyi para calcular la contribución promedio de cada jugador a la sinergia de todas las coaliciones de las que es miembro.

John Charles Harsanyi, en colaboración con John Forbes Nash and Reinhard Selten, propusieron en 1959 [6] la formula alternativa:

donde son los llamados **dividendos Harsanyi** el cual se define como[7]:

El dividendo Harasanyi identifica el excedente que se crea por una coalición de jugadores en un juego cooperativo. Para especificar este superávit, el valor de esta coalición se corrige por el superávit que ya crean las subcoaliciones.

Una formula explicita para es:

Podemos recuperar de con la ayuda de la formula

(En otras palabras, el "valor total" de la coalición proviene de sumar las *divisiones* de cada subconjunto posible de .

Nota: Los dividendos de Harsanyi pueden encontrarse en algunas literaturas como **sinergia**.

Casos de estudio

Para este proyecto se realizaron cuatro casos de estudio, dos de ellos que se analizan teoricamente y analiticamente, los cuales son el avaluo de vivienda en california y el juego de los guantes, por otro lado se tienen dos conjuntos de datos que se analizaran directamente con la herramienta shap los cuales son Iris y diabetes, dos datasets perfectos para modelarse y analizarlos como modelos de caja negra para ponerlos en la herramienta shap.

Explicación de cálculo de valores de Shapley con el ejemplo del juego de los guantes

El desarrollo del ejemplo fue tomado de artículo de Wikipedia de *Shapley value* para comprobar no solo el resultado sino su desarrollo. La implementación se realizó en el notebook juego-guantes.ipynb de nuestro repositorio.

Dependencias

Las dependencias mas destacadas para este desarollo fueron *Sympy* y *IPython*.

```
import sys
import math
import sympy as sy
from IPython.display import Math, Latex, Markdown, HTML, display
```

Ejemplo de juego de los guantes

```
# Ejemlpo del juego de los guantes
"""_summary_
El juego de los guantes es un juego cooperativo en el que los jugadores tienen
guantes de la
mano izquierda y de la mano derecha y el objetivo es formar parejas
"""
N_sym, n_sym, S_sym, Sp_sym, s_sym, i_sym = sy.symbols("N n S S' s i")
v_sym = sy.Function('v')
# Crear jugadores del juego
n = 3
N = sy.FiniteSet('A', 'B', 'C')
display(Markdown(f'Tenemos $|N|=n={n}$ jugadores: $N={sy.latex(N)}$'))
# Crear funcion caracteristica
v = {
  sy.FiniteSet('A', 'C'): 1,
  sy.FiniteSet('B', 'C'): 1,
  sy.FiniteSet('A', 'B', 'C'): 1
}

display(Markdown(f'Tenemos la funcion caracteristica es: $v={sy.latex(v)}$'))
display(Markdown(f'Si $i={i}$, entonces, $N \\setminus \\{{ {i} \\}} =
{sy.latex(N_sin_i)}$'))
i = 'A'
N_sin_i = N - sy.FiniteSet(i)
display(Markdown(f'Las coaliciones $S \\subseteq N \\setminus \\{{ {i} \\}}$
son:'))
R = sy.PowerSet(N_sin_i).rewrite(sy.FiniteSet)
for S in R:
  display(S)
```

Tenemos $|N| = n = 3$ jugadores: $N = \{A, B, C\}$

Tenemos la funcion caracteristica es: $v = \{\{A, C\} : 1,\ \{B, C\} : 1,\ \{A, B, C\} : 1\}$

Si $i = A$, entonces, $N \setminus \{A\} = \{B, C\}$

Las coaliciones $S \subseteq N \setminus \{A\}$ son:

$\emptyset$

$\{B\}$

$\{C\}$

$\{B, C\}$

Fórmula

`phi_sym` representa la fórmula simbólica para calcular la contribución individual de un jugador en el juego. La fórmula sigue la definición matemática de los valores de Shapley expueta en el marco teórico, donde se considera la combinación de características de conjuntos y la diferencia en la función de valor (`v_sym`) entre el conjunto ampliado (`Sp_sym`) y el conjunto original (`S_sym`).

```
phi_sym = (sy.factorial(s_sym) * sy.factorial(n_sym-s_sym-1)) /
sy.factorial(n_sym) * (v_sym(Sp_sym) - v_sym(S_sym))
display(phi_sym)
```

$$\frac{\left(-v(S) + v(S')\right) s! \left(n - s - 1\right)!}{n!}$$

La función `valor_shapley` toma como entrada un conjunto de jugadores `N` y una función de valor `v` que asigna un valor a cada conjunto de jugadores. Luego, para cada jugador `i` en `N`, calcula el valor de Shapley utilizando la fórmula definida anteriormente. La función itera sobre todos los subconjuntos posibles de jugadores, calcula las combinaciones y las contribuciones, y acumula el resultado en `phi`. Además, se muestra una visualización detallada de cada paso del

cálculo para mayor claridad.

```
def valor_shapley(N,v):
  for i in N:
    display(Markdown(f"Para $i={i}$"))
    n = len(N)
    phi = 0
    N_sin_i = N - sy.FiniteSet(i)
    R = sy.PowerSet(N_sin_i).rewrite(sy.FiniteSet)
    for S in R:
      s = len(S)
      combinations = (sy.factorial(s) * sy.factorial(n-s-1)) / sy.factorial(n)
      Sp = sy.FiniteSet(i).union(S)
      CM = (v.get(Sp, 0) - v.get(S, 0))
      termino = combinations * CM
      phi += termino
      display(Markdown(f"* con $S={sy.latex(S)}$ y $S'={sy.latex(Sp)}$ tenemos
${sy.latex(phi_sym)} = {sy.latex(phi_sym.subs({s_sym: s, n_sym: n }))} =
{sy.latex(combinations)} \\cdot {sy.latex(CM)}$ "))
    display(Markdown(f'El valor de Shapley para el jugador $i={i}$ es: {phi}'))
```

Ejecución

```
valor_shapley(N,v)
```

Para $i = A$

- con $S = \emptyset$ y $S' = \{A\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{3} + \frac{v\left(S'\right)}{3} = \frac{1}{3} \cdot 0$
- con $S = \{B\}$ y $S' = \{A, B\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{6} + \frac{v\left(S'\right)}{6} = \frac{1}{6} \cdot 0$
- con $S = \{C\}$ y $S' = \{A, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{6} + \frac{v\left(S'\right)}{6} = \frac{1}{6} \cdot 1$
- con $S = \{B, C\}$ y $S' = \{A, B, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{3} + \frac{v\left(S'\right)}{3} = \frac{1}{3} \cdot 0$

El valor de Shapley para el jugador $i = A$ es: 1/6

Para $i = B$

- con $S = \emptyset$ y $S' = \{B\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{3} + \frac{v\left(S'\right)}{3} = \frac{1}{3} \cdot 0$
- con $S = \{A\}$ y $S' = \{A, B\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{6} + \frac{v\left(S'\right)}{6} = \frac{1}{6} \cdot 0$
- con $S = \{C\}$ y $S' = \{B, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{6} + \frac{v\left(S'\right)}{6} = \frac{1}{6} \cdot 1$
- con $S = \{A, C\}$ y $S' = \{A, B, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{3} + \frac{v\left(S'\right)}{3} = \frac{1}{3} \cdot 0$

El valor de Shapley para el jugador $i = B$ es: 1/6

Para $i = C$

- con $S = \emptyset$ y $S' = \{C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{3} + \frac{v\left(S'\right)}{3} = \frac{1}{3} \cdot 0$
- con $S = \{A\}$ y $S' = \{A, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{6} + \frac{v\left(S'\right)}{6} = \frac{1}{6} \cdot 1$
- con $S = \{B\}$ y $S' = \{B, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{6} + \frac{v\left(S'\right)}{6} = \frac{1}{6} \cdot 1$
- con $S = \{A, B\}$ y $S' = \{A, B, C\}$ tenemos $\frac{\left(-v(S)+v\left(S'\right)\right)s!(n-s-1)!}{n!} = -\frac{v(S)}{3} + \frac{v\left(S'\right)}{3} = \frac{1}{3} \cdot 1$

El valor de Shapley para el jugador $i = C$ es: 2/3

Gráficas SHAP en el modelo de regresión lineal de Avaluo de Vivienda de California

Antes de utilizar los valores de Shapley para explicar modelos complicados, es útil entender cómo funcionan para modelos simples. Uno de los tipos de modelos más simples es la regresión lineal estándar, y a continuación entrenamos un modelo de regresión lineal en el conjunto de datos de viviendas de California.

Esta implementación se encuentra en los botebooks del repositorio del proyecto[8]

Conjunto de Datos de Viviendas en California

Este conjunto de datos consta de 20,640 bloques de casas en California en 1990, donde nuestro objetivo es predecir el logaritmo natural del precio medio de la vivienda a partir de 8 características diferentes:

1. `MedInc` - ingreso medio en el grupo de bloques
2. `HouseAge` - edad media de la vivienda en el grupo de bloques
3. `AveRooms` - número medio de habitaciones por hogar
4. `AveBedrms` - número medio de dormitorios por hogar
5. `Población` - población del grupo de bloques
6. `AveOccup` - número medio de miembros por hogar
7. `Latitud` - latitud del grupo de bloques
8. `Longitud` - longitud del grupo de bloques

> Datos de Cartas S&P
>
> Recopilamos información sobre las variables utilizando todos los grupos de bloques en California del Censo de 1990. En esta muestra, un grupo de bloques incluye en promedio a 1425.5 individuos que viven en un área geográficamente compacta.

- Naturalmente, el área geográfica incluida varía inversamente con la densidad de población.
- Calculamos distancias entre los centroides de cada grupo de bloques, medidas en latitud y longitud.
- Excluimos todos los grupos de bloques que informaron cero entradas para las variables independientes y dependientes.
- Los datos finales contenían 20,640 observaciones sobre 9 variables.
- La variable dependiente es el .

El archivo contiene todas las variables. Específicamente, incluye el valor medio de la vivienda, ingreso medio, edad media de la vivienda, total de habitaciones, total de dormitorios, población, hogares, latitud y longitud, en ese orden. [9]

Dependencias

```
from IPython.display import display, Math, Latex
import numpy as np
import pandas as pd
import seaborn as sns
```

Conjunto de Datos

```
import shap
X, y = shap.datasets.california(n_points=1000)
```

`X` (Matriz de características): Es una matriz que contiene información sobre las características o variables independientes del conjunto de datos. Cada fila de la matriz representa una observación, y cada columna representa una característica específica. En el contexto del conjunto de datos de California, las características podrían incluir información como ingreso medio, edad de la vivienda, número medio de habitaciones por hogar, etc.

```
X
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
14740	4.1518	22.0	5.663073	1.075472	1551.0	4.180593	32.58	-117.05
10101	5.7796	32.0	6.107226	0.927739	1296.0	3.020979	33.92	-117.97
20566	4.3487	29.0	5.930712	1.026217	1554.0	2.910112	38.65	-121.84
2670	2.4511	37.0	4.992958	1.316901	390.0	2.746479	33.20	-115.60

2670	2.4511	37.0	4.992958	1.316901	390.0	2.746479	35.20	-119.00
15709	5.0049	25.0	4.319261	1.039578	649.0	1.712401	37.79	-122.43
...	...	...	...	...	...	...	...	...
13339	5.7530	14.0	6.071023	0.980114	1151.0	3.269886	34.03	-117.66
2791	1.8325	25.0	4.279221	1.070130	1477.0	1.918182	37.37	-118.39
1550	6.8806	16.0	8.273632	1.042289	1272.0	3.164179	37.74	-121.93
17652	3.8936	13.0	4.128079	1.160099	710.0	1.748768	37.26	-121.88
17474	5.5184	25.0	6.030303	1.017316	1669.0	3.612554	34.44	-119.89

1000 rows × 8 columns

Modelo de regresión lineal simple

En modelos de regresión, especialmente cuando se trabaja con valores que abarcan un rango amplio, como los precios de las viviendas, es común aplicar transformaciones a la variable de respuesta para abordar problemas de sesgo o heterocedasticidad. En este caso, se está utilizando el **logaritmo natural** () del precio medio de la vivienda como la variable objetivo.

La transformación logarítmica se elige por varias razones:

1. **Estabilización de la Varianza:** La transformación logarítmica puede ayudar a estabilizar la varianza cuando esta aumenta con el nivel de la variable.
2. **Manejo de Sesgo:** Si la distribución de la variable objetivo está sesgada, la transformación logarítmica puede ayudar a reducir ese sesgo.
3. **Interpretación más Lineal:** La relación entre las variables predictoras y la variable objetivo puede volverse más lineal después de aplicar una transformación logarítmica.

```
import sklearn
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(X, y,
test_size=0.2, random_state=0)
X100 = shap.utils.sample(X_train, 100)
model = sklearn.linear_model.LinearRegression()
model.fit(X_train, y_train)
```

Examinando los coeficientes del modelo

La forma más común de comprender un modelo lineal es examinar los coeficientes aprendidos para cada característica. Estos coeficientes nos indican cuánto cambia la salida del modelo cuando cambiamos cada una de las características de entrada:

```
print("Coeficientes del modelo:\\n")
for i in range(X_train.shape[1]):
    print(X_train.columns[i], "=", model.coef_[i].round(5))
```

```
Coeficientes del modelo:\n
MedInc = 0.41027
HouseAge = 0.00903
AveRooms = -0.09932
AveBedrms = 0.6323
Population = 3e-05
AveOccup = -0.23056
Latitude = -0.49566
Longitude = -0.49432
```

Estos coeficientes indican cómo cada característica contribuye a la predicción del precio de la vivienda en el modelo de regresión lineal. Un coeficiente positivo sugiere una asociación positiva, mientras que un coeficiente negativo sugiere una asociación negativa. Es importante considerar la magnitud de los coeficientes para evaluar la fuerza de la influencia de cada característica en la predicción.

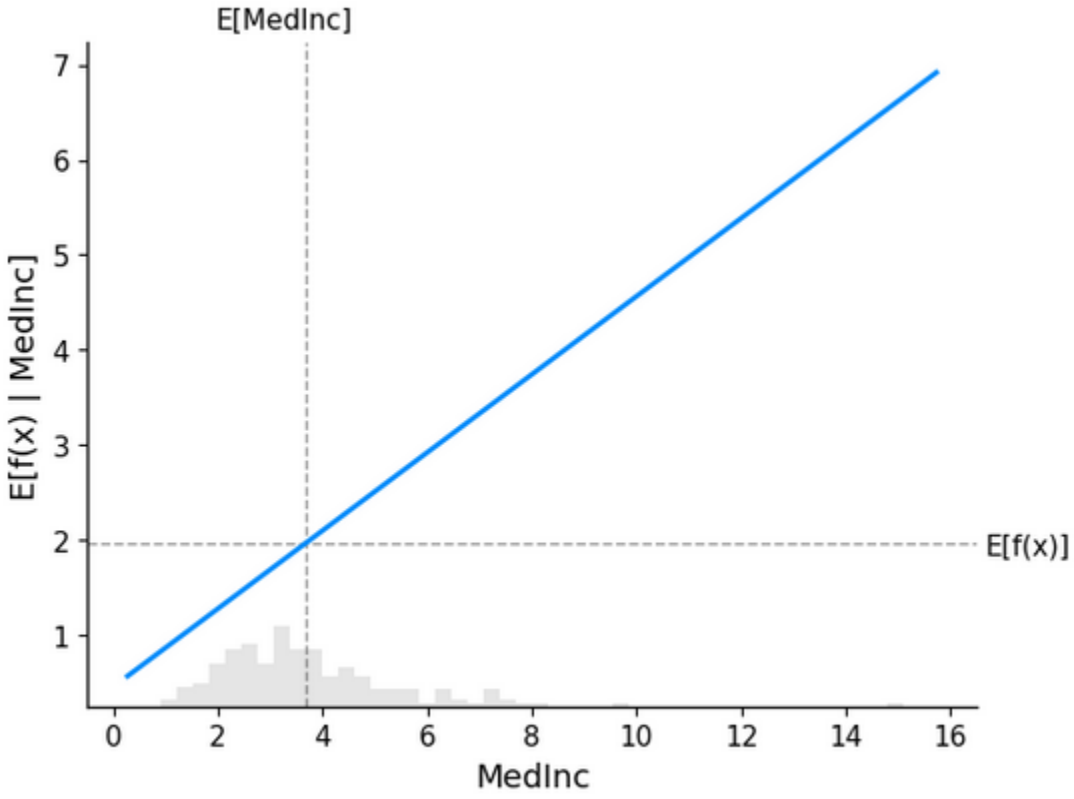
Si bien los coeficientes son útiles para indicarnos qué sucederá cuando cambiamos el valor de una característica de entrada, por sí solos no son una excelente manera de medir la importancia general de una característica. Esto se debe a que el valor de cada coeficiente depende de la escala de las características de entrada.

Por ejemplo, si midieramos la edad de una casa en minutos en lugar de años, entonces los coeficientes para la característica HouseAge se convertirían en 0.0115 / (365 * 24 * 60) = 2.18e-8. Claramente, el número de años desde que se construyó una casa no es más importante que el número de minutos, sin embargo, su valor de coeficiente es mucho mayor. Esto significa que la magnitud de un coeficiente no es necesariamente una buena medida de la importancia de una característica en un modelo lineal.

Una imagen más completa mediante gráficos de dependencia parcial

Para comprender la importancia de una característica en un modelo, es necesario entender tanto cómo el cambio en esa característica afecta la salida del modelo, como también la distribución de los valores de esa característica. Para visualizar esto en un modelo lineal, podemos construir un gráfico de dependencia parcial clásico y mostrar la distribución de los valores de la característica como un histograma en el eje x:

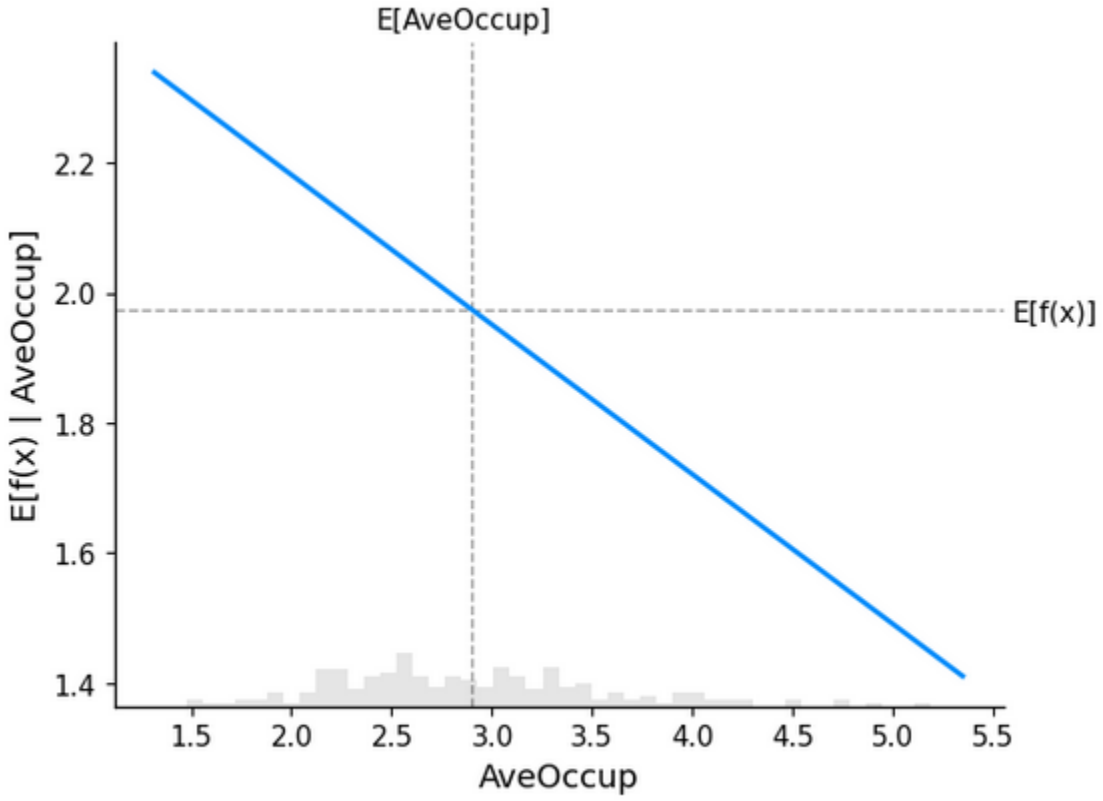
```
shap.partial_dependence_plot(
    "MedInc",
    model.predict,
    X_test,
    ice=False,
    model_expected_value=True,
    feature_expected_value=True,
)
print("Valor en la muestra 20:",X.iloc[indice_muestra]["MedInc"])
```




```
Valor en la muestra 20: 2.5859
```

```
shap.partial_dependence_plot(
    "AveOccup",
    model.predict,
    X_test,
    ice=False,
    model_expected_value=True,
    feature_expected_value=True,
)
print("Valor en la muestra 20:",X.iloc[indice_muestra]["AveOccup"])
```

```
print("Promedio de MedInc: ",np.mean(X["AveOccup"]))
```




```
Valor en la muestra 20: 3.107317073170732
Promedio de MedInc:  2.9328615962197926
```

Estas expresiones están relacionadas con conceptos estadísticos y matemáticos. Aquí está el significado de cada una:

1. Línea gris horizontal **:** En el gráfico, hay una línea horizontal gris que representa el valor esperado del modelo aplicado al conjunto de datos de viviendas de California. se refiere a la esperanza matemática o el valor esperado de la función de predicción del modelo . La esperanza matemática es una medida de tendencia central que representa el valor medio de una función ponderado por la probabilidad de cada valor en el dominio de la función. Matemáticamente, se define como:

   Donde es la función de densidad de probabilidad de la variable aleatoria . En el contexto de modelos estadísticos o de aprendizaje automático, podría ser la función de predicción del modelo para una variable de interés .

2. Línea gris vertical **:** La línea vertical gris representa el valor promedio de la característica “MedInc”. En este caso, se está calculando el valor esperado de la variable , que representa el ingreso medio. La expresión se interpreta como el promedio ponderado del ingreso medio en el conjunto de datos, donde cada valor de ingreso se pondera por su probabilidad de ocurrencia.

3. La intersección de estas dos líneas se considera el “centro” del gráfico de dependencia parcial con respecto a la distribución de datos.

4. Línea azul **:** La línea azul en el gráfico de dependencia parcial muestra el valor promedio de la salida del modelo cuando se fija la característica “MedInc” a un valor específico. representa la esperanza condicional de la función dado un valor específico de . En otras palabras, es el valor esperado de la función de predicción condicionado a que tiene un valor particular. Matemáticamente, se define como:

   Donde es la función de densidad de probabilidad condicional de dado un valor específico de .

Leyendo valores SHAP desde gráficos de dependencia parcial

La idea central detrás de las explicaciones basadas en valores SHAP (Shapley) de modelos de aprendizaje automático es utilizar resultados de asignación justa de la teoría de juegos cooperativos para asignar crédito a la salida de un modelo entre sus características de entrada.

Para evaluar un modelo existente cuando solo un subconjunto de características forma parte del modelo, integramos las otras características utilizando una formulación de valor esperado condicional. Esta formulación puede tomar dos formas:

o

En la primera forma, conocemos los valores de las características en porque los *observamos*. En la segunda forma, conocemos los valores de las características en porque los *configuramos*.

En general, la segunda forma suele ser preferible, tanto porque nos dice cómo se comportaría el modelo si interviniéramos y cambiáramos sus entradas, como porque es mucho más fácil de calcular. En este tutorial nos centraremos exclusivamente en la segunda formulación.

También utilizaremos el término más específico "valores SHAP" para referirnos a los valores Shapley aplicados a una función de expectativa condicional de un modelo de aprendizaje automático.

Los valores SHAP pueden ser muy complicados de calcular (en general, son NP-duros), pero los modelos lineales son tan simples que podemos leer los valores SHAP directamente de un gráfico de dependencia parcial.

Cuando estamos explicando una predicción, el valor SHAP para una característica específica es simplemente la diferencia entre la salida esperada del modelo y el gráfico de dependencia parcial en el valor de esa característica:

```
# calcular los valores SHAP para el modelo lineal
explainer = shap.Explainer(model.predict, X100)
shap_values = explainer(X_train)
```

Valores de SHAP del `Explainer`: Notamos las valores SHAP para cada característica en cada una de las 1000 muestras

```
shap_values_values = pd.DataFrame(shap_values.values)
shap_values_values.columns = X.columns
shap_values_values
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	-0.601507	0.189540	0.069317	-0.001785	-0.004325	-0.282248	0.687677	-0.327538
1	-0.861661	-0.027077	-0.504140	0.604135	-0.020319	0.154173	-1.850096	0.394172
2	-0.552930	-0.063180	0.116828	-0.047245	0.042115	0.038302	1.475775	-1.370558
3	0.088778	0.054154	-0.061548	-0.048238	-0.010657	0.167897	-0.263988	-0.194071
4	-0.063844	-0.216617	0.044294	-0.052527	0.021799	0.155009	0.866114	-0.999816
...	...	...	...	...	...	...	...	...
795	1.885240	-0.171489	-0.121660	-0.096117	0.000134	-0.096018	1.024725	-0.970157
796	0.348316	-0.126360	-0.047144	-0.037621	0.032535	-0.152593	0.890897	-0.871292
797	2.664841	0.189540	-0.185941	0.015745	-0.016383	0.025971	-0.962867	1.363044
798	-0.787360	0.054154	0.169873	-0.028714	-0.018888	0.190398	0.796722	-0.629074
799	0.836213	-0.063180	-0.101802	-0.046394	0.001291	0.026927	-1.200783	1.442135

800 rows × 8 columns

Los valores de SHAP, el valor esperado y los datos reales de la característica "MedInc" para la muestra 20.

```
# Seleccionar la fila 20 y todas la columna de `MedInc`
shap_values[indice_muestra : indice_muestra + 1, "MedInc"]
```

```
.values =
array([-1.22233169])

.base_values =
array([2.16080401])

.data =
array([1.1859])
```

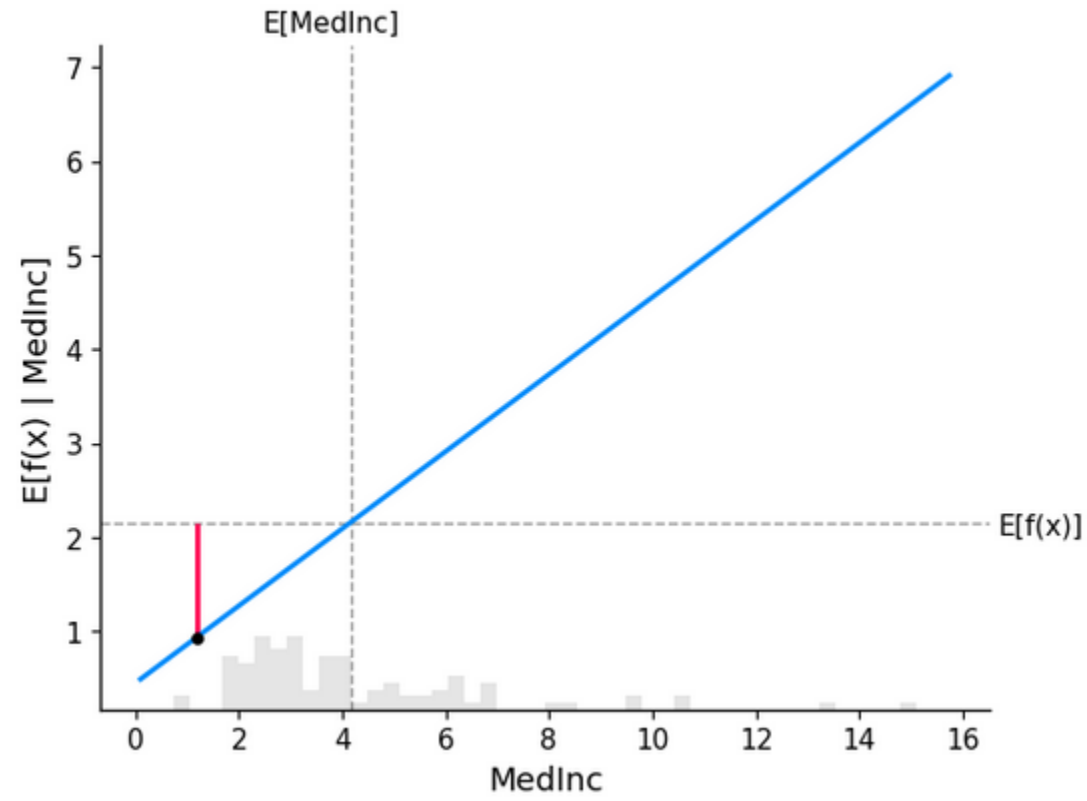
1. **`.values`:** Este es el valor específico del SHAP para la característica "MedInc" en la muestra 20. Indica la contribución de la característica "MedInc" en la predicción del modelo para la observación 20. Un valor negativo sugiere que la presencia de "MedInc" en la muestra 20 ha disminuido la predicción del modelo.

2. **`.base_values`:** Este es el valor base o el valor de referencia para la salida del modelo. Es el valor esperado de la salida del modelo cuando no se considera ninguna característica específica. En este caso, es el valor esperado del modelo para la muestra 20 sin tener en cuenta la característica "MedInc".

3. **`.data`:** Este es el valor específico de la característica "MedInc" en la muestra 20. Indica el valor real de "MedInc" en la observación 20. Es lo mismo que `X.iloc[indice_muestra]["MedInc"]`

La estrecha relación entre los valores Shapley y el gráfico de dependecia

```
shap.partial_dependence_plot(
    "MedInc",
    model.predict,
    X100,
    model_expected_value=True,
    feature_expected_value=True,
    ice=False,
    shap_values=shap_values[indice_muestra : indice_muestra + 1, :],
)
```



La linea roja que vemos en la gráfica la podemos expresar como la distancia entre y valor SHAP para la muestra 20

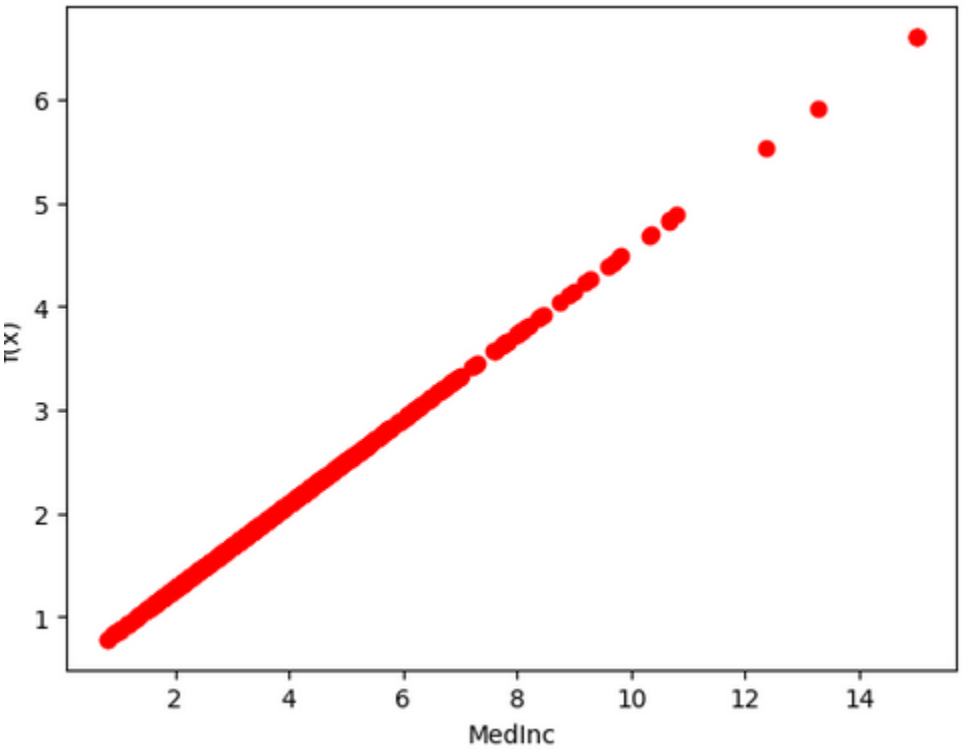
```
print("Valor independiente en la muestra 20
x:",shap_values[indice_muestra,"MedInc"].data)
print("Valor real del precio en la muestra 20 f(x):",y_train[indice_muestra])
print("Espectativa de salida E[f(X)]:",shap_values.base_values[indice_muestra])
print("E[f(X)] + SHAP:",shap_values.base_values[indice_muestra] +
shap_values[indice_muestra, "MedInc"].values)
print("Valor SHAP:",shap_values[indice_muestra, "MedInc"].values)
```

```
Valor independiente en la muestra 20 x: 1.1859
Valor real del precio en la muestra 20 f(x): 2.417
Espectativa de salida E[f(X)]: 2.160804011001239
E[f(X)] + SHAP: 0.9384723231832828
Valor SHAP: -1.2223316878179564
```

La estrecha correspondencia entre el clásico gráfico de dependencia parcial y los valores SHAP significa que si representamos gráficamente el valor SHAP para una característica específica en todo un conjunto de datos, trazaremos

exactamente una versión centrada en la media del gráfico de dependencia parcial para esa característica:

```
Y_shap = shap_values.base_values[:] + shap_values[:, "MedInc"].values
X_shap = shap_values[:,"MedInc"].data
# Graficar
plt.scatter(X_shap, Y_shap,color='r')
plt.xlabel("MedInc")
plt.ylabel("f(x)")
plt.show()
```



La naturaleza aditiva de los valores Shapley y el gráfico cascada

Una de las propiedades fundamentales de los valores Shapley es que siempre suman la diferencia entre el resultado del juego cuando todos los jugadores están presentes y el resultado del juego cuando ningún jugador está presente. Para los modelos de aprendizaje automático, esto significa que los valores SHAP de todas las características de entrada siempre sumarán la diferencia entre la salida del modelo de referencia (esperada) y la salida actual del modelo para la predicción que se está explicando. La forma más fácil de ver esto es a través de un gráfico de cascada que comienza en nuestra expectativa previa de fondo para el precio de una vivienda y luego agrega características una a una hasta llegar a la salida actual del modelo:

```
print("Valores SHAP en la muestra 20:")
display(shap_values_values.iloc[indice_muestra])
```

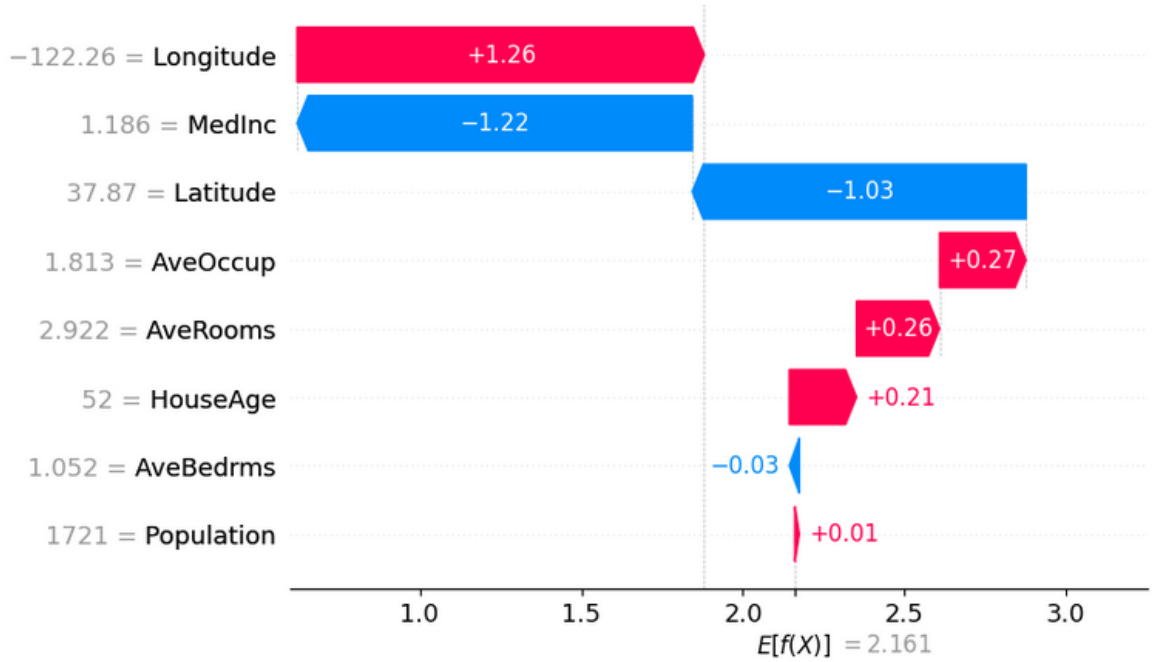
```
Valores SHAP en la muestra 20:
MedInc       -1.222332
HouseAge      0.207592
AveRooms      0.256761
AveBedrms    -0.030645
Population    0.013348
AveOccup      0.267623
Latitude     -1.032259
Longitude     1.259236
Name: 20, dtype: float64
```

```
fx=shap_values.base_values[indice_muestra]
```

```
for i in shap_values_values.iloc[indice_muestra]:
        fxp = fx + i
        print(f"{i} + {fx} = {fxp}")
        fx=fxp
```

```
-1.2223316878179564 + 2.160804011001239 = 0.9384723231832828
0.2075916325190298 + 0.9384723231832828 = 1.1460639557023127
0.2567605703603766 + 1.1460639557023127 = 1.4028245260626893
-0.030645212791689556 + 1.4028245260626893 = 1.3721793132709996
0.01334794921670136 + 1.3721793132709996 = 1.385527262487701
0.2676234197536575 + 1.385527262487701 = 1.6531506822413586
-1.0322592614761505 + 1.6531506822413586 = 0.6208914207652081
1.2592361689838567 + 0.6208914207652081 = 1.8801275897490648
```

```
shap.plots.waterfall(shap_values[indice_muestra])
```




```
print("Valor real del precio en la muestra 20:",y_train[indice_muestra])
print("Valor predicho por f(x) en la muestra
20:",model.predict(X_train.iloc[indice_muestra:indice_muestra+1]))
print("Espectativa de salida E[f(x)]:",shap_values.base_values[indice_muestra])
print("Valores reales en la muestra 20:")
display(X_train.iloc[indice_muestra])
```

```
Valor real del precio en la muestra 20: 2.417
Valor predicho por f(x) en la muestra 20: [1.88012759]
Espectativa de salida E[f(x)]: 2.160804011001239
Valores reales en la muestra 20:
MedInc          1.185900
HouseAge       52.000000
AveRooms        2.922023
AveBedrms       1.051633
Population    1721.000000
AveOccup        1.813488
Latitude       37.870000
Longitude    -122.260000
Name: 461, dtype: float64
```

Un gráfico de resumen simple tipo beeswarm

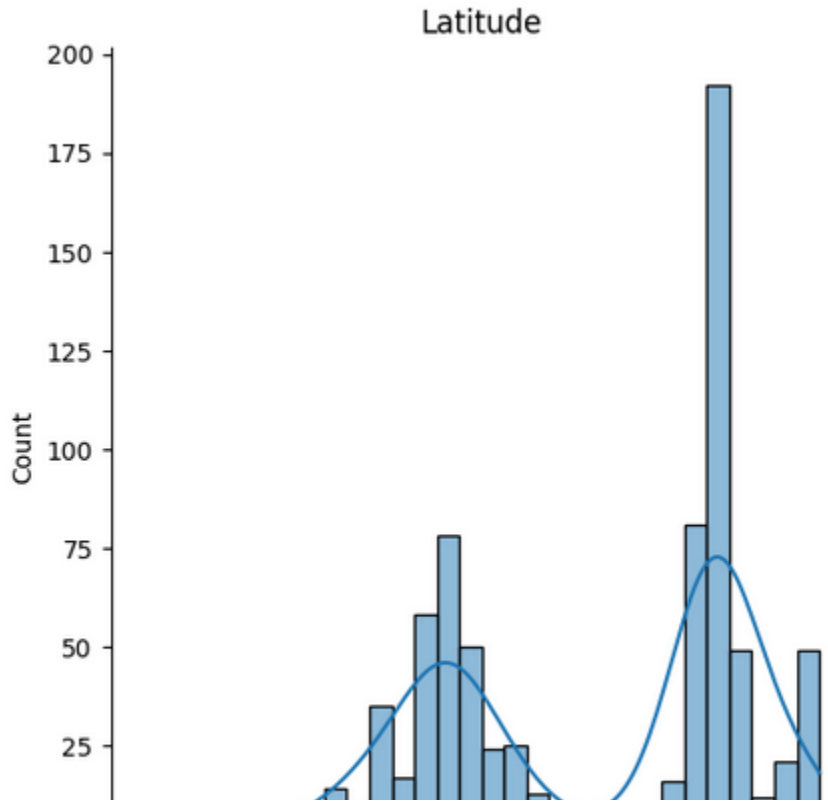
El gráfico de resumen beeswarm está diseñado para mostrar un resumen densamente informativo de cómo las principales características en un conjunto de datos afectan la salida del modelo. Cada instancia de la explicación dada se representa con un único punto en cada fila de características. La posición x del punto está determinada por el valor SHAP (shap_values.value[instancia, característica]) de esa característica, y los puntos se "amontonan" a lo largo de cada fila de características para mostrar la densidad. El color se utiliza para mostrar el valor original de una característica (shap_values.data[instancia, característica]).

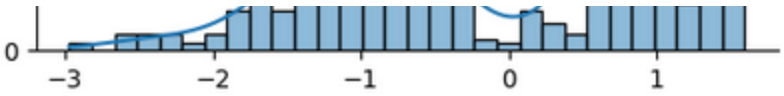
```
shap.plots.beeswarm(shap_values)
```

El gráfico al principio es confuso pero se puede explicar mejor simplemente como la dsitribución de valores de Shapley

```
sns.displot(shap_values[:, "Latitude"].values, bins=30,
kde=True).set(title='Latitude')
```

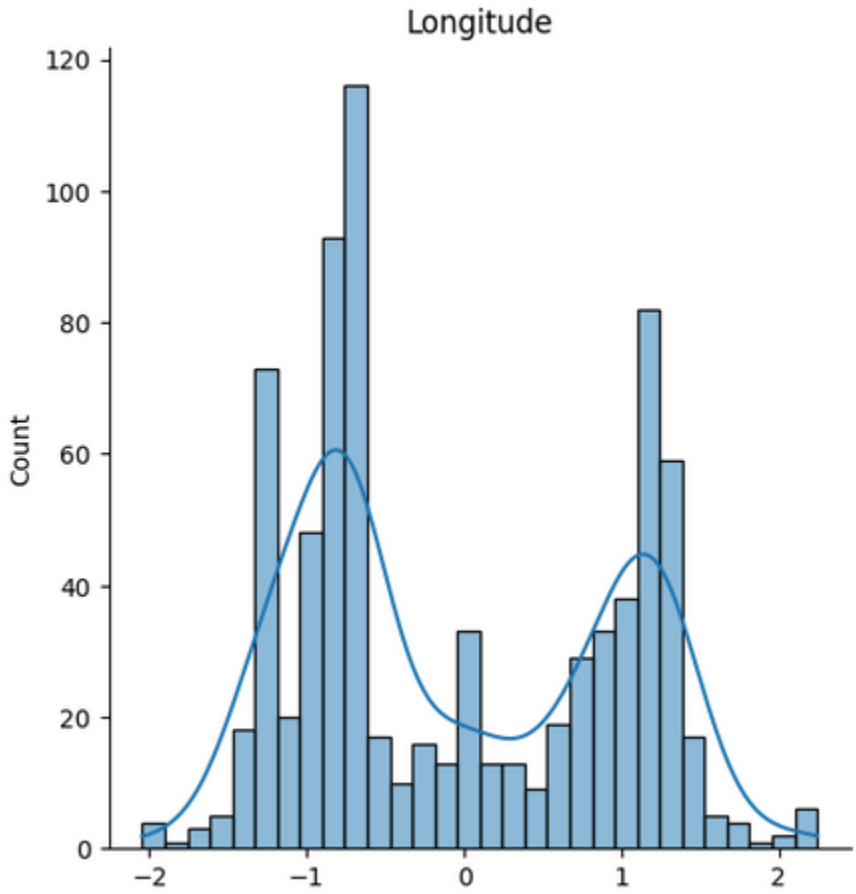
```
<seaborn.axisgrid.FacetGrid at 0x7fb821587c70>
```





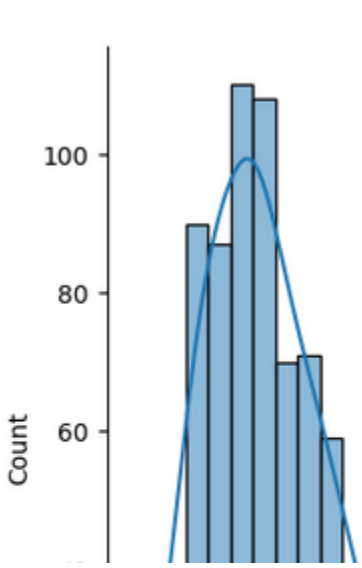

```
sns.displot(shap_values[:, "Longitude"].values, bins=30,
kde=True).set(title='Longitude')
```

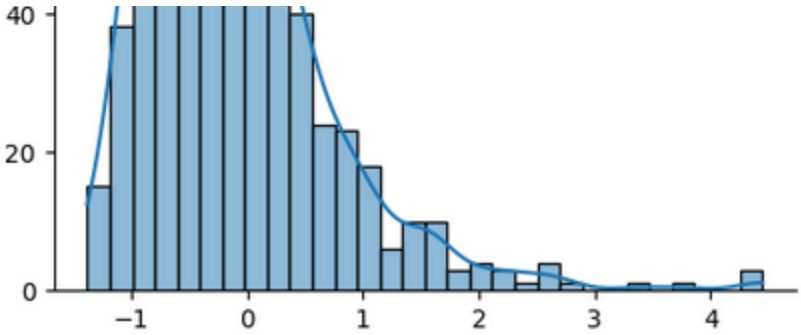
```
<seaborn.axisgrid.FacetGrid at 0x7fb8217a7d60>
```




```
sns.displot(shap_values[:, "MedInc"].values, bins=30,
kde=True).set(title='MedInc')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb8214e3430>
```





Aplicacion libreria shap dataset Irirs, modelacion Random Forest

Iris es un conjunto clásico de datos que nos sirve en el ámbito de la aprendizaje automático y la estadística. Contiene 150 observaciones de iris (flor), cada una perteneciente a una de las tres especies: setosa, versicolor y virginica. Cada observación tiene cuatro características: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo, todas medidas en centímetros. Para este conjunto de datos, dividimos los datos en conjuntos de entrenamiento y prueba, se puede entrenar a cualquier tipo de modelo, en este caso se utiliza KNN y random forest, con esto podemos realizar predicciones y evalúar la precisión del modelo utilizando métricas como el informe de clasificación y algunos sesgos de la muestra de datos.

(	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
..	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

Para este conjunto de datos, se necesita las siguientes librerias.

```
import shap
import time

import numpy as np
import sklearn
from sklearn.model_selection import train_test_split
```

Desde sklearn importamos train test split que nos sirve para dividir matrices en subconjuntos especialmente 2 tipos, para pruebas y entrenamiento. Posteriormente se puede definir una funcion de exactitud, que nos sirve para mirar que tan exacto es el modelo.

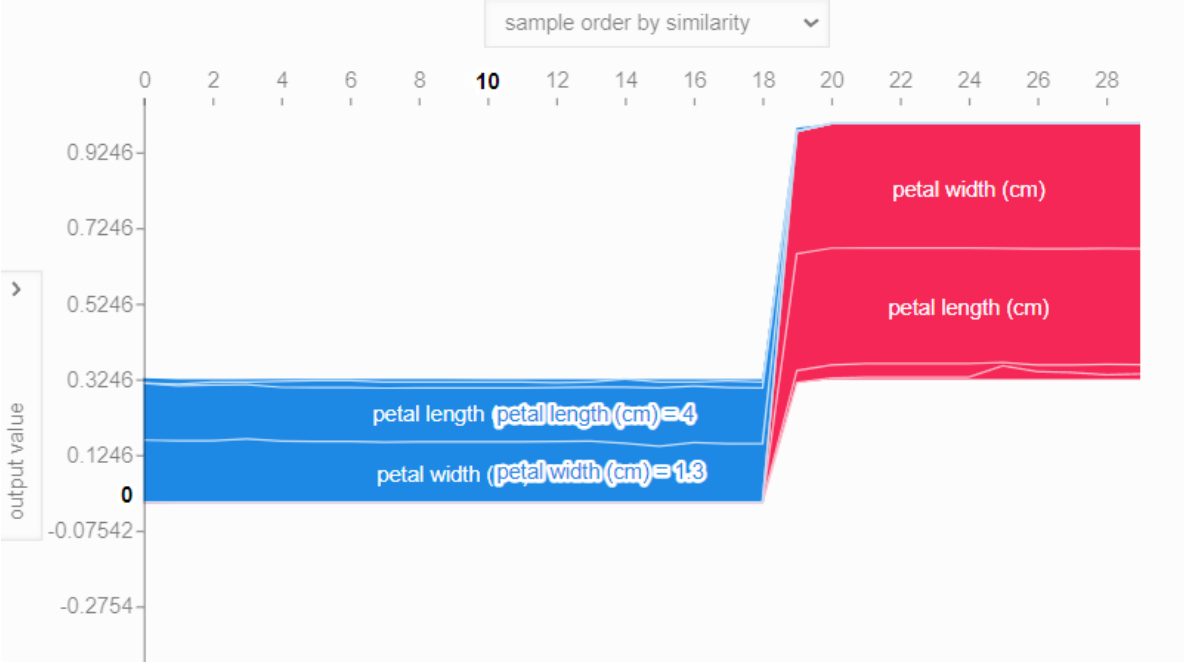
```
X_train, X_test, Y_train, Y_test = train_test_split(*shap.datasets.iris(),
test_size=0.2, random_state=0)
def print_accuracy(f):
    print("Accuracy = {}%".format(100 * np.sum(f(X_test) == Y_test) /
len(Y_test)))
    time.sleep(0.5)  # to let the print get out before any progress bars
shap.initjs()
```

Para el primer caso, vamos a utilizar random Forest, con esto podemos aproximar las predicciones y tambien connluir que tipo de modelacion es la mas precisa. from sklearn.ensemble import RandomForestClassifier

```
rforest = RandomForestClassifier(
    n_estimators=100, max_depth=None, min_samples_split=2, random_state=0
)
rforest.fit(X_train, Y_train)
print_accuracy(rforest.predict)

# explain all the predictions in the test set
```

```
explainer = shap.KernelExplainer(rforest.predict_proba, X_train)
shap_values = explainer.shap_values(X_test)
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test)
```



Con este gráfico tenemos los valores de shapley en el eje Y, en el eje X tenemos una muestra de 30 flores, los colores indican que tan fuerte es la caracteristica, rojo es mas fuerte, muestra las 4 caracteristicas de la flor iris y nos determina de que tipo de flor es la muestra de datos.

Aplicacion libreria shap dataset Diabetes, modelacion con regresión lineal

El conjunto de datos de diabetes contiene 442 muestras de pacientes. Cada muestra tiene 10 características las cuales son:

1. age age in years
2. sex
3. bmi body mass index
4. bp average blood pressure
5. s1 tc, total serum cholesterol
6. s2 ldl, low-density lipoproteins
7. s3 hdl, high-density lipoproteins
8. s4 tch, total cholesterol / HDL
9. s5 ltg, possibly log of serum triglycerides level
10. s6 glu, blood sugar level

El objetivo con estas caracteristicas es determinar cual de ellas incide mas para que un paciente tenga diabetes. De manera concisa, utilizamos el mismo metodo del caso de estudio de iris, con las mismas librerias, para modelara los datos de este data set vamos a usar regresion lineal.

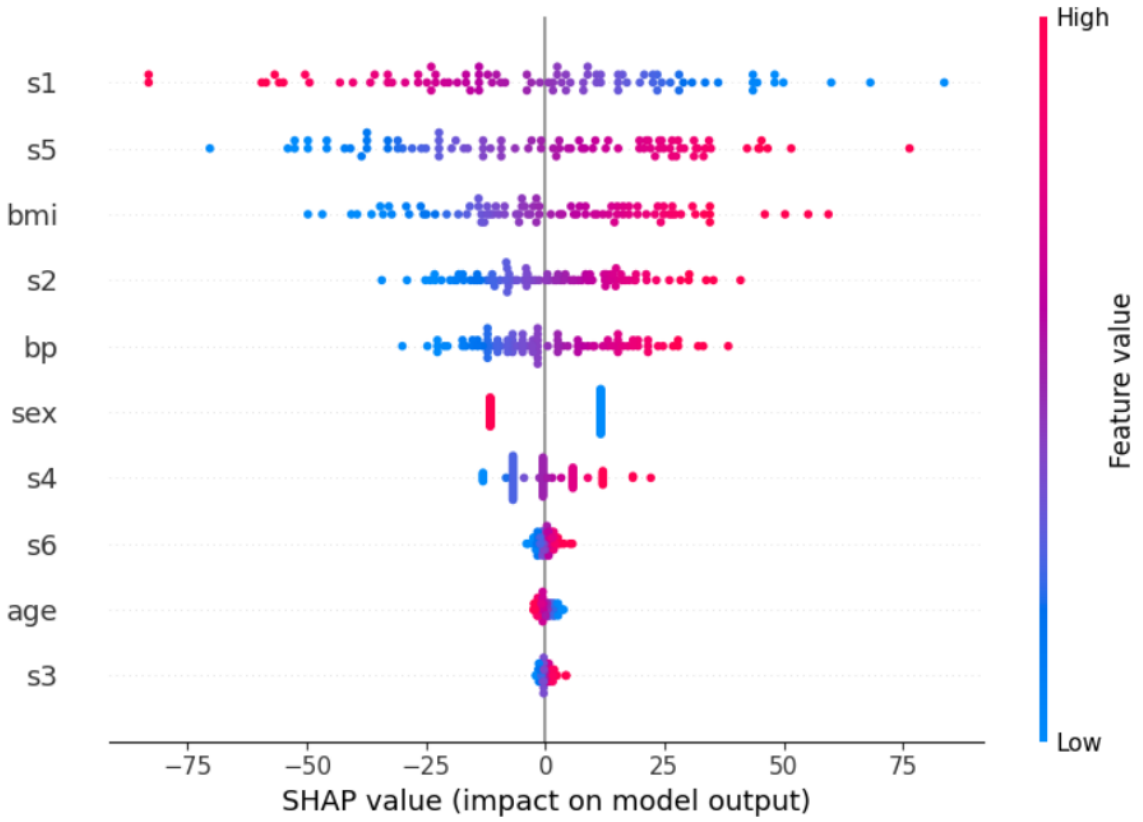
```
from sklearn import linear_model

lin_regr = linear_model.LinearRegression()
lin_regr.fit(X_train, y_train)

print_accuracy(lin_regr.predict)
Root mean squared test error = 58.51717127731562
ex = shap.KernelExplainer(lin_regr.predict, X_train_summary)
shap_values = ex.shap_values(X_test.iloc[0, :])
```

```
shap.force_plot(ex.expected_value, shap_values, X_test.iloc[0, :])
shap_values = ex.shap_values(X_test)
shap.summary_plot(shap_values, X_test)
```

La grafica generada es la siguiente:



En este grafico tenemos los valores de shapley en el eje X y en el Y tenemos las caracteristicas de este conjunto, S1 colesterol sérico, es la caracteristica mas predominante, quiere decir que los pacientes con colesterol sérico alto tienden a tener diabetes.

¿Y ahora qué?

Explotación de modelos de intelegincia artificial

Una forma de utilizar SHAP para auditar la seguridad es identificar las características que tienen un impacto significativo en las predicciones del modelo. Estas características pueden ser objetivos para los atacantes, ya que pueden ser manipuladas para alterar las predicciones del modelo. Por ejemplo, si un modelo de clasificación se utiliza para predecir si una persona es elegible para un préstamo, las características que tienen un impacto significativo en las predicciones podrían ser la edad, el historial crediticio y los ingresos. Un atacante podría intentar manipular estas características para aumentar las posibilidades de que una persona sea aprobada para un préstamo.

Otra forma de utilizar SHAP para auditar la explotabilidad es identificar las combinaciones de características que pueden conducir a predicciones incorrectas. Estas combinaciones pueden ser explotadas por atacantes para engañar al modelo para que haga predicciones falsas. Por ejemplo, si un modelo de clasificación se utiliza para diagnosticar una enfermedad, las combinaciones de características que pueden conducir a predicciones incorrectas podrían ser síntomas que son comunes a varias enfermedades. Un atacante podría intentar manipular estas características para que el modelo diagnostique incorrectamente a una persona con una enfermedad que no tiene.

Otro ejemplo sería, si un modelo de clasificación de malware estático se utiliza para identificar archivos maliciosos en base a su contenido, las características que tienen un impacto significativo en las predicciones podrían ser la presencia de ciertas cadenas de caracteres, la estructura del archivo o el uso de ciertas funciones. Un atacante podría intentar manipular estas características para que el modelo clasifique incorrectamente un archivo malicioso como no malicioso.

Conclusión

Los valores SHAP son una herramienta de interpretación de modelos de aprendizaje automático que nos permiten comprender cómo las características individuales contribuyen a la salida del modelo.

El informe aborda la aplicación de la teoría de juegos cooperativos, específicamente el valor de Shapley, en el contexto del "Juego de los Guantes". Proporciona un análisis detallado de cómo asignar valores de manera justa a los jugadores en un juego colaborativo.

Luego, se exploran aplicaciones prácticas en el campo del aprendizaje automático. Se emplea un modelo de regresión lineal para el avalúo de viviendas en California, utilizando el enfoque SHAP para interpretar las predicciones del modelo. Se destacan las ventajas de SHAP, como la interpretación intuitiva y la visualización clara de la importancia de las características.

El informe continúa presentando un modelo de clasificación Random Forest en el conjunto de datos Iris y un modelo de regresión lineal en el conjunto de datos Diabetes. En ambos casos, se utiliza SHAP para comprender las decisiones del modelo y se discuten las ventajas de esta metodología.

Se enfatiza la importancia de la visualización de gráficos SHAP para simplificar la interpretación. Se concluye que, aunque SHAP es versátil y ofrece transparencia en modelos de inteligencia artificial, la interpretación puede volverse desafiante en modelos extremadamente complejos y no lineales. Pero también se conlcuyen las siguientes ventajas y desventajas:

Los valores SHAP tienen una serie de ventajas, entre ellas:

- **Son interpretables:** Los valores SHAP tienen un significado intuitivo, ya que representan la contribución marginal de cada característica a la salida del modelo.
- **Son robustos:** Los valores SHAP son relativamente robustos a los cambios en los datos de entrenamiento.

Sin embargo, los valores SHAP también tienen algunas desventajas, entre ellas:

- **Pueden ser computacionalmente costosos de calcular.**
- **Pueden ser engañosos en presencia de multicolinealidad.**

Referencias

[5^]: Handbook of the Shapley Value. (2019). United Kingdom: CRC Press. Retrived from: http://rguir.inflibnet.ac.in/bitstream/123456789/16159/1/Handbook of the Shapley Value.pdf

---

1. Molnar, C. (2023, 30 de noviembre). SHAP: A Unified Approach to Explain the Output of Any Machine Learning Model. [En línea]. Recuperado de https://shap.readthedocs.io/en/latest/ ↩
2. Bamio Martínez, D. (2023). El valor de Shapley. Trabajo Fin de Grao, Universidad de Santiago de Compostela. [En línea]. Disponible en: https://minerva.usc.es/xmlui/bitstream/handle/10347/26022/Bamio Martínez, David.pdf ↩
3. Vesga Ferreira, J. C., Granados Acuña, G., & Sierra Carrillo, J. E. (2015). El valor de shapley como estrategia de optimización de recursos sobre Power Line Communication (PLC). Disponible en: http://www.scielo.org.co/pdf/ince/v11n22/v11n22a09.pdf ↩
4. L. S. Shapley, "A value for n-persons games in Contributions to the Theory of Games II," Annals of Mathematics Studies, no. 28, pp. 307–317, 1953. ↩
5. Colaboradores de los proyectos Wikimedia. (2022, March 04). Diferencia de conjuntos - Wikipedia, la enciclopedia libre. Retrieved from https://es.wikipedia.org/w/index.php?title=Diferencia_de_conjuntos ↩
6. Harsanyi, J.C. (1959) A bargaining model for cooperative n-person games. In: Tucker, A.W., Luce, R.D. (eds) Contributions to the theory of games IV. Princenton University Press, Princenton, pp. 325-355 ↩
7. Grabisch, M. (2016). Set Functions, Games and Capacities in Decision Making. Germany: Springer International Publishing. ↩
8. Rosero, F., Díaz, M., Burgos, S., (2023). Proyecto SHAP. Tomado de https://github.com/IA-2023-2-SHAP/proyecto-shap ↩
9. Torgo, L. (2023). California Housing Prices Dataset. Retrieved from https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html ↩