VALORACIÓN DE DATOS BASADA EN TEORÍA DE JUEGOS

UNIVERSIDAD COMPLUTENSE DE MADRID
UNIVERSIDAD POLITÉCNICA DE MADRID

TRABAJO FINAL DE MÁSTER

10 de septiembre de 2023

Autor: Francisco AGUILAR MARTÍNEZ
Tutores: Carlos GREGORIO RODRÍGUEZ
Miguel DE BENITO DELGADO

UNIVERSIDAD
COMPLUTENSE
MADRID

VALORACIÓN DE DATOS BASADA EN TEORÍA DE JUEGOS

UNIVERSIDAD COMPLUTENSE DE MADRID
UNIVERSIDAD POLITÉCNICA DE MADRID

TRABAJO FINAL DE MÁSTER

10 de septiembre de 2023

Autor: Francisco AGUILAR MARTÍNEZ
Tutores: Carlos GREGORIO RODRÍGUEZ
Miguel DE BENITO DELGADO

Índice general

Resumen

Este trabajo se centra en la valoración de datos basada en teoría de juegos. Se comienza llevando a cabo una revisión exhaustiva de la literatura actual sobre el tema, introduciendo posteriormente los conceptos básicos necesarios para la comprensión de este, y destacando la relevancia de los juegos cooperativos, los cuales actuarán como punto de unión entre la valoración de datos y la teoría de juegos en este estudio.

El objetivo principal es comparar los diferentes métodos descritos en la literatura, haciendo especial énfasis en *Data Banzhaf* [1]. A lo largo del estudio, se aborda el concepto de *safety margin*, el cual servirá para medir la robustez de los métodos de valoración. Cualidad en la que *data Banzhaf* ha demostrados sobresalir por encima del resto de semivalores.

Otro objetivo que perseguía este trabajo era familiarizarse con buenas prácticas tanto en ciencia de datos como en ingeniería de software. Para ello, se ha utilizado DVC como control de versiones especializado para proyectos de ML y se han seguido las recomendaciones dadas en ML Reproducibility Challenge de papers with code. Con esto se ha conseguido un código reproducible, una fundamental en la ciencia, pero no muy común en ciencia de datos. El código se encuentra disponible en Fran-AM/TFM.

Finalmente, los métodos estudiados se comparan en dos tareas de ML que suelen usarse como métrica de rendimiento para este tipo de estudios: detección de datos mal etiquetados y entrenamiento ponderado de modelos. Para estos experimentos, se seleccionan varios conjuntos de datos comúnmente referenciados en la literatura en trabajos similares.

Introducción

Obtener datos de entrenamiento adecuados suele ser uno de los desafíos más grandes y costosos en aprendizaje automático (ML). En muchas aplicaciones prácticas, parte de los datos puede estar contaminada con ruido debido a valores atípicos o errores en el etiquetado. Entender qué tipos de datos son más beneficiosos para el entrenamiento puede guiar a una mejor adquisición y selección de estos. Debido a estos factores, la valoración de datos ha emergido como un área esencial de investigación en el campo del ML.

Dentro del contexto del ML, el valor de un dato particular está intrínsecamente relacionado con otros datos utilizados en el entrenamiento del modelo. Por ejemplo, el valor de una muestra específica puede disminuir si se incorporan al conjunto de entrenamiento otras similares a esta. Por ello, nos centraremos en un contexto de aprendizaje supervisado, donde se dispone con un conjunto de datos, un modelo, y una métrica de error prefijados. Es importante enfatizar que no estamos definiendo un valor universal para los datos, sino un valor que depende de los elementos previamente prefijados.

Asignar valor a los datos es una tarea compleja que implica consideraciones multifactoriales. En este contexto, la teoría de juegos emerge como una herramienta para abordar esta problemática. En el ámbito del aprendizaje automático, en lugar de jugadores, contamos con datos individuales. Si consideramos un problema de ML como un juego, el objetivo es maximizar una métrica de rendimiento, como la precisión del algoritmo. Aquí, el "beneficio" es el impacto positivo que un dato (o conjunto de datos) tiene en el rendimiento del algoritmo.

Una de las piezas más destacadas en este entramado es el valor de Shapley, procedente de la teoría cooperativa de juegos. Este concepto permite asignar a cada jugador (o dato, en nuestro caso) una porción del valor total del juego, basándose en sus contribuciones a lo largo de todas las posibles coaliciones. Es en torno al valor de Shapley que se desarrollan todos los conceptos que estudiaremos a lo largo de este trabajo, como es *data Banzhaf*, derivado del valor de Banzhaf, que ha probado ser el más robusto entre todos los valores discutidos en la literatura.

Sin embargo, aunque métricas como el valor de Shapley o Banzhaf son teóricamente atractivas, calcularlas en escenarios reales es prácticamente inviable debido a la explosión combinatoria de posibles coaliciones. Por esto, estimar los valores de los datos mediante técnicas basadas en teoría de juegos es un gran desafío, ya

que implica equilibrar la precisión y la eficiencia computacional. Los estimadores juegan un papel fundamental en este aspecto, proporcionando soluciones prácticas para aproximarse a los valores exactos sin incurrir en costes prohibitivos.

¿Qué se aporta?

En este trabajo se han construido las bases para llevar a cabo una comparativa de calidad de los diferentes métodos de valoración de datos basados en teoría de juegos y sus estimadores. Mediante el desarrollo de una infraestructura de software escalable, reproducible y bien documentada, que permitirá continuar con el desarrollo de las pruebas y experimentos.

CAPÍTULO 1

Estado del arte

Desde la aparición del valor de Shapley como un método de reparto justo de recompensas en juegos cooperativos [2], este concepto se ha utilizado en diversos campos como la economía [3], estudio de sistemas multiagente [4] e incluso en áreas en las que su aplicación puede resultar menos evidente como la genética [5]. Esta versatilidad se debe, en parte, a su sólida base matemática y a sus intuitivas interpretaciónes, entre las que podemos resaltar:

- Pago justo: el valor de Shapley de un jugador es la cantidad que este debería recibir si los pagos se distribuyesen de manera que los jugadores fueran compensados en función de su contribución al beneficio total.
- Poder de negociación: El valor de Shapley puede interpretarse como una medida del poder de negociación de un jugador. Un jugador tiene más poder de negociación si su ausencia causa una mayor disminución en la recompensa total que se puede obtener.

El valor de Shapley se basa en una serie de axiomas fundamentales, que garantizan propiedades como su equidad, eficiencia y simetría. En economía, relajar estos axiomas con el objetivo de dar lugar a nuevas formas de reparto de recomenpensa ha sido uno de los principales temas de estudio. Ejemplo de esto sería el concepto de semivalor, el cual se obtiene al eliminar el axioma de eficiencia [6, 7]. Este axioma asegura que la suma de los valores de todos los jugadores sea igual a la recompensa total disponible. Por tanto, al suprimirlo, los semivalores permiten cierta flexibilidad en este aspecto, lo que puede ser útil en situaciones en las que no todos los beneficios se pueden distribuir. Del mismo modo, eliminar el axioma de simetría, que establece que dos jugadores con igual contribución deben recibir igual recompensa, lleva al valor de Banzhaf [8], que proporciona una medida de poder de un jugador basada en cuánto puede cambiar el resultado de un juego al unirse o abandonar una coalición. Cabe destacar que tanto el valor de Shapley como el valor de Banzhaf pueden obtenerse como particularizaciones del concepto de semivalor.

Debido a la naturaleza combinatoria del valor de Shapley, el cálculo de este es altamente costoso a nivel computacional, y resulta en una tarea cuya complejidad crece exponencialmente al aumentar el número de jugadores. Es por esto que surgen métodos de estimación del mismo, la mayoría de estos métodos se basan en técnicas de Montecarlo. En 1960, Irwin Mann y el propio Shapley mencionan las estimaciones basadas en muestreo de permutaciones [9]. Pero no es hasta 2015 que se lleva a cabo un análisis de la complejidad a la hora de muestrear usando dicha técnica [10]. Tras esto, en 2019, Covert propone un nuevo método de estimación basado en la técnica de muestreo por importancia que mejora los actuales [11]. De entre los métodos de estimación, cabe mencionar el *ApproShapley* propuesto en [12] y desarrollado por los profesores J.Castro, D.Gómez y J.Tejada de la UCM.

Una de las primeras apariciones del valor de Shapley en el campo del aprendizaje computacional data del año 2005 como un método de selección de variables [13]. Más tarde, en 2017, se utiliza en el diseño del marco SHAP [14], enfocado en la evaluación de la importancia de variables en modelos de predicción. Sin embargo, no es hasta el año 2019 en el que se introduce como una alternativa a los métodos de valoración de datos del momento [15], acuñándose así el concepto *Data Shapley*.

Al igual que el cálculo del valor de Shapley, el cálculo de *Data Shapley* es altamente costoso a nivel computacional, por lo que surgen también varios métodos de aproximación, entre los que podemos destacar *Group Testing* [16], métodos de aproximación y cálculo exacto para problemas en los que se aplican métodos como KNN o derivados de este [17] y diversas técnicas basadas en métodos de Montecarlo como las vistas en [15].

Cuando se prueba la eficacia de *Data Shapley* en problemas de aprendizaje computacional como la detección de outliers o datos corruptos [15], la investigación sigue el mismo camino que años antes en teoría de juegos y se empiezan a reciclar conceptos como los semivalores o el valor de Banzhaf. Es en la línea de los semivalores que en 2022 surge *Beta Shapley* [18], una generalización de *Data Shapley* que supera los resultados de los métodos más actuales de valoración de datos en varias tareas como son detección de muestras mal etiquetadas y selección de puntos problemáticos a la hora de entrenar un modelo.

En 2023, aparece *Data Banzhaf* [1], un nuevo método de valoración de datos derivado del valor de Banzhaf. Este nuevo método surge como una solución a la falta de robustez de las herramientas de valoración de datos existentes. Esta falta de robustez es causada en parte por factores difíciles de controlar como la aleatoriedad del método del descenso del gradiente estocástico, el cual es ampliamente usado hoy en día. Para solventar esta falta de robustez, *Data Banzhaf* se apoya en el concepto de *Safety Margin*, y demuestra que el valor de Banzhaf es el semivalor con mayor *Safety Margin*. *Data Banzhaf* supera a los existentes métodos de valoración basados en semivalores en varias tareas de aprendizaje automático. Como son detección de datos mal etiquetados y entrenamiento ponderado.

Aunque en este trabajo nos centramos en métodos de valoración de datos que se derivan directamente de conceptos de la teoría de juegos, existen otros métodos que, aún utilizando teoría de juegos, siguen enfoques relativamente distintos. Podemos destacar algunas obras como [19] en la que sugieren un método de valoración de datos para modelos generativos que utiliza la discrepancia media máxima (MMD) entre la fuente de datos y la distribución real de datos. En [20] proponen una medida de diversidad, llamada volumen robusto (RV), para valorar las fuentes de datos. La robustez de RV se discute en términos de la estabilidad frente a la replicación de datos. Finalmente en [21] utilizan diferencias estadísticas entre los datos de origen y un conjunto de datos de referencia como la métrica de valoración. Estas diferencias estadísticas se miden mediante el uso de los conceptos de diversidad y relevancia de los datos previamente comentados.

Por tener una visión completa de las diferentes técnicas, se incluye una referencia que se utiliza una técnica muy distinta a las comentadas. Esta consiste en llevar a cabo valoración de datos mediante aprendizaje por refuerzo [22]. En dicho trabajo, se utiliza una red neuronal profunda para obtener un estimador de la probabiliadad de cada dato de ser usado en el entrenamiendo del modelo de predicción. Este estimador se obtiene mediante aprendizaje por refuerzo.

CAPÍTULO 2

Marco teórico

2.1. Teoría de juegos

Definición y conceptos básicos

La *teoría de juegos* puede ser entendida como la rama de las matemáticas que analiza situaciones en las que el resultado para cada jugador o participante depende no solo de sus propias decisiones, sino también de las tomadas por otros jugadores. Estas situaciones se conocen como *juegos*.

Definición 2.1.1. Un *juego* es una interacción entre jugadores racionales[1], mutuamente conscientes, en la que las decisiones de un jugador impactan en las ganancias de otros. Un juego se define por:

- **Los jugadores que intervienen.** Cada *jugador* es un agente que tiene a su disposición diversas estrategias basadas en las posibles recompensas que podría recibir.
- **Las estrategias disponibles para cada jugador.** Una *estrategia* es un plan de acción que un jugador puede adoptar dentro de un juego. Esta estrategia dicta las acciones que tomará en cada situación que se presente en el juego.
- **Las ganancias de cada jugador en función de los resultados.** Las *ganancias* son representaciones de las motivaciones de los jugadores, pudiendo representar beneficios, cantidades, o simplemente reflejar la conveniencia de los diferentes desenlaces.

Un tipo particular de juego, que cobra especial importancia en este trabajo son los juegos cooperativos.

Definición 2.1.2. Un *juego cooperativo* es aquel en el que los jugadores pueden comunicarse y negociar con el fin de establecer acuerdos vinculantes.

[1] Entendemos la racionalidad como el hecho de que cada jugador intenta maximizar su propio beneficio.

Los acuerdos mencionados se denominan *coaliciones* . Se trata de grupos de jugadores que eligen actuar juntos para lograr un objetivo común. Las coaliciones pueden variar desde la compuesta por todos los jugadores hasta la que incluye a un único jugador.

Un juego cooperativo se define completamente a través de su conjunto de jugadores N y de su función característica. Esta función establece la relación entre la formación de coaliciones y los beneficios que obtienen los jugadores, asignando a cada posible coalición (es decir, a cada subconjunto del conjunto potencia de N, denotado 2^N) un beneficio específico que pueden alcanzar.

Definición 2.1.3. Una *función característica* v es una función

$$\begin{aligned} v : 2^N &\longrightarrow \mathbb{R} \\ S &\longmapsto v(S) \end{aligned}$$

que asigna a cada posible coalición S la máxima ganancia que sus jugadores pueden obtener, independientemente de lo que haga el resto de jugadores.

Un interrogante esencial emerge tras establecer coaliciones y ganancias: ¿cómo se reparten los beneficios entre los miembros de la coalición? Esta pregunta nos conduce directamente al valor de Shapley.

2.1.1. El valor de Shapley

A partir de ahora, consideraremos la situación de un juego cooperativo. Supondremos un conjunto de N jugadores e identificaremos dicho juego mediante su función característica v.

El *valor de Shapley* es un concepto de teoría de juegos que asigna de manera equitativa las ganancias entre los miembros de una coalición. Propuesto por Lloyd Shapley en 1952 [2], se fundamenta en los siguientes axiomas:

- **Simetría**. Si dos jugadores son simétricos, es decir, si su contribución a cualquier coalición es la misma, entonces poseen el mismo valor.

  Si $\forall S \subseteq N \setminus \{i, j\}$, $v(S \cup \{i\}) = v(S \cup \{j\}) \implies \phi_{Shapley}(i; v) = \phi_{Shapley}(j; v)$.

- **Eficiencia**. El valor total producido por la coalición conformada por todos los jugadores se distribuye entre los jugadores. Es decir,

  $$v(N) = \sum_{i \in N} \phi_{Shapley}(i; v).$$

- **Linealidad**. Dados dos juegos u y v, el valor del juego $u + v$ es la suma de los valores de cada juego. Es decir,

  $$\phi_{Shapley}(i; u + v) = \phi_{Shapley}(i; u) + \phi_{Shapley}(i; v) \ \forall i \in N.$$

- **Jugador nulo**. Los jugadores que no aportan a ninguna coalición tendrán valor nulo. Es decir,

$$\text{Si } \forall S \subseteq N \setminus \{i\},\ v(S \cup \{i\}) = v(S) \implies \phi_{Shapley}(i; v) = 0.$$

El siguiente teorema, extraído de [2], prueba que se trata del único método de valoración de datos que satisface estos axiomas.

Teorema 2.1.4. *Existe una única función $\phi_{Shapley}$ satisfaciendo los axiomas de simetría, eficiencia, linealidad y jugador nulo, y viene dada por la fórmula:*

$$\begin{aligned}\phi_{Shapley}(i; v) &= \sum_{S \subseteq N} \frac{(s-1)!(n-s)!}{n!}(v(S) - v(S \setminus \{i\})) \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!}(v(S \cup \{i\}) - v(S)).\end{aligned}$$

Donde $n = |N|$ y $s = |S|$ son los cardinales de N y S respectivamente.

La demostración detallada se encuentra en la sección 3 de [2].

Más adelante nos será útil expresar el valor de Shapley de una forma alternativa, que nos permitirá relacionarlo mejor con otros conceptos.

Proposición 2.1.5. *El valor de Shapley puede ser expresado como:*

$$\phi_{Shapley}(i; v) = \frac{1}{n} \sum_{k=1}^{n} \binom{n-1}{k-1}^{-1} \sum_{\substack{S \subseteq N \setminus \{i\} \\ |S| = k-1}} (v(S \cup \{i\}) - v(S)). \tag{2.1}$$

Demostración. Teniendo en cuenta que

$$\binom{n-1}{k-1}^{-1} = \frac{(k-1)!(n-k)!}{(n-1)!},$$

podemos reescribir la ecuación 2.1 como:

$$\begin{aligned}\phi_{Shapley}(i, v) &= \frac{1}{n} \sum_{k=1}^{n} \frac{(k-1)!(n-k)!}{(n-1)!} \sum_{\substack{S \subseteq N \setminus \{i\} \\ |S| = k-1}} (v(S \cup \{i\}) - v(S)) \\ &= \sum_{k=1}^{n} \frac{(k-1)!(n-k)!}{n!} \sum_{\substack{S \subseteq N \setminus \{i\} \\ |S| = k-1}} (v(S \cup \{i\}) - v(S)) \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!}(v(S \cup \{i\}) - v(S)).\end{aligned}$$

□

2.1.2. Semivalores

Los semivalores son una generalización del valor de Shapley, que surgen al relajar el axioma de eficiencia. Así, permiten una variedad más amplia de métodos de valoración en el estudio de juegos cooperativos, siendo especialmente útiles en escenarios donde no es necesario o deseado que la totalidad de la recompensa sea distribuida.

La clave de los semivalores es que asignan pesos a las coaliciones basándose en el tamaño de estas. Estos pesos son utilizados para calcular la contribución marginal promedio de cada jugador. A diferencia del valor de Shapley, que es único dada su definición basada en axiomas [2], hay múltiples semivalores posibles dependiendo de cómo se determinen los pesos. Esta variabilidad en los semivalores queda formalizada en el siguiente teorema:

Teorema 2.1.6. *Representación de semivalores [7].*

Una función ϕ es un semivalor sii existe una función peso $w : \{1, \ldots, n\} \to \mathbb{R}$ tal que

$$\sum_{k=1}^{n} \binom{n-1}{k-1} w(k) = n,$$

que permite representar ϕ mediante la expresión:

$$\phi(i; v) = \sum_{k=1}^{n} \frac{w(k)}{n} \sum_{\substack{S \subseteq N \setminus \{i\} \\ |S| = k-1}} (v(S \cup \{i\} - v(S))).$$

La demostración se puede consultar en [7].

Este teorema establece una biyección entre semivalores y funciones de peso. Esta correspondencia uno a uno nos permite identificar y estudiar semivalores a través de sus funciones de peso. Además de esto, proporciona una fórmula general para representar cualquier semivalor en términos de su función de peso.

Finalmente, es importante destacar que el valor de Shapley es un caso particular de semivalor. En este caso la función de peso será $w_{Shapley} = \binom{n-1}{k-1}^{-1}$.

2.1.3. El valor de Banzhaf

El *valor de Banzhaf*, también denominado índice de poder de Banzhaf [8], es una métrica de teoría de juegos cooperativos que busca cuantificar el poder e influencia de un jugador dentro de una coalición. Este índice fue introducido por John F. Banzhaf III en 1965, con la finalidad de ofrecer una herramienta analítica que pudiese determinar el poder de influencia de un jugador, especialmente en escenarios de votación ponderada.

Para comprender mejor la esencia y la utilidad del valor de Banzhaf, es fundamental familiarizarse con ciertos conceptos relacionados:

- *Sistema de votación ponderado*: es un sistema de votación en el que cada jugador tiene un peso o poder de voto particular. Para que una propuesta sea

aprobada, la suma de los pesos de los jugadores que votan a favor debe superar un umbral o cuota establecida.

- *Jugador pivote*: ee considera que un jugador actúa como pivote si, al modificar su voto de negativo a positivo, la propuesta es aprobada. Sin embargo, si se abstuviera o mantuviera su voto en contra, la propuesta sería rechazada.
- *Índice de poder de Banzhaf*: este índice mide la frecuencia con la que un jugador se convierte en pivote. Es importante destacar que el poder de un jugador no siempre es directamente proporcional a su peso en la votación.

Definición 2.1.7. El *valor de Banzhaf* de un jugador i en un juego cooperativo v viene dado por la expresión:

$$\phi_{Banzhaf}(i;v) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus \{i\}} [v(S \cup \{i\}) - v(S)]. \tag{2.2}$$

A pesar de sus similitudes con el valor de Shapley, el índice de Banzhaf se distingue en su enfoque y la forma de asignar poder a los jugadores. Mientras que el valor de Shapley se basa en contribuciones marginales promediadas, el índice de Banzhaf se enfoca en la capacidad de un jugador de influir en el resultado final de una votación. Específicamente, es un semivalor con un peso asociado dado por $w_{Banzhaf} = \frac{1}{2^{n-1}}$.

2.2. Valoración de datos

Hoy en día, el dato representa uno de los recursos más valiosos en el mundo para negocios, gobiernos y particulares. La toma de decisiones basada en datos está presente en casi todos los ámbitos de la sociedad, desde la medicina predictiva hasta la publicidad personalizada. Debido a esto, la habilidad para determinar el valor de un dato se ha vuelto indispensable. Es aquí donde entra en juego la valoración de datos.

Cuando nos referimos al valor de un dato, es esencial comprender que dicho valor no es unidimensional. Un dato puede ser valorado desde diferentes perspectivas y categorizado basándose en diversas cualidades:

1. **Valor Intrínsico vs. Extrínseco**:
   - *Valor intrínseco*: se refiere al valor inherente al propio dato, basado en su precisión y calidad. Este valor es independiente del uso que se le dé al dato.
   - *Valor extrínseco*: se corresponde al valor que se le atribuye al dato en función de su uso. Depende, pues, del contexto en el que se use el dato.
2. **Valor Directo vs. Indirecto**:
   - *Valor directo*: alude al beneficio inmediato que se obtiene de un dato, como podría ser al venderlo.
   - *Valor indirecto*: se refiere al beneficio derivado del uso estratégico del dato.

En este trabajo nos centraremos en estudiar el valor extrínseco e indirecto de los datos. Esta investigación nos permitirá, posteriormente, determinar el valor directo de los datos y detectar posibles problemas en su valor intrínseco.

A pesar de que la valoración de datos es un concepto multifacético que depende de varios factores, nos ajustaremos al enfoque propuesto en [15, 18], el cual se compone de tres elementos esenciales:

1. Denominaremos N al *conjunto prefijado de datos de entrenamiento*, siendo $N = \{(x_i, y_i)\}_1^n$. Aquí, x_i hace referencia a las características del dato i-ésimo, e y_i a su categoría en problemas de clasificación o su valor en problemas de regresión.
2. El *algoritmo de aprendizaje* $\mathcal{A}$, será tratado como una caja negra que toma un conjunto de entrenamiento N y genera un predictor f.
3. La *función de utilidad* v es una aplicación que asigna a cada subconjunto de N un valor, reflejando la utilidad de ese subconjunto. Para problemas de clasificación, la opción común para v es la precisión del modelo entrenado con el subconjunto dado, es decir $v(S) = acc(\mathcal{A}(S))$. Sin pérdida de generalidad asumiremos a lo largo del documento que $v(S) \in [0, 1]$ para cualquier $S \subseteq N$.

Por lo tanto, podemos entender la valoración de datos como el proceso de asignar un valor a cada dato del conjunto N, reflejando su contribución en el entrenamiento del modelo. Cada uno de estos valores estará determinado por N, $\mathcal{A}$ y v, pero por simplicidad lo expresaremos como $\phi(i; v)$. A estas puntuaciones se les denomina *data values*.

Los distintos enfoques que seguiremos para calcular estos *data values* se desarrollan a continuación.

LOO Error

El método más sencillo para valorar datos consiste en medir la contribución de un punto individual al desempeño global del conjunto de entrenamiento:

$$\phi_{loo}(i; v) = v(N) - v(N \setminus \{i\}).$$

Este método es conocido como *leave-one-out* (LOO). Para calcular el valor exacto de los valores LOO para un conjunto de entrenamiento de tamaño N, sería necesario reentrenar el modelo n veces. Este procedimiento resulta poco práctico cuando el tamaño del conjunto de datos es considerablemente grande [16].

Data Shapley

En la sección 2.1.1 se presentó el valor de Shapley, y en [2], se propone el desarrollo de un método para lograr una valoración equitativa: *Data Shapley* . El nombre se debe al hecho de que las condiciones de la valoración equitativa coinciden con los axiomas del valor de Shapley. La fórmula de *Data Shapley* es la misma que la vista en la ecuación 2.1

Beta Shapley

En [18] proponen utilizar la función Beta con parámetros positivos (α, β) para definir un caso particular de semivalor, con la siguiente función de peso:

$$\begin{aligned} w_{\alpha,\beta}(j) :&= n \int_0^1 t^{j-1}(1-t)^{n-j} \frac{t^{\beta-1}(1-t)^{\alpha-1}}{Beta(\alpha,\beta)} dt \\ &= n \frac{Beta(j+\beta-1, n-j+\alpha)}{Beta(\alpha,\beta)}. \end{aligned}$$

Donde $Beta(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ es la función Beta, y $\Gamma(\circ)$ es la función Gamma. Utilizando algunas de las propiedades vistas en el Apéndice A podemos simplificar la expresión anterior a:

$$w_{\alpha,\beta}(j) = n \frac{\prod_{k=1}^{j-1}(\beta+k-1) \prod_{k=1}^{n-j}(\alpha+k-1)}{\prod_{k=1}^{n-1}(\alpha+\beta+k-1)}.$$

Con esto, el nuevo semivalor propuesto, al que llamaremos Beta(α, β)-Shapley, queda definido como:

$$\phi_{Beta}(i; v) := \frac{1}{n} \sum_{j=1}^{n} w_{\alpha,\beta}(j) \sum_{\substack{S \subseteq N \setminus \{i\} \\ |S|=j-1}} (v(S \cup \{i\}) - v(S)).$$

Los hiperparámetros (α, β) determinan la distribución de pesos para cada elemento de N. Por ejemplo, cuando $\alpha = \beta = 1$ tendremos, $w_{1,1}(j) = \binom{n-1}{j-1}^{-1}$ para todo $j \in N$, dando el peso uniforme en las contribuciones marginales, es decir, $Beta(1, 1)$-Shapley es exactamente el valor de Shapley de los datos originales. Cuando $\alpha \geq \beta = 1$, el peso normalizado asigna grandes pesos a conjuntos de pequeña cardinalidad y elimina el ruido de los conjuntos de cardinalidad mayor. Por el contrario, $Beta(1, \beta)$ pone más pesos en la gran cardinalidad y se acerca a LOO a medida que β aumenta.

Data Banzhaf

Se denomina *Data Banzhaf* ($\phi_{Banzhaf}$) a la aplicación del valor de Banzhaf (Ecuación 2.2) en el contexto de valoración de datos. Este semivalor presenta una serie de ventajas que lo hacen especialmente interesante. Dichas propiedades serán estudiadas en detalle en secciones siguientes.

2.3. Midiendo la robustez

En diversas situaciones, como la selección de datos, el orden de los *data values* es lo que aporta valor [18]. Un ejemplo podría ser el filtrado de datos de baja calidad. El escenario ideal sería aquel en el que, incluso estando perturbada la función de utilidad, se preserva el mismo orden de *data values*.

En este contexto, la robustez alude a la resistencia de los métodos de valoración de datos ante perturbaciones o ruido. Del mismo modo que un modelo de aprendizaje robusto debería resistir entradas ruidosas, un método de valoración de datos robusto debería conservar el orden de los *data values* a pesar del ruido intrínseco de los algoritmos de aprendizaje automático.

Ahora, vamos a establecer los conceptos requeridos para formalizar y medir la robustez. Recordemos que un semivalor está determinado por su función de peso w. Así, definimos la diferencia escalada como:

Definición 2.3.1. Dados $i, j \in N$, la diferencia escalada entre los *data values* $\phi(i; v)$ y $\phi(j; v)$ se define como:

$$\begin{aligned} D_{i,j}(v, w) &:= n(\phi(i; v) - \phi(j; v)) \\ &= \sum_{k=1}^{n-1} (w(k) + w(k+1)) \binom{n-2}{k-1} \Delta_{i,j}^{k}(v). \end{aligned}$$

Donde $\Delta_{i,j}^{k}(v) := \binom{n-2}{k-1}^{-1} \sum_{\substack{S \subseteq N \setminus \{i,j\} \\ |S|=k-1}} (v(S \cup \{i\}) - v(S \cup \{j\}))$ se denomina *distinguibilidad promedio entre i y j* en subconjuntos de tamaño k usando una función de utilidad sin ruido v.

Sea ahora $\hat{v}$ un estimador de v. Es fácil ver que $\hat{v}$ y v generarán diferentes *data values* para un par de puntos i y j sii $D_{i,j}(v, w) D_{i,j}(\hat{v}, w) \leq 0$.

Se podría pensar inicialmente en definir la robustez de un semivalor como la menor cantidad de ruido $||\hat{v} - v||$ que alteraría el orden de los *data values*. Sin embargo, una definición así dependería de la función de utilidad original v. Si la función original v no es capaz de diferenciar dos puntos i y j ($\Delta_{i,j}^{k}(v) \simeq 0 \; \forall k = 1, \ldots, n-1$), entonces $D_{i,j}(v, w)$ será casi 0, y cualquier mínima perturbación podría modificar el orden entre $\phi(i; v)$ y $\phi(j; v)$.

Por ello, para definir de formar razonable la robustez de un semivalor, debemos considerar solo las funciones de utilidad que sean capaces de distinguir entre i y j.

Definición 2.3.2. Diremos que un par de puntos (i, j) son τ-distinguibles por la función de utilidad v si, y solo si, $\Delta_{i,j}^{k}(v) \geq \tau$ para todo $k \in \{1, \ldots, n-1\}$.

Sea ahora $\mathcal{V}_{i,j}^{k}$ el conjunto de todas las funciones de utilidad v que son capaces de τ-distinguir i y j. Usando la definción anterior, podemos caracterizar la robustez de un semivalor mediante su *safety margin*, que representa la menor cantidad de ruido $||\hat{v} - v||$ que, al añadirse, invertiría el orden de los *data values* de al menos un par de puntos (i, j), para al menos una función de utilidad $v \in \mathcal{V}_{i,j}^{k}$.

Definición 2.3.3. Dado $\tau > 0$, definimos el *safety margin* de un semivalor para un par de puntos $i, j \in N$ como:

$$\text{Safe}_{i,j}(\tau; w) := \min_{v \in \mathcal{V}_{i,j}^{(\tau)}} \min_{\hat{v} \in V} ||\hat{v} - v||.$$

Donde $V = \{\hat{v} : D_{i,j}(v; w) D_{i,j}(\hat{v}; w) \leq 0\}$.

El *safety margin* de un semivalor es:

$$\text{Safe}(\tau; w) := \min_{i,j \in N, i \neq j} \text{Safe}_{i,j}(\tau; w).$$

2.3.1. Robustez del valor de Banzhaf

Los resultados aquí mostrados pertenecen a la sección 4 de [1].

Teorema 2.3.4. *Para cualquier $\tau > 0$, el valor de Banzhaf alcanza el mayor safety margin*

$$\textit{Safe}(\tau; w_{\textit{Banzhaf}}) = \frac{\tau}{2^{\frac{n}{2}-1}}.$$

de entre todos los semivalores.

La demostración se puede consultar en el Apéndice C de [1].

Intuitivamente, este resultado es consecuencia de cómo los semivalores asignan diferentes pesos en función del tamaño de los subconjuntos evaluados. Así, es posible construir una perturbación de la función de utilidad que maximice la influencia sobre el semivalor correspondiente, introduciendo ruido en los subconjuntos con mayor peso asignado. De ahí que la estrategia óptima para robustecer sea asignar pesos uniformes a todos los subconjuntos, tal como lo hace el valor de Banzhaf.

Además, se puede demostrar que el valor de Banzhaf es el semivalor más robusto en el sentido de que el ruido de la utilidad afecta mínimamente a los cambios en los *data values*. En concreto, el valor de Banzhaf alcanza la menor constante de Lipschitz L tal que $||\phi(v) - \phi(\hat{v})|| \leq L||v - \hat{v}||$, para todos los posibles pares de funciones de utilidad v y $\hat{v}$.

Teorema 2.3.5. *El valor de Banzhaf, con $w_{\textit{Banzhaf}}(k) = \frac{n}{2^{n-1}}$, logra la menor constante de Lipschitz, $L = \frac{1}{2^{\frac{n}{2}-1}}$ de entre todos los semivalores.*

La demostración se puede consultar en el Apéndice C.4 de [1].

2.3.2. Estimación eficiente

Dado que es prácticamente imposible calcular los *data values* exactos en métodos de valoración de datos basados en semivalores, debido a la necesidad de un número exponencial de evaluaciones de la función de utilidad, se debe recurrir a

métodos de aproximación. A continuación, introducimos el concepto de error asociado a un estimador, el cual nos será de utilidad a la hora de comparar distintos estimadores.

Definición 2.3.6. Un estimador de un semivalor $\hat{\phi}$ es una (ϵ, δ)-aproximación del semivalor ϕ en norma l_p si, y solo si,

$$P_{\hat{\phi}}[||\hat{\phi} - \phi||_p \leq \epsilon] \geq 1 - \delta.$$

Donde la aleatoriedad se da en la construcción del estimador.

Estimador Simple de Montecarlo

El valor de Banzhaf (Ecuación 2.2) puede ser reformulado como:

$$\phi_{Banzhaf}(i; v) = \mathbb{E}_{S \sim \text{Unif}(2^{N \setminus \{i\}})}[v(S \cup \{i\}) - v(S)].$$

Donde $\mathbb{E}$ denota la esperanza.

Apartir de esto, un método de Montecarlo directo para estimar $\phi_{Banzhaf}(i; v)$ consistiría en generar muestras uniformes de $\mathcal{S}_i \subset 2^{N \setminus \{i\}}$ y calcular:

$$\hat{\phi}_{MC}(i; v) = \frac{1}{|\mathcal{S}_i|} \sum_{S \in \mathcal{S}_i} [v(S_j \cup \{i\}) - v(S_j)]. \tag{2.3}$$

Al repetir este proceso para cada $i \in N$, obtendremos el estimador $\hat{\phi}_{MC} = [\hat{\phi}_{MC}(1), \ldots, \hat{\phi}_{MC}(n)]$.

Teorema 2.3.7. *El estimador de Montecarlo simple $\hat{\phi}_{MC}$ es una (ϵ, δ)-aproximación de $\phi_{Banzhaf}$ en norma l_p con $\mathcal{O}(\frac{n^2}{\epsilon^2} \log(\frac{n}{\delta}))$ evaluaciones de v, y $\mathcal{O}(\frac{n}{\epsilon^2} \log \frac{n}{\delta})$ evaluaciones de v en la norma l_∞.*

La demostración se puede consultar en el Apéndice C.1.2 de [1].

Estimador de máxima reutilización

El método anterior resulta poco eficiente, dado que cada muestra $S \in \mathcal{S}_i$ generada solo contribuye a la estimación de $\hat{\phi}_{Banzhaf}(i; v)$ y esto introduce un factor de n en la complejidad, ya que es necesario generar un mismo número de muestras para cada dato.

En este contexto surge el concepto de estimador de máxima reutilización (MSR) [1]. La idea es explotar la linealidad de la esperanza, de forma que partiendo de 2.3 llegamos a:

$$\phi_{Banzhaf}(i; v) = \mathbb{E}_{S \sim \text{Unif}(2^{N \setminus \{i\}})}[v(S \cup \{i\})] - \mathbb{E}_{S \sim \text{Unif}(2^{N \setminus \{i\}})}[v(S)]. \tag{2.4}$$

Tomemos como ejemplo un conjunto de m muestras $\mathcal{S} = \{S_1, \ldots, S_m\}$ generado de manera uniforme. Para cada $i \in N$, podemos clasificar las muestras de $\mathcal{S}$ en dos categorías:

- $\mathcal{S}_{\ni i}$: el conjunto de muestras que contienen el dato i, es decir, $\mathcal{S}_{\ni i} = \{S \in \mathcal{S} : i \in S\}$.
- $\mathcal{S}_{\not\ni i}$: el conjunto de muestras que no contienen el dato i, esto es, $\mathcal{S}_{\not\ni i} = \{S \in \mathcal{S} : i \notin S\}$.

Así, para cada jugador, diferenciamos entre las muestras que incluyen a dicho jugador y las que no. Utilizando esta clasificación y la ecuación 2.4, podemos estimar $\phi_{Banzhaf}(i; v)$ de la siguiente manera:

$$\hat{\phi}_{MSR}(i; v) = \frac{1}{|\mathcal{S}_{\ni i}|} \sum_{S \in \mathcal{S}_{\ni i}} v(S) - \frac{1}{|\mathcal{S}_{\not\ni i}|} \sum_{S \in \mathcal{S}_{\not\ni i}} v(S).$$

Este estimador se conoce como *estimador de máxima reutilización* (MSR).

Teorema 2.3.8. *$\hat{\phi}_{MSR}$ es una (ϵ, δ)-aproximación de $\phi_{Banzhaf}$ en norma l_p con $\mathcal{O}(\frac{n}{\epsilon^2} \log(\frac{n}{\delta}))$ evaluaciones de v, y $\mathcal{O}(\frac{1}{\epsilon^2} \log \frac{n}{\delta})$ evaluaciones de v en la norma l_∞.*

La demostración se puede consultar en el Apéndice C.1.2 de [1].

Una de las grandes ventajas que presenta el valor de Banzhaf respecto a los demás semivalores es que se trata del único semivalor que permite la implementación del algoritmo MSR, como puede verse en el Apéndice C.2 de [1].

El siguiente teorema nos da una estimación de la bondad del MSR frente al resto de estimadores posibles.

Teorema 2.3.9. *Todo estimador aleatorio del valor de Banzhaf que sea una (ϵ, δ)-aproximación en norma l_∞ con $\delta \in (0, \frac{1}{2})$ requiere al menos $\Omega(\frac{1}{\epsilon})$.*

La demostración detallada se puede consultar en el Apéndice C.1.3 de [1].

Como hemos visto anteriormente el algoritmo MSR presenta una complejidad de $\mathcal{O}(\frac{1}{\epsilon^2} \log(\frac{n}{\delta}))$ en la norma l_∞. Esto quiere decir que se aleja de la optimalidad en un factor de $\mathcal{O}(\frac{1}{\epsilon} \log(\frac{n}{\delta}))$.

CAPÍTULO 3

Análisis Experimental

En este capítulo buscamos comprender cómo los valores de datos pueden impactar el entrenamiento de modelos y, en consecuencia, las decisiones derivadas de estos. A lo largo del capítulo, ofrecemos detalles sobre los experimentos llevados a cabo, los *datasets* empleados y las herramientas esenciales que respaldaron nuestra investigación. Todos el código desarrollado en este trabajo puede ser consultado en Fran-AM/TFM.

3.1. Metodología

3.1.1. Datasets

Los *datasets* utilizados en los experimentos están listados en la tabla 3.1. Estos conjuntos son comúnmente empleados en la literatura como *benchmarks* para estudios similares al nuestro [1].

Las características de todos estos son similares. Contienen datos etiquetados, con dos clases en la variable objetivo. En cuanto al preprocesado, se llevó a cabo un submuestreo en todos ellos para equilibrar el número de muestras de cada una de las clases. Dicho preprocesado se puede consultar en la figura B.1 del Apéndice B

Dataset	Source
Click	https://www.openml.org/d/1218
Phoneme	https://www.openml.org/d/1489
Wind	https://www.openml.org/d/847
CPU	https://www.openml.org/d/761
2DPlanes	https://www.openml.org/d/727

TABLA 3.1: Datasets usados en los experimentos

3.1.2. Herramientas

Entre las herramientas utilizadas, es relevante destacar pyDVL, una librería de appliedAI, y DVC, como instrumento de control de versiones específica para proyectos de ciencia de datos.

pyDVL

pyDVL es una librería, actualmente en su versión `0.7.0`, que engloba diversos algoritmos enfocados al cálculo de *data values*. La mayoría de los métodos que implementa están basados en teoría de juegos cooperativos. Con *pyDVL*, es posible calcular de manera sencilla y automática los *data values* de un *dataset* determinado, empleando diferentes métodos de valoración, técnicas de muestreo y estimadores.

El proceso de cálculo de valores de datos en *pyDVL* se divide en tres pasos esenciales:

1. Construcción del dataset de pyDVL a partir de tus datos.
2. Creación de la utilidad, un concepto abstracto que interrelaciona el modelo a usar, el dataset y la métrica de error.
3. Cálculo de los valores de datos utilizando el método seleccionado.

Se dispone de un ejemplo de uso en la sección B.2 del Apéndice B

DVC

DVC, acrónimo de Data Version Control, es una herramienta concebida para proyectos de ciencia de datos, desarrollada con el propósito de ayudar a equipos de ML en la gestión de grandes volúmenes de datos, asegurar la reproducibilidad de los proyectos y fortalecer el trabajo colaborativo. *DVC* es compatible con cualquier terminal y también puede ser invocado como una biblioteca de Python. Su objetivo principal es brindar una experiencia al estilo Git para estructurar datos, modelos y experimentos en proyectos de Ciencia de Datos y Aprendizaje Automático. Para información más detallada, consultar la Documentación de oficial.

3.2. Experimentos

3.2.1. Detección de datos mal etiquetados

Investigamos la habilidad de distintos métodos de valoración de datos para identificar puntos mal etiquetados en escenarios con funciones de utilidad ruidosas. Esta capacidad de detectar y corregir puntos mal etiquetados es esencial para mejorar el rendimiento de los modelos de ML y reducir el tiempo de entrenamiento de los mismos.

Al igual que en [1], utilizaremos como modelo a lo largo de todo el experimento un perceptrón multicapa, con una única capa oculta compuesta por 100 neuronas. La función de activación será ReLU, y el *learning rate* inicial se establece en 10^{-2}. Se usará el optimizador Adam durante el proceso de entrenamiento. El tamaño de los *batches* es de 32 para todos los *datasets*. Dado que el proceso de valoración de datos es muy costoso desde el punto de vista computacional, trabajaremos con subconjuntos de los *datasets* originales. De cada conjunto, seleccionaremos 200 muestas al azar para llevar a cabo el experimento. Aunque nuestra intención era usar el estimador MSR, no fue posible debido a falta de tiempo para implementarlo y testearlo correctamente. Por lo que utilizaremos estimadores basados en muestreo permutacional, recomendado en la documentación de *pyDVL*.

Para la implementación de este experimento generamos muestras con etiquetas intercambiadas invirtiendo las etiquetas del 10 % de los puntos de datos de entrenamiento. A la hora de la evaluación del rendimiento de cada método, un punto se considera incorrectamente etiquetado si su *data value* se encuentra por debajo del percentil 10 en relación con todas las demás puntuaciones. Los resultados obtenidos pueden verse en la tabla 3.2.

En la tabla 3.2 se muestran los resultados obtenidos. Se puede observar que el método $Beta(4,1)$ es el que presenta los mejores resultados. Este método otorga mayor peso a conjuntos de menor cardinalidad al calcular los *data values*. Su eficacia ya ha sido evidenciada en otros trabajos, como se refleja en [1] y [18]. De hecho, el método óptimo para cada uno de los *datasets* concuerda con los resultados presentados en [1]. Considerando el factor de mejora y teniendo en cuenta que un detector aleatorio solo conseguiría un $F1 = 0{,}1$, se puede observar que, a excepción de LOO, todos los métodos logran, en promedio, triplicar el rendimiento de un detector aleatorio. El método $Beta(4,1)$ llega a multiplicar por 5 el rendimiento de un detector aleatorio.

3.2.2. Entrenamiento ponderado

Estudiamos ahora cómo se puede aplicar un sistema de ponderación basado en *data values* para realizar una suerte de submuestreo de los datos.

Utilizaremos como modelo la regresión logística con descenso del gradiente estocástico (SGDClassifier) de la librería scikit-learn. Adoptaremos regularización

Dataset	Data Banzhaf	LOO	Beta(16,1)	Beta(4,1)	Data Shapley	Beta(1,4)	Beta(1,16)
Click	0.25	0.20	0.3	0.20	0.10	0.3	**0.35**
Phoneme	0.20	0.20	0.35	0.55	**0.6**	0.25	0.20
Wind	0.25	0.05	0.40	**0.45**	**0.45**	0.40	0.25
CPU	0.5	0.05	0.5	**0.65**	0.45	0.6	0.35
2DPlanes	0.45	0.15	0.55	**0.65**	0.45	0.55	0.35

TABLA 3.2: Comparación de los valores de $F1$ de los modelos relacionada con su habilidad para detectar datos mal etiquetados. Se comparan los siete métodos de valoración en los cinco datasets de clasificación.

estándar L_2 con un coeficiente $\alpha = 10^{-4}$. Estableceremos un máximo de 10^3 épocas de entrenamiento y una tolerancia de 10^{-3} para el criterio de parada.

Optamos por la regresión logística debido principalmente a la limitada capacidad de cómputo para entrenar modelos más sofisticados. El empleo del método de descenso del gradiente estocástico nos facilitará la incorporación de ruido a la función de utilidad. Al igual que en el experimento anterior, trabajaremos con subconjuntos de 200 puntos y destinaremos el resto de puntos para el conjunto de test, con el que hemos calculado los valores de $F1$ que se muestran en la tabla 3.3. Los valores se corresponden con la media tras 5 repeteciones del experimento.

Para la ponderación, normalizaremos los *data values* al intervalo $[0, 1]$. Durante el proceso de entrenamiento, cada muestra se multiplicará por su respectivo peso, lo que significa que los puntos con mayores valores ejercerán una influencia más significativa en la regresión logística. Finalmente, entrenaremos los clasificadores con los conjuntos de entrenamiento ponderados y evaluaremos su precisión en los conjuntos de test. Adoptaremos la métrica $F1$ como criterio de evaluación, en línea con lo realizado en [1].

Podemos observar que *Data Banzhaf* no presenta el mejor desempeño. No obstante, este hecho no es sorprendente; en el experimento original, los valores de $F1$ obtenidos eran semejantes entre varios de los métodos estudiados. En nuestro experimento, existen diversos factores que difieren del original, tales como el tamaño de los conjuntos, el empleo de un modelo diferente y no usar MSR como estimador.

Al analizar el resultado global y dejando de lado casos específicos, como el observado con el *dataset 2DPlanes*, se constata que el entrenamiento ponderado según los *data values* obtenidos mejora el rendimiento del modelo. Incluso el LOO suele ofrecer una mejora, aunque sea leve.

Dataset	Data Banzhaf	LOO	Beta(16,1)	Beta(4,1)	Data Shapley	Beta(1,4)	Beta(1,16)	Uniform
Click	0.599(0.020)	0.558(0.053)	0.543(0.056)	**0.608(0.014)**	0.541(0.029)	0.564(0.072)	0.585(0.045)	0.555
Phoneme	0.665(0.082)	0.657(0.088)	**0.728(0.013)**	0.660(0.070)	0.718(0.035)	0.698(0.085)	0.616(0.162)	0.635
Wind	0.815(0.017)	0.813(0.033)	0.822(0.007)	**0.832(0.019)**	0.824(0.009)	0.820(0.008)	0.808(0.015)	0.804
CPU	0.893(0.006)	0.899(0.006)	0.893(0.010)	0.898(0.005)	**0.903(0.001)**	0.890(0.010)	0.894(0.015)	0.893
2DPlanes	0.819(0.01)	0.818(0.012)	0.813(0.010)	0.811(0.019)	0.810(0.012)	**0.820(0.016)**	0.816(0.015)	0.820

TABLA 3.3: Comparación de los valores de $F1$ de los modelos entrenados con pesos ponderados. Se comparan los siete métodos de valoración en los cinco datasets de clasificación. Se muestra la media y la desviación estándar en formato 'avg(std)'.

CAPÍTULO 4

Conclusiones y Trabajo Futuro

Conclusiones

Como se discutió en el capítulo anterior, los resultados obtenidos concuerdan con lo anticipado tras revisar la literatura. No obstante, se esperaban mejores resultados por parte de *Data Banzhaf*. Este ligero decremento en el rendimiento del método seguramente esté influenciado por dos factores principales: el primero es la falta de capacidad de cómputo, que impidió utilizar tamaños de muestra mayores y obtener un mayor número de muestras para construir los estimadores. El segundo factor es la ausencia de la implementación del estimador MSR, que ha demostrado ser más eficaz en términos de convergencia. Estas limitaciones afectan a los resultados de todos los métodos en general, pero en especial al *Data Banzhaf*, siendo una de sus principales ventajas respecto al resto de métodos basados en semivalores la posibilidad de utilizar el estimador MSR.

Considerando una perspectiva general, se observa que los métodos de valoración de datos proporcionan buenos resultados en las tareas evaluadas. Sin embargo, aún no se consideran una solución realista debido a la alta carga computacional que implican.

Trabajo Futuro

Respecto al trabajo futuro, se identifican varias áreas de mejora potencial para el proyecto:

- Implementación del estimador MSR y posterior integración en *pyDVL*.
- Uso de todos los *datasets* citados en [1]. Aunque ya se han desarrollado funciones de lectura y preprocesado para estos conjuntos de datos, no se han empleado en este trabajo. Dichas funciones están disponibles en Fran-AM/TFM.

- Integración de los modelos de *pyTorch* mencionados en [1] en *pyDVL*. Para lograr esto, será necesario implementar el protocolo SupervisedModel, en todos estos modelos.
- Desarrollo de test unitarios para la mejora de la calidad del código.

Bibliografía.

[1] Jiachen T. Wang y Ruoxi Jia. «Data Banzhaf: A Robust Data Valuation Framework for Machine Learning». Proceedings of Machine Learning Research 206, 2023. Ed. por Francisco J. R. Ruiz, Jennifer G. Dy y Jan-Willem van de Meent, págs. 6388-6421.

[2] Lloyd S Shapley. «A Value for n-Person Games», 1952. Ed. por Harold W. Kuhn y Albert W. Tucker, págs. 307-317.

[3] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[4] Shaheen S. Fatima, Michael Wooldridge y Nicholas R. Jennings. «A linear approximation method for the Shapley value». *Artificial Intelligence* 172.14, 2008, págs. 1673-1699.

[5] Stefano Moretti, Vito Fragnelli, Fioravante Patrone y Stefano Bonassi. «Using coalitional games on biological networks to measure centrality and power of genes». *Bioinformatics (Oxford, England)* 26, 2010, págs. 2721-30.

[6] Pradeep Dubey y Robert J. Weber. *Probabilistic Values for Games*. Cowles Foundation Discussion Papers 471. Cowles Foundation for Research in Economics, Yale University, 1977.

[7] Pradeep Dubey, Abraham Neyman y Robert James Weber. «Value Theory without Efficiency». *Mathematics of Operations Research* 6.1, 1981, págs. 122-128.

[8] J.F. Banzhaf. «Weighted voting doesn't work: A mathematical analysis». *Rutgers Law Review* 19.2, 1965, págs. 317-343.

[9] Irwin Mann y Lloyd S. Shapley. «Values of Large Games, IV: Evaluating the Electoral College by Montecarlo Techniques», 1960.

[10] Sasan Maleki. «Addressing the computational issues of the Shapley value with applications in the smart grid». Tesis doct. University of Southampton, ago. de 2015.

[11] Ian C Covert, Scott M Lundberg y Su-In Lee. «Shapley feature utility». *IEEE Transactions on Information Theory*, 2019.

[12] Javier Castro, Daniel Gómez y Juan Tejada. «Polynomial calculation of the Shapley value based on sampling». *Computers & Operations Research* 36.5, 2009. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X), págs. 1726-1730.

[13] Shay Cohen, Eytan Ruppin y Gideon Dror. «Feature selection based on the shapley value». *other words* 1.98Eqr, 2005, pág. 155.

[14] Scott M. Lundberg y Su-In Lee. «A Unified Approach to Interpreting Model Predictions», 2017. Ed. por Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan y Roman Garnett, págs. 4765-4774.

[15] Amirata Ghorbani y James Y. Zou. «Data Shapley: Equitable Valuation of Data for Machine Learning». Proceedings of Machine Learning Research 97, 2019. Ed. por Kamalika Chaudhuri y Ruslan Salakhutdinov, págs. 2242-2251.

[16] Ruoxi Jia et al. «Scalability vs. Utility: Do We Have to Sacrifice One for the Other in Data Importance Quantification?», 2019.

[17] Ruoxi Jia et al. «Efficient Task-Specific Data Valuation for Nearest Neighbor Algorithms». *Proc. VLDB Endow.* 12.11, 2019, págs. 1610-1623.

[18] Yongchan Kwon y James Zou. «Beta Shapley: a Unified and Noise-reduced Data Valuation Framework for Machine Learning». Proceedings of Machine Learning Research 151, 2022. Ed. por Gustau Camps-Valls, Francisco J. R. Ruiz e Isabel Valera, págs. 8780-8802.

[19] Sebastian Shenghong Tay, Xinyi Xu, Chuan Sheng Foo y Bryan Kian Hsiang Low. «Incentivizing Collaboration in Machine Learning via Synthetic Data Rewards», 2022, págs. 9448-9456.

[20] Xinyi Xu, Zhaoxuan Wu, Chuan Sheng Foo y Bryan Kian Hsiang Low. «Validation Free and Replication Robust Volume-based Data Valuation». 34, 2021. Ed. por M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang y J. Wortman Vaughan, págs. 10837-10848.

[21] Mohammad Mohammadi Amiri, Frederic Berdoz y Ramesh Raskar. «Fundamentals of Task-Agnostic Data Valuation», 2023. Ed. por Brian Williams, Yiling Chen y Jennifer Neville, págs. 9226-9234.

[22] Jinsung Yoon, Sercan Arik y Tomas Pfister. «Data Valuation using Reinforcement Learning». Proceedings of Machine Learning Research 119, 2020. Ed. por Hal Daumé III y Aarti Singh, págs. 10842-10851.

Índice alfabético

Símbolos

C

D

E

F

G

J

L

M

S

T

V

APÉNDICE A

Funciones Gamma y Beta

A.1. Función Gamma

Definición A.1.1. Llamaremos función Gamma (Γ) a la función definida en el intervalo $(0, +\infty)$ por

$$\Gamma(p) = \int_0^{+\infty} e^{-t} t^{p-1}\, dt. \tag{A.1}$$

Teorema A.1.2. *Para todo $p > 0$ se tiene que $\Gamma(p+1) = p\Gamma(p)$.*
En particular, $\Gamma(n+1) = n!$ para todo $n \in \mathbb{N}$.

Demostración. Integrando por partes tenemos

$$\Gamma(p+1) = \int_0^{+\infty} e^{-t} t^p\, dt = \left[-e^{-t} t^p\right]_0^{+\infty} + p \cdot \int_0^{+\infty} e^{-t} t^{p-1}\, dt.$$

Como $\left[-e^{-t} t^p\right]_0^{+\infty} = 0$ al ser $p > 0$, tenemos que $\Gamma(p+1) = p\Gamma(p)$.

Para la segunda afirmación, sabiendo que $\Gamma(1) = \left[-e^{-t}\right]_0^{+\infty} = 1$, tendremos

$$\Gamma(n+1) = n\Gamma(n) = n(n-1)\Gamma(n-1) = \cdots = n!.$$

□

Teorema A.1.3. *Si $a \in \mathbb{C}$, tal que $Re(a) > 0$, y $p > 0$, entonces:*

$$\int_0^{+\infty} e^{-at} t^{p-1}\, dt = \frac{\Gamma(p)}{a^p}.$$

Demostración. Realizando el cambio de variable $t = au$ en la integral de la función Γ, tenemos:

$$\Gamma(p) = \int_0^{+\infty} e^{-t} t^{p-1}\, dt = \int_0^{+\infty} e^{-au} (au)^{p-1} \cdot a\, du = a^p \int_0^{+\infty} e^{-au} u^{p-1}\, du.$$

Despejando tenemos

$$\frac{\Gamma(p)}{a^p} = \int_0^{+\infty} e^{-au} u^{p-1}\, dt.$$

Esta igualdad se tiene para todo $a \in \mathbb{R}$, ahora bien, si $a \in \mathbb{C}$, con $Re(a) > 0$, tenemos que las funciones complejas a cada lado de la igualdad son analíticas en a y coinciden en el eje real positivo, por lo que coinciden en todos los puntos en los que sean holomorfas, es decir, al menos en el semiplano real positivo. □

A.2. Función Beta

Definición A.2.1. Llamaremos función *Beta* a la función definida para todo par de puntos $p, q > 0$ por la integral:

$$Beta(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}\, dt.$$

Teorema A.2.2. *La función Beta es simétrica en sus dos variables, es decir,* $Beta(p,q) = Beta(q,p)$.

Demostración. Realizando el cambio de variable $t = 1 - u$ en la integral de la función *Beta*, tenemos:

$$Beta(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}\, dt = \int_0^1 (1-u)^{p-1}u^{q-1}\, du.$$

Intercambiando p y q tenemos:

$$\begin{aligned} Beta(p,q) &= \int_0^1 t^{p-1}(1-t)^{q-1}\, dt = -\int_1^0 (1-u)^{p-1}u^{q-1}\, du \\ &= \int_0^1 u^{q-1}(1-u)^{p-1}\, du = Beta(q,p). \end{aligned}$$

□

Teorema A.2.3. *Para todo* $p, q > 0$ *se tiene que*

$$Beta(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}.$$

Demostración. Realizamos el cambio de variables $t = \frac{v}{1+v}$ en la integral que define la función *Beta*, se tiene:

$$Beta(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}\, dt = \int_0^{+\infty} \frac{v^{p-1}}{(1+v)^{p+q}}\, dv.$$

Usando el teorema A.1.3, y tomando $a = 1 + v$ y $p + q$ como primer parámetro, tenemos:

$$\frac{1}{(1+v)^{p+q}} = \frac{1}{\Gamma(p+q)} \int_0^{+\infty} e^{-(1+v)t} t^{p+q-1} dt.$$

Sustituyendo esta última igualdad en la expresión anterior de *Beta* se tiene:

$$\begin{aligned} Beta(p,q) &= \int_0^{+\infty} \frac{v^{p-1}}{(1+v)^{p+q}} dv \\ &= \frac{1}{\Gamma(p+q)} \int_0^{+\infty} \int_0^{+\infty} v^{p-1} e^{-(1+v)t} t^{p+q-1} \, dv \, dt \\ &= \frac{1}{\Gamma(p+q)} \int_0^{+\infty} e^{-t} t^{p+q-1} \left(\int_0^{+\infty} e^{-tv} v^{p-1} dv \right) dt. \end{aligned}$$

Volviendo a usar la expresión

$$\int_0^{+\infty} e^{-at} t^{p-1} \, dt = \frac{\Gamma(p)}{a^p}, \; a, p > 0$$

se deduce que

$$\begin{aligned} Beta(p,q) &= \frac{1}{\Gamma(p+q)} \int_0^{+\infty} e^{-t} t^{p+q-1} \left(\int_0^{+\infty} e^{-tv} v^{p-1} \, dv \right) dt \\ &= \frac{1}{\Gamma(p+q)} \int_0^{+\infty} t^{p+q-1} e^{-t} \frac{\Gamma(p)}{t^p} \, dt. \end{aligned}$$

Y utilizando la expresión de la función Γ, se concluye el resultado buscado

$$\begin{aligned} Beta(p,q) &= \frac{1}{\Gamma(p+q)} \int_0^{+\infty} e^{-t} t^{p+q-1} \frac{\Gamma(p)}{t^p} \, dt \\ &= \frac{\Gamma(p)}{\Gamma(p+q)} \int_0^{+\infty} e^{-t} t^{q-1} \, dt = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}. \end{aligned}$$

□

APÉNDICE B

Código relevante

Se muestran algunos fragmentos de código que se consideran interesantes y pueden ayudar a seguir de una forma más sencilla el desarrollo de este documento.

B.1. Preprocesado de datos

```
def get_openML_data(
    dataset: str,
    n_data: int,
    flip_ratio: float = 0.0,
)->Dataset:
if dataset in ds_map:
    try:
        data_dict =
        pickle.load(open(openML_path + ds_map[dataset], 'rb'))
        data, target = data_dict['X_num'], data_dict['y']
        target = (target == 1).astype(np.int32)
        data, target = undersamp_equilibration(data, target)
    except FileNotFoundError:
        raise FileNotFoundError(f"File not found for dataset '{
   dataset}'")
else:
    raise ValueError(f"Dataset '{dataset}' not found")

x_train, x_test, y_train, y_test = train_test_split(
    data, target, train_size=n_data, random_state=42
)

# Normalization
x_mean, x_std= np.mean(x_train, 0), np.std(x_train, 0)
norm = lambda x: (x - x_mean) / np.clip(x_std, 1e-12, None)
x_train, x_test = norm(x_train), norm(x_test)

# Flip labels
if len(y_train.shape) != 1:
    raise ValueError("Expected y_train to be a 1-dimensionalarray, "
                        "but got a different shape.")

n_flip = int(n_data*flip_ratio)
y_train[:n_flip] = 1 - y_train[:n_flip]

return Dataset(
    x_train=x_train,
    y_train=y_train,
    x_test=x_test,
    y_test=y_test
)

```

FIGURA B.1: Función para la lectura y preprocesado de los datos de OpenML.

B.2. Ejemplo de uso de pyDVL

```
# Contruccion del pyDVL dataset
x_train, x_test, y_train, y_test = train_test_split(
data, target, train_size=n_data, random_state=42
)
pyDataset = Dataset(
    x_train=x_train,
    y_train=y_train,
    x_test=x_test,
    y_test=y_test
    )

# Construccion del modelo y la utilidad
model = MLPClassifier(
hidden_layer_sizes = (md_params["hidden_neurons"],),
learning_rate_init = md_params["learning_rate"],
batch_size = md_params["batch_size"],
max_iter = md_params["max_iter"],

utility = Utility(model, pyDataset, 'accuracy')

# Calculo de los valores de Shapley
values = compute_shapley_values(
    u=utility,
    mode="permutation_montecarlo",
    done=HistoryDeviation(n_steps=100, rtol=0.05),
    truncation=RelativeTruncation(utility, rtol=0.01),
    progress=True,
    n_jobs=6
)

```