Árboles de Clasificación y Regresión (CART) & Random Forest

HE2: Consultoría Económica con IA Responsable

Santiago Neira & Catalina Bernal

Universidad de los Andes
Departamento de Economía

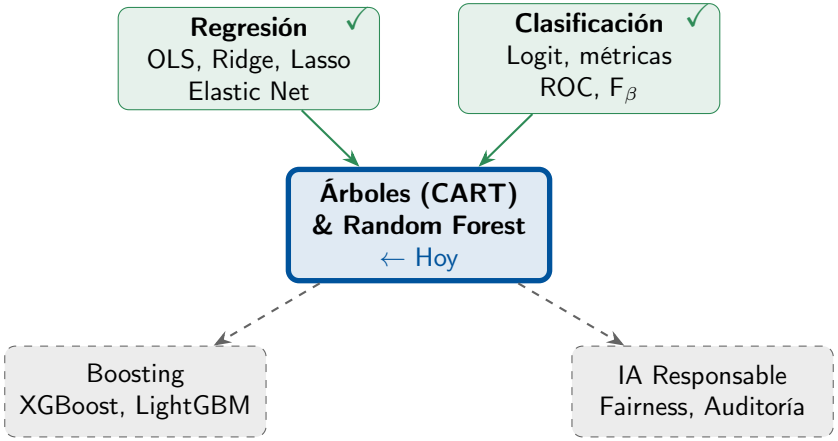
Febrero 2026

Agenda de hoy

1. Roadmap: ¿Dónde Estamos?
2. Historia y Motivación: Del CART al Credit Scoring
3. Construcción Formal del CART
4. Ejemplo Clásico: El Titanic
5. Hiperparámetros del CART
6. De un Árbol a un Bosque: Random Forest
7. Interpretabilidad Post-hoc: Feature Importance y SHAP Values

Agenda de hoy

El Camino que Hemos Recorrido



Hoy entramos a una familia de algoritmos que sirve tanto para regresión como para clasificación, y que es *state-of-the-art* en la mayoría de problemas con datos tabulares.

¿Por Qué Árboles y Ensambles?

Hasta ahora, nuestros modelos principales eran **lineales** (o lineales en transformaciones):

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Hoy veremos modelos que:

- No asumen **ninguna** forma funcional
- Capturan **interacciones** entre variables automáticamente
- Funcionan para Y continua (regresión) y Y categórica (clasificación)
- Manejan variables categóricas y numéricas sin transformación
- Son la base de los algoritmos más usados en la industria (XGBoost, LightGBM)

Agenda de hoy

Breve Historia del CART

1963: Morgan y Sonquist desarrollan AID (*Automatic Interaction Detection*) — la primera idea de particiones recursivas para análisis de encuestas.

1977–1984: Leo Breiman y Charles Stone (UC Berkeley) junto con Jerome Friedman y Richard Olshen (Stanford) desarrollan **CART** (*Classification and Regression Trees*). Su libro de 1984 es uno de los textos más influyentes en estadística aplicada.

¿Por qué fue revolucionario?

- No asume linealidad ni distribución paramétrica
- Maneja variables categóricas y continuas simultáneamente
- Produce reglas interpretables: *if-then-else*
- Captura **interacciones** entre variables automáticamente

1996: Breiman propone *Bagging*. **2001:** Breiman publica **Random Forests**.

Ref: Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.

Credit Scoring y Género: Lo que los Árboles Revelaron

La aplicación de árboles de decisión y ML en credit scoring ha revelado un resultado consistente a nivel mundial: **las mujeres tienen, en promedio, mejores tasas de repago que los hombres**.

¿Por qué los modelos lineales tradicionales no capturaban esto adecuadamente?

- Los scorecards tradicionales promedian el efecto de género *linealmente*
- Un árbol puede descubrir **interacciones**: subgrupos donde mujeres son significativamente mejores pagadoras (ej: género $\times$ tipo de empleo $\times$ zona geográfica)
- El árbol particiona la población y encuentra heterogeneidad que el modelo lineal pierde

La Paradoja: ¿Usar o No Usar Género?

Enfoque 1: "No usar género"

- Goldman Sachs / Apple Card (2019)
- Lógica: si no veo género, no discrimino
- Problema: el modelo aprende proxies de género (estado civil, ocupación, zona)
- Resultado: puede *perjudicar* a las mujeres

Enfoque 2: "Modelos diferenciados"

- CEGA/Berkeley + banco en Rep. Dominicana
- Modelo separado para mujeres
- Resultado: 93% de mujeres obtuvieron scores más altos
- Problema: en muchos países es ilegal

Tensión para IA Responsable

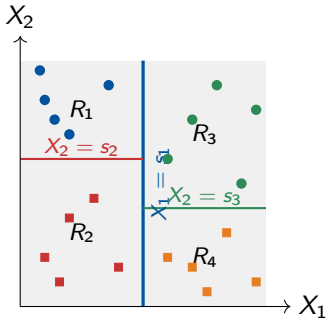
No usar información sensible no garantiza equidad. Los árboles y modelos de ML hacen esta tensión visible porque capturan heterogeneidad que los modelos lineales pierden. Esta es una discusión activa en la regulación de IA.

Ref: Financial Alliance for Women (2024). `https://financialallianceforwomen.org/news-events/gender-differentiated-credit-scoring-a-potential-game-changer-for-women/`

Agenda de hoy

La Idea Central: Particiones Recursivas

Objetivo: Dividir el espacio de features $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ en **regiones rectangulares** $R_1, R_2, \ldots, R_M$ tales que las observaciones dentro de cada región sean lo más **homogéneas** posible.



Algoritmo Greedy:
En cada paso, buscar la variable X_j y el punto de corte s que produce la mejor partición:

$$\min_{j,s} \left[C(R_1(j,s)) + C(R_2(j,s)) \right]$$

donde $C(\cdot)$ es una medida de **impureza**.

Particiones binarias:

$$R_{\text{izq}}(j,s) = \{X \mid X_j \leq s\}$$

$$R_{\text{der}}(j,s) = \{X \mid X_j > s\}$$

El algoritmo es *greedy*: toma la mejor decisión local en cada paso, sin garantía de óptimo global.

Predicción en Cada Nodo Terminal

Para clasificación, la predicción en cada hoja m es la **clase mayoritaria**:

$$\hat{y}_m = \arg\max_k \ \hat{p}_{mk} \qquad \text{donde } \hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} 1(y_i = k)$$

Para regresión, la predicción es el **promedio**:

$$\hat{y}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

El algoritmo evalúa exhaustivamente todas las variables $j = 1, \ldots, p$ y todos los posibles puntos de corte s para encontrar la partición que minimiza la impureza total.

Complejidad: $O(p \cdot N \log N)$ por nodo (ordena cada variable y prueba cada punto).

Medidas de Impureza en Clasificación

Sea $\hat{p}_{mk}$ la proporción de observaciones de clase k en el nodo m:

Error de Clasificación

$$1 - \max_k \hat{p}_{mk}$$

Intuitivo pero **no diferenciable**. No es bueno para crecer árboles.

Índice de Gini

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Mide la probabilidad de clasificar incorrectamente. **El default en scikit-learn.**

Entropía

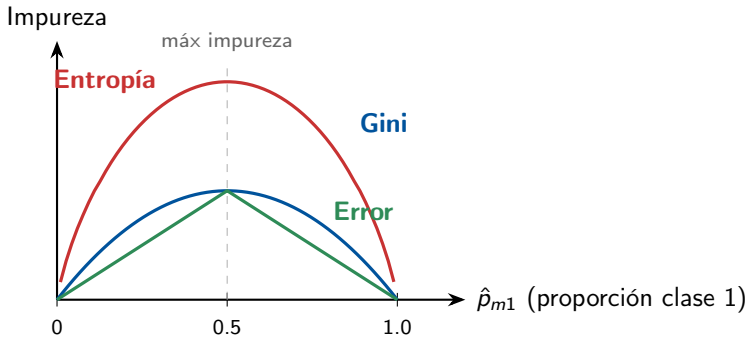
$$-\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

De teoría de la información. Similar al Gini en la práctica.

Para regresión: se usa el **MSE** como criterio:

$$C(R_m) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{y}_m)^2$$

Gini vs. Entropía vs. Error de Clasificación



Observaciones:

- Gini y Entropía son casi idénticos en la práctica. Ambos son cóncavos y diferenciables.
- El error de clasificación no es suave $\rightarrow$ no es sensible a cambios en las proporciones.
- En `scikit-learn`: `criterion='gini'` (default) o `criterion='entropy'`.

La Ganancia de Información

Al dividir un nodo t en hijos t_L y t_R, la **reducción de impureza** es:

$$\Delta I(t) = I(t) - \frac{N_{t_L}}{N_t} I(t_L) - \frac{N_{t_R}}{N_t} I(t_R)$$

El algoritmo elige la partición (j, s) que **maximiza** $\Delta I(t)$.

Intuición económica

Es como un análisis costo-beneficio: el "costo" es la impureza restante, el "beneficio" es la reducción lograda. La mejor partición tiene el mayor beneficio neto, ponderado por el tamaño de los grupos resultantes.

Elecciones binarias por tipo de variable:

- **Continua:** $X_j \leq s$? (ej: `Age` $\leq$ `6.5`?)
- **Categórica:** $X_j \in \mathcal{S}$? (ej: `Sex` $\in$ `{male}`?)
- **Ordinal:** $X_j \leq$ nivel k ? (ej: `Pclass` $\leq$ `2`?)

Para categóricas con L niveles: $2^{L-1} - 1$ particiones posibles. Con L grande, costoso.

Agenda de hoy

El Dataset del Titanic

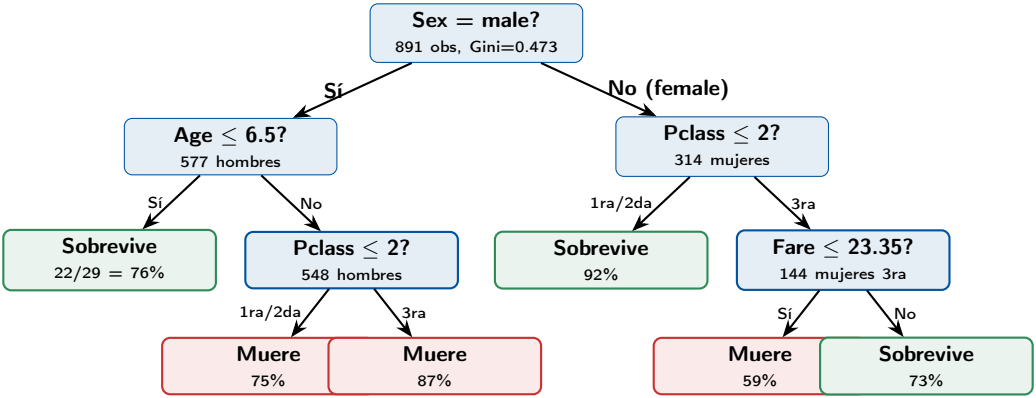
El hundimiento del RMS Titanic (1912) es el ejemplo canónico de árboles de decisión. De 2,224 personas a bordo, solo 710 sobrevivieron (32%).

Variable	**Tipo**	**Descripción**
`Survived`	Binaria	0 = No, 1 = Sí (**target**)
`Pclass`	Categórica	Clase: 1ra, 2da, 3ra
`Sex`	Binaria	male / female
`Age`	Continua	Edad en años
`SibSp`	Entera	Hermanos/cónyuge a bordo
`Parch`	Entera	Padres/hijos a bordo
`Fare`	Continua	Tarifa pagada
`Embarked`	Categórica	Puerto de embarque

Pregunta: ¿Qué factores determinaron la supervivencia?

Dataset disponible en: `https://www.kaggle.com/c/titanic`

El Árbol del Titanic



Insight: La primera partición es Sex. El famoso "mujeres y niños primero" emerge directamente de los datos sin que nadie se lo diga al modelo.

Leyendo el Árbol del Titanic

Reglas que el algoritmo descubrió:

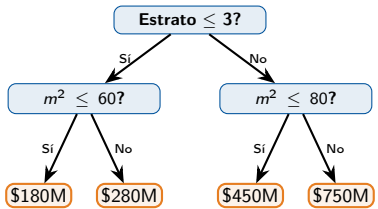
1. **Ser mujer** es el predictor más importante (mayor reducción de Gini en el primer split).
2. **Mujeres de 1ra y 2da clase** sobrevivieron al 92%. Cercanía a botes + protocolo social.
3. **Niños $\leq$ 6.5 años** (incluso varones) tenían 76% de sobrevivir.
4. **Hombres adultos de 3ra clase**: solo 13% sobrevivió.
5. **Fare** funciona como proxy de riqueza, capturando información adicional a la clase.

Valor para consultoría

Un árbol comunica hallazgos de forma que un cliente no técnico entiende inmediatamente. No necesita interpretar coeficientes, p-valores, ni log-odds. Las reglas *if-then* son accionables.

Árbol de Regresión: Ejemplo

En lugar de clasificar, predecimos un valor continuo (ej: precio de vivienda).



Diferencias con clasificación:

- Cada hoja predice un **promedio** (no una clase)
- La impureza se mide con MSE, no Gini
- La predicción es una **función escalonada**

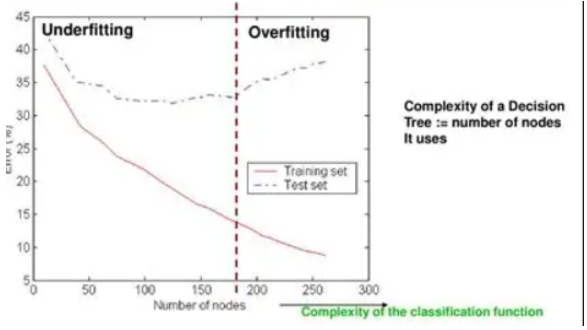
Limitación: Valores constantes por región. No puede capturar tendencias lineales dentro de una región.

Esto motiva ensambles (Random Forest, Boosting) que suavizan la función.

Agenda de hoy

El Problema: Overfitting en Árboles

Un árbol sin restricciones crece hasta que cada hoja tiene una sola observación. Esto es overfitting extremo.



Soluciones:

- **Pre-pruning:** Restringir el crecimiento (limitar profundidad, mín. observaciones por hoja, etc.)
- **Post-pruning:** Crecer el árbol completo y después podar (cost-complexity pruning, α)

Todos los Hiperparámetros del CART (`scikit-learn`)

Hiperparámetro	Default	Efecto
`max_depth`	`None`	Profundidad máxima. **El más importante.**
`min_samples_split`	2	Mínimo de obs. para dividir un nodo.
`min_samples_leaf`	1	Mínimo de obs. en cada hoja terminal.
`max_features`	`None`	N° de features a considerar por split.
`max_leaf_nodes`	`None`	Máximo número de hojas.
`min_impurity_decrease`	0.0	Solo dividir si $\Delta I \geq$ este valor.
`ccp_alpha` (α)	0.0	Poda por complejidad-costo.
`criterion`	`gini`	`gini`, `entropy`, `log_loss`.
`class_weight`	`None`	Pondera clases (útil con desbalance).

Cost-Complexity Pruning (α)

La poda por complejidad-costo (Breiman et al., 1984):

$$R_\alpha(T) = R(T) + \alpha \cdot |T|$$

- $R(T)$: error total del árbol (misclassification rate o MSE)
- $|T|$: número de hojas terminales (complejidad)
- $\alpha \geq 0$: parámetro de penalización

Comportamiento:

- $\alpha = 0$: árbol completo (máx. overfitting)
- $\alpha \to \infty$: solo la raíz (máx. underfitting)
- Se tunea con Cross-Validation

Analogía con regularización

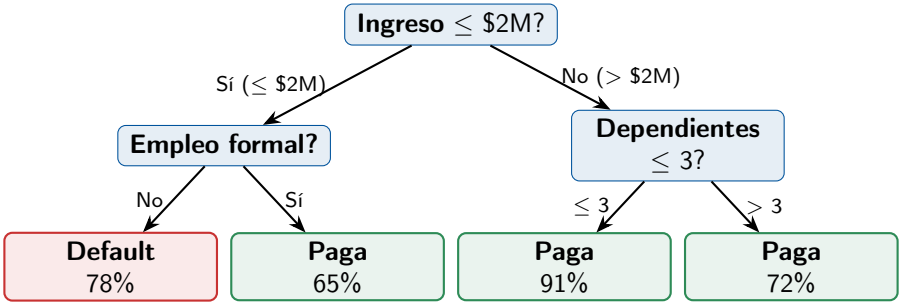
α en CART $\equiv \lambda$ en Lasso/Ridge

Misma lógica: penalizar complejidad para mejorar generalización.

`ccp_alpha` en scikit-learn.

Ejemplo: Elección Binaria — ¿Otorgar Crédito?

Contexto: Un banco predice default (1) o no default (0).



Lectura: El ingreso es la variable más discriminante. Para ingresos bajos, tener empleo formal es decisivo. Para ingresos altos, el número de dependientes modula el riesgo. Noten la **interacción** ingreso $\times$ empleo que un logit no capturaría sin especificarla manualmente.

Ventajas y Limitaciones del CART

Ventajas

- Interpretabilidad: reglas if-then
- No requiere estandarización
- Maneja missing values (surrogate splits)
- Captura interacciones automáticamente
- Robusto a outliers
- Datos mixtos sin transformación

Limitaciones

- **Alta varianza:** pequeños cambios en datos $\rightarrow$ árboles completamente distintos
- Fronteras solo ortogonales
- Greedy: no óptimo global
- Propenso a overfitting
- Ineficiente para relaciones lineales
- Sesgo hacia variables con muchos niveles

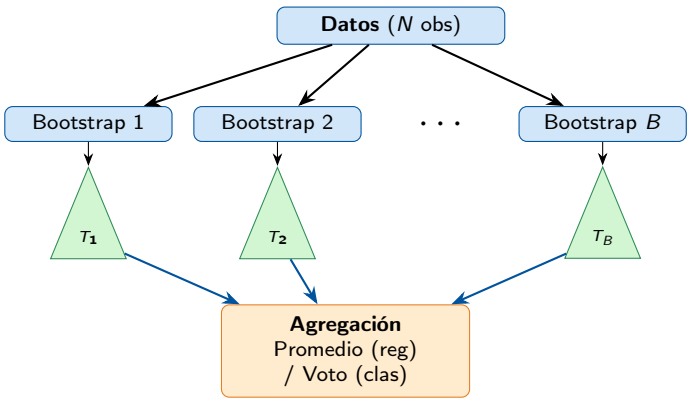
¿Podemos mantener la flexibilidad pero reducir la varianza?

$\rightarrow$ **Sí: Random Forest**

Agenda de hoy

Bagging: La Idea (Breiman, 1996)

Problema: Un solo árbol tiene alta varianza.
Solución: Entrenar **muchos árboles** en muestras diferentes y **agregar** sus predicciones.



Bootstrap: muestreo **con reemplazo** de N obs. Cada muestra omite $\approx 37\%$ de los datos (out-of-bag).

Random Forest = Bagging + Aleatorización de Features

Breiman (2001) añadió un ingrediente clave: en cada split, solo considerar un **subconjunto aleatorio** de m variables (de las p disponibles).

¿Por qué? Sin esto, si hay una variable dominante (como Sex en el Titanic), *todos* los árboles hacen el mismo primer split $\rightarrow$ árboles correlacionados $\rightarrow$ promediar no reduce la varianza.

Fórmula clave: varianza del ensamble

Si tenemos B árboles con varianza σ^2 y correlación promedio ρ:

$$\text{Var}\left(\frac{1}{B}\sum_{b=1}^{B} T_b(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- Aumentar B reduce el segundo término, pero **no** el primero
- La única forma de reducir ρ es la **aleatorización de features**

Reglas de dedo: $m \approx \sqrt{p}$ (clasificación), $m \approx p/3$ (regresión).

Todos los Hiperparámetros de Random Forest

Hiperparámetro	Default	Efecto
— Del Bosque —		
`n_estimators`	100	N° de árboles B. Más es mejor, retornos decrecientes.
`max_features`	`sqrt`	Features por split (m). **El más importante del RF.**
`bootstrap`	`True`	Si usa muestreo con reemplazo.
`oob_score`	`False`	Error out-of-bag (estimación "gratis" de test error).
`max_samples`	`None`	Proporción de datos por bootstrap.
— De Cada Árbol —		
`max_depth`	`None`	Profundidad máxima por árbol.
`min_samples_split`	2	Mínimo para dividir un nodo.
`min_samples_leaf`	1	Mínimo por hoja.
`max_leaf_nodes`	`None`	Máximo de hojas por árbol.
`min_impurity_decrease`	0.0	Umbral de reducción de impureza.
`criterion`	`gini`	gini / entropy / log_loss.
`class_weight`	`None`	Ponderación de clases.

Guía de Tuning para Random Forest

Prioridad Alta:

1. `n_estimators`: empezar con 500
2. `max_features`: tunear $[1, \sqrt{p}, p/3, p/2]$
3. `max_depth`: probar `None`, 10, 20, 30

Prioridad Media:

4. `min_samples_leaf`: 1, 5, 10, 20
5. `min_samples_split`: 2, 10, 20
6. `class_weight`: `balanced` si hay desbalance

Prioridad Baja:

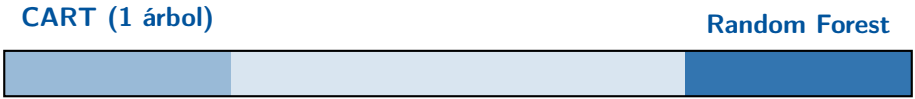
7. `max_leaf_nodes`: rara vez necesario
8. `criterion`: diferencia mínima en práctica
9. `max_samples`: probar 0.7–1.0

OOB Score

Con `oob_score=True`, cada árbol se evalúa en los $\approx$37% de datos que no usó.

Estimación del test error **sin CV**. Útil con datasets grandes.

CART vs. Random Forest: El Trade-off



¿Cuándo usar cada uno?

- **CART:** Interpretabilidad esencial (regulación, explicar a un juez, política pública con accountability directa)
- **RF:** Precisión prioritaria y se puede explicar vía feature importance / SHAP

Agenda de hoy

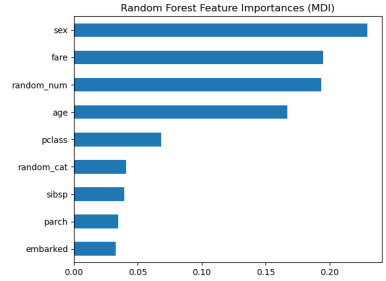
Feature Importance en Random Forest

Al perder interpretabilidad directa, necesitamos herramientas para **explicar** el modelo.

Mean Decrease in Impurity (MDI):
Suma las reducciones de impureza de X_j en todos los nodos:

$$\text{Imp}_{\text{MDI}}(X_j) = \frac{1}{B} \sum_{b=1}^{B} \sum_{t \in T_b} \Delta I_t \cdot 1(v_t = j)$$

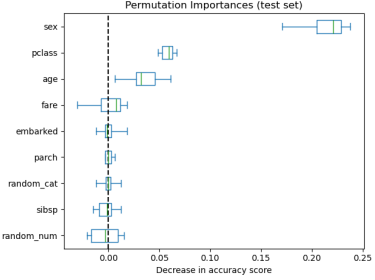
Sesgo: favorece variables con muchos valores únicos.



Permutation Importance:
Permuta X_j y mide cuánto **empeora** el score:

$$\text{Imp}_{\text{Perm}}(X_j) = S_{\text{orig}} - S_{\text{permutado}}$$

Más robusta. Calculable sobre OOB o test set.



Limitaciones de Feature Importance Tradicional

Problemas con MDI y Permutation Importance:

- Son medidas **globales**: dicen qué variable importa en promedio, pero no cómo afecta a una predicción *individual*
- MDI es **inconsistente**: cambiar el modelo para que dependa más de una variable puede *reducir* su importancia asignada (Lundberg et al., 2018)
- Con variables correlacionadas, ambos métodos pueden dar resultados engañosos
- No capturan la **dirección** del efecto (¿la variable aumenta o disminuye la predicción?)

Necesitamos algo mejor → SHAP Values

SHAP Values: Fundamento en Teoría de Juegos

SHAP = **SH**apley **A**dditive ex**P**lanations (Lundberg & Lee, NeurIPS 2017).

Viene del **valor de Shapley** (1953), un concepto de teoría de juegos cooperativos:

Idea

¿Cuál es la contribución "justa" de cada feature X_j a la predicción $\hat{f}(x)$ de una observación específica?

El valor de Shapley de la feature j es:

$$\phi_j = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left[f(S \cup \{j\}) - f(S) \right]$$

Es decir: para **todas** las posibles coaliciones de features S, calcula la contribución marginal de añadir X_j, y promedia ponderando por el número de coaliciones de cada tamaño.

Complejidad exacta: $O(2^p)$ — exponencial. Por eso se necesitan aproximaciones eficientes.

Propiedades de SHAP (y por qué importan)

SHAP es el **único** método de atribución aditiva que satisface simultáneamente:

1. **Eficiencia (Local Accuracy):** Las contribuciones suman a la predicción.

$$\hat{f}(x) = \phi_0 + \sum_{j=1}^{p} \phi_j$$

   donde $\phi_0 = E[\hat{f}(X)]$ (predicción promedio del modelo).

2. **Simetría:** Si dos features contribuyen igual, reciben el mismo SHAP value.
3. **Nulidad (Missingness):** Si una feature no cambia la predicción para ninguna coalición, su $\phi_j = 0$.
4. **Consistencia:** Si cambio el modelo para que dependa más de X_j, su ϕ_j no disminuye.

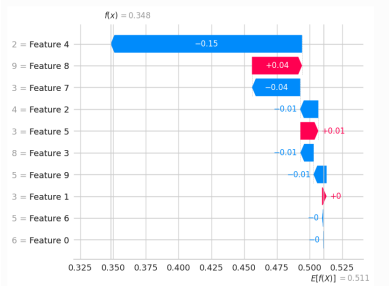
Ref: Lundberg, S. M., & Lee, S.-I. (2017). "A Unified Approach to Interpreting Model Predictions." NeurIPS.

Ref: Shapley, L. S. (1953). "A Value for n-Person Games." Contributions to the Theory of Games.

Visualizaciones de SHAP

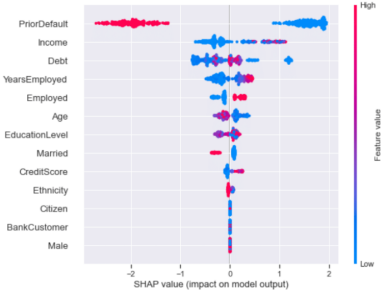
SHAP ofrece visualizaciones que van de lo **individual** a lo **global**:

Local (una predicción):



¿Por qué este cliente fue rechazado?
Cada barra muestra cuánto contribuyó cada feature a mover la predicción desde ϕ_0 (baseline) hasta $\hat{f}(x)$.

Global (todo el modelo):



¿Qué features importan y cómo?
Cada punto es una observación. Color = valor del feature. Posición horizontal = SHAP value (contribución).

SHAP: De Local a Global

La potencia de SHAP: la importancia global es *consistente* con las explicaciones locales.

Global importance = promedio de los valores absolutos de SHAP:

$$\text{Imp}_{\text{SHAP}}(X_j) = \frac{1}{N}\sum_{i=1}^{N} |\phi_j^{(i)}|$$

¿Por qué es mejor que MDI / Permutation?

- **Consistente:** si el modelo depende más de X_j, su importancia sube (garantizado)
- **Aditivo:** las contribuciones suman exactamente a la predicción
- **Direccional:** sabemos si la feature *aumenta* o *reduce* la predicción
- **Individual:** podemos explicar cada predicción, no solo promedios

Aplicación en consultoría

En focalización de programas sociales: SHAP explica *por qué* cada hogar fue clasificado como elegible o no. Esto es esencial para **accountability** y **derecho a explicación** en IA Responsable.

Referencias

- CGAP (2024). "Gender-Intentional Credit Scoring." https://www.cgap.org/research/publication/gender-intentional-credit-scoring
- CEGA/Berkeley. "Gender-Differentiated Credit Algorithms." https://cega.berkeley.edu/research/gender-differentiated-credit-algorithms-using-machine-learning/
- MIT Technology Review (2019). "There's an easy way to make lending fairer for women." https://www.technologyreview.com/2019/11/15/131935/
- Breiman, L. et al. (1984). *Classification and Regression Trees*. Wadsworth.
- Breiman, L. (2001). "Random Forests." *Machine Learning*, 45(1), 5–32.
- Lundberg, S. M. & Lee, S.-I. (2017). "A Unified Approach to Interpreting Model Predictions." NeurIPS.
- Lundberg, S. M. et al. (2020). "From Local Explanations to Global Understanding with Explainable AI for Trees." *Nature Machine Intelligence*, 2, 56–67.
- Grinsztajn, L. et al. (2022). "Why do tree-based models still outperform deep learning on tabular data?" arXiv:2207.08815.

¡Gracias!

s.neira10@uniandes.edu.co