Facsímil 07

Evaluar, calibrar e interpretar

Capítulo 06

Interpretabilidad práctica y evaluación

686f6c61 · 2026

Capítulo 06: Interpretabilidad práctica y evaluación

Entrando en el tema

Este facsímil empezó con una idea sobria: una eval existe para tomar una decisión. Después medimos errores, evaluamos RAG, diseñamos evaluadores, calibramos scores y convertimos incertidumbre en política. Ahora queda una pregunta que aparece siempre en ingeniería de IA:

> ¿Podemos explicar lo suficiente para depurar, publicar, limitar o rechazar este sistema?

Interpretabilidad no es una palabra para tranquilizar a alguien en una reunión. Tampoco es una imagen bonita, una tabla de pesos o una respuesta del modelo diciendo “he decidido esto por...”. Interpretabilidad, en ingeniería, es una herramienta para hacer mejores preguntas: qué feature pesa, qué caso cambia, qué slice se rompe, qué explicación es estable, qué parte del sistema no entendemos todavía y qué decisión podemos defender.

La idea central del capítulo es esta: **una explicación no se acepta porque suene bien; se acepta porque ayuda a diagnosticar y resiste comprobaciones**.

Qué deberías poder hacer al terminar

Este capítulo no pretende que salgas repitiendo nombres de librerías. Pretende que puedas mirar una explicación y decidir si sirve para depurar un sistema, para documentar una release, para pedir revisión humana o para decir “todavía no podemos publicar esto”.

Al terminar deberías poder hacer esto:

RESULTADO DE APRENDIZAJE	EVIDENCIA DE QUE LO SABES HACER
Separar interpretabilidad, explicabilidad y transparencia.	No usas esas palabras como sinónimos automáticos.
Elegir método según pregunta.	Distingues explicación local, global, contrafactual, conceptual y mecánica.
Evaluar una explicación.	Mides fidelidad, estabilidad, sensibilidad y utilidad operativa.
Leer atribuciones con cautela.	Sabes por qué una atribución plausible puede ser infiel.
Diseñar contrafactuales útiles.	Separas cambios accionables de cambios imposibles o injustos.
Conectar interpretabilidad con EvalOps.	Conviertes explicaciones en checks, model card y casos de regresión.
Cerrar el facsímil con criterio.	Sabes integrar métricas, RAG, evaluadores, calibración e interpretación en una decisión de release defendible.

El problema: una explicación puede sonar perfecta y no explicar nada

Imagina un sistema que prioriza tickets académicos. Para un caso concreto devuelve:

```
{
  "score": 0.86,
  "decision": "urgente",
  "explanation": "El caso parece urgente por su tono y por la fecha cercana."
}
```

La frase suena humana. Pero un ingeniero debería preguntar:

PREGUNTA	POR QUÉ IMPORTA
¿El modelo tenía una feature llamada `tono`?	Si no existe, la explicación puede ser una racionalización.
¿Qué pasa si eliminamos la feature principal?	Si la salida no cambia, la explicación no sostiene la decisión.
¿La explicación cambia ante pequeñas perturbaciones?	Si cambia demasiado, no es estable.
¿El caso era parte de un slice problemático?	Una explicación local puede esconder un patrón global malo.
¿El cambio recomendado es accionable?	No todo contrafactual sirve para una persona o un equipo.

Esto es especialmente delicado en LLMs. Una respuesta puede construir una narración convincente después de producir la salida. En ese caso, la explicación puede ser plausible para nosotros, pero no fiel al proceso que produjo el resultado.

Por eso el capítulo no va de “hacer explicable la IA” en abstracto. Va de crear un expediente técnico: qué método usamos, qué pregunta responde, qué prueba lo contradice y qué decisión permite tomar.

Qué no es interpretabilidad

Interpretabilidad no es necesariamente simplicidad. Un modelo lineal puede ser fácil de leer y aun así estar usando features mal definidas, proxies pobres o datos incompletos. Un árbol pequeño puede ser comprensible y seguir aprendiendo una regla poco útil.

Tampoco es una promesa de verdad. Una explicación post hoc puede aproximar el comportamiento de un modelo complejo, pero no convertirse en el modelo real. LIME aproxima localmente; SHAP reparte contribuciones bajo supuestos; saliency maps señalan sensibilidad; contrafactuales proponen cambios; ninguna de esas piezas sustituye una evaluación.

Y, sobre todo, interpretabilidad no es decoración de compliance. Si nadie puede decir qué decisión cambia gracias a la explicación, quizá solo hemos añadido otra pantalla.

CONFUSIÓN	LECTURA DE INGENIERÍA
“Tenemos explicación, así que el sistema es fiable”.	La explicación también se evalúa.
“El mapa de calor marca la zona importante”.	Hay que probar sensibilidad, estabilidad y sanity checks.
“El modelo dice por qué decidió”.	Una explicación textual puede no ser fiel.
“SHAP lo arregla”.	SHAP responde una familia concreta de preguntas bajo supuestos concretos.
“Contrafactual significa recomendación”.	Solo algunos cambios son accionables y aceptables.

Lipton advertía que “interpretabilidad” suele mezclar propiedades distintas: transparencia, simulabilidad, decomponibilidad, post hoc explanations y confianza humana.[1] Doshi-Velez y Kim proponían tratar la interpretabilidad como una cuestión evaluable, no como una etiqueta estética.[2]

Qué sí es interpretar un sistema de IA

Interpretar es responder una pregunta situada. No “explícame el modelo”, sino:

PREGUNTA	MÉTODO RAZONABLE
¿Por qué este caso se marcó como urgente?	Explicación local y prueba de borrado.
¿Qué features pesan globalmente?	Importancia por permutación, SHAP agregado o modelo interpretable.
¿Qué tendría que cambiar para otra decisión?	Contrafactual accionable.
¿Qué concepto humano usa el modelo?	TCAV o análisis por conceptos.
¿Qué región de una imagen activó una clase?	Grad-CAM u otro método visual con sanity checks.
¿Qué parte interna participa en una asociación factual?	Intervenciones causales o análisis mecanicista.
¿Qué explicación puedo mostrar a usuario final?	Una explicación validada por utilidad, no solo por fidelidad técnica.

Hay dos ejes que conviene escribir siempre:

EJE	PREGUNTA
Local frente a global	¿Explicamos un caso concreto o el comportamiento general?
Fidelidad frente a plausibilidad	¿Refleja el modelo o solo convence a la persona?
Diagnóstico frente a comunicación	¿Sirve para depurar o para informar una decisión?
Accionable frente a descriptivo	¿Permite hacer algo distinto?

Jacovi y Goldberg separan explícitamente fidelidad y plausibilidad en NLP: una explicación puede parecer buena a una persona y no reflejar el mecanismo que produjo la salida.[3] Esa distinción es clave para LLMs.

Fecha de corte del estado del arte

Fecha de corte: 6 de junio de 2026.
Fuentes consultadas: trabajos clásicos sobre ciencia de la interpretabilidad, LIME, SHAP, Integrated Gradients, Grad-CAM, sanity checks, contrafactuales, modelos interpretables para decisiones sensibles, TCAV, interpretabilidad fiel en NLP y localización causal de asociaciones factuales en GPT.

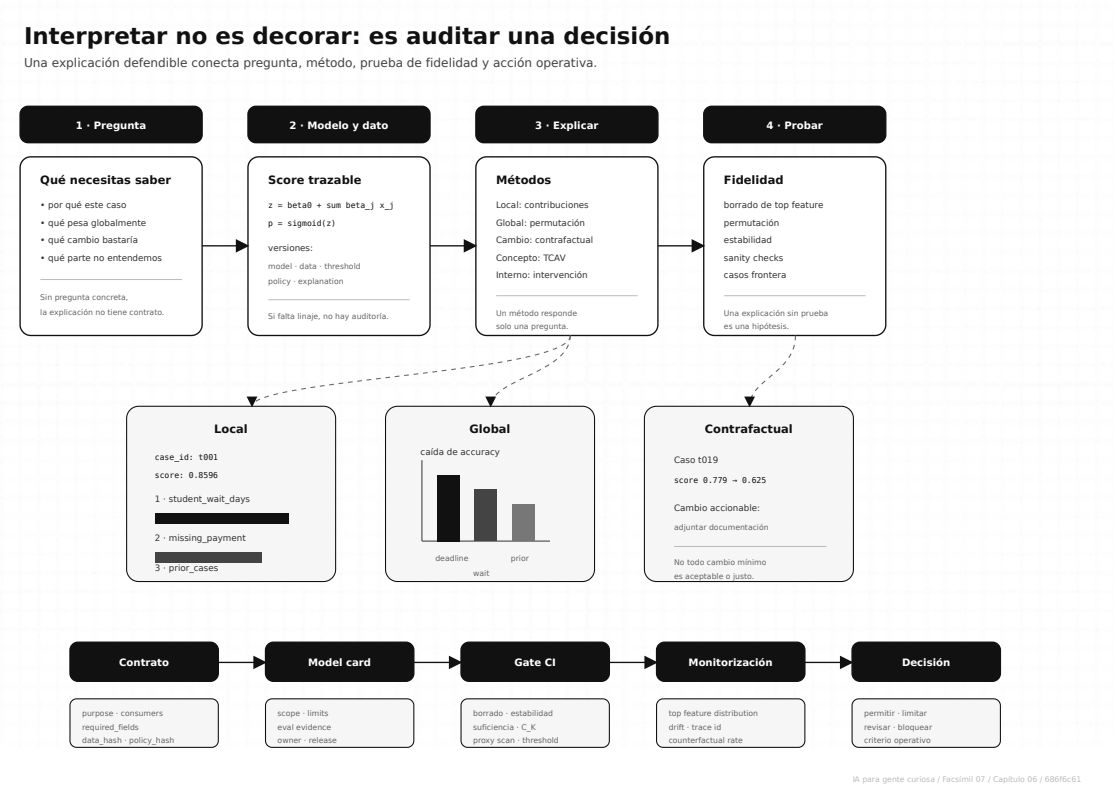
LIME propone aproximar localmente un modelo complejo con un modelo interpretable alrededor de una predicción concreta.[4] SHAP conecta atribuciones aditivas con valores de Shapley y un marco común para asignar importancia a features.[5] Integrated Gradients plantea axiomas como sensibilidad e invariancia de implementación para atribuciones en redes profundas.[6]

Grad-CAM localiza regiones relevantes para modelos visuales usando gradientes hacia capas convolucionales.[7] Adebayo y otros mostraron que algunos mapas de saliencia pueden fallar sanity checks: si la explicación apenas cambia al aleatorizar parámetros o etiquetas, hay que desconfiar.[8]

Los contrafactuales explican decisiones indicando cambios mínimos que producirían otra salida.[9] Rudin defiende que en decisiones de alto impacto conviene preferir modelos interpretables cuando sea posible, en lugar de explicar una caja negra después.[10]

Para conceptos humanos, TCAV cuantifica sensibilidad a direcciones conceptuales aprendidas.[11] En modelos de lenguaje, trabajos como ROME usan intervenciones causales para localizar asociaciones factuales en GPT.[12]

Anatomía de una auditoría de interpretabilidad



> Una auditoría de interpretabilidad empieza con una pregunta y termina en una decisión. Entre medias hay método, prueba y documentación.

La explicación empieza en el contrato de datos

Antes de hablar de SHAP, LIME o contrafactuales hay una pregunta más aburrida y más útil: ¿qué significa cada feature? Si el dataset no tiene contrato, la explicación hereda ambigüedad. Una feature llamada `prior_cases` puede significar casos previos reales, tickets duplicados, incidencias abiertas, expedientes rechazados o simples contactos con soporte. La explicación “prior_cases empuja la decisión” no vale lo mismo en cada caso.

En ingeniería de IA conviene documentar cada feature con cuatro piezas:

PIEZA	PREGUNTA	EJEMPLO DEL CUADERNO
Semántica	¿Qué representa exactamente?	`student_wait_days` mide días de espera acumulada.
Procedencia	¿De qué sistema sale y cuándo se actualiza?	CRM académico o sistema de tickets.
Rango válido	¿Qué valores son posibles y cuáles son errores?	0 a 21 en `student_wait_days`.
Uso permitido	¿Puede usarse para decidir, revisar o solo diagnosticar?	`missing_payment` puede activar revisión, no comunicación automática.

Esto importa porque una explicación puede ser fiel al modelo y aun así mala para producto. Si `prior_cases` está muy correlacionada con `student_wait_days`, quizá ambas features cuentan la misma historia con dos nombres distintos. Si `contains_policy_keyword` se dispara por palabras mal tokenizadas, la explicación puede parecer jurídica sin serlo. Si `docs_attached` se usa como recomendación, hay que confirmar que adjuntar documentos sea realmente una acción disponible para la persona o para el equipo.

La interpretabilidad útil, por tanto, no termina en “la feature pesa”. Termina en una decisión sobre datos: mantener, renombrar, separar, eliminar, auditar por slice o convertir en una regla de revisión humana. Esa es la diferencia entre una explicación de demo y una explicación que un equipo puede defender.

Las fórmulas que sí conviene saber

En el cuaderno del facsímil usamos un modelo logístico lineal porque permite ver las tripas sin depender de librerías. No porque todos los sistemas reales sean lineales, sino porque es el punto más honesto para aprender a auditar explicaciones. La formulación viene de la familia de modelos lineales generalizados y aparece en textos estándar de aprendizaje estadístico como Hastie, Tibshirani y Friedman.[13]

El modelo calcula un logit:

$$z(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j$$

Y lo convierte en probabilidad con una sigmoide:

$$\hat{p}(x) = \sigma(z) = \frac{1}{1 + e^{-z(x)}}$$

En palabras: primero sumamos señales ponderadas en escala logit y después aplicamos la sigmoide para obtener un valor entre 0 y 1. Ese valor puede leerse como probabilidad estimada solo si el modelo y la calibración lo sostienen; por eso el capítulo anterior fue tan importante.

SÍMBOLO	SIGNIFICADO	EJEMPLO
x	Caso que evaluamos.	Ticket `t001`.
x_j	Valor de la feature j.	`student_wait_days = 12`.
β_j	Peso de la feature j.	0,13.
β_0	Intercepto.	-2,35.
$z(x)$	Suma lineal antes de probabilidad.	1,812.
$\hat{p}(x)$	Probabilidad estimada.	0,8596.

En un modelo lineal aditivo, la contribución local de una feature puede leerse como el término que aporta al logit. Esto no es una métrica nueva inventada para el capítulo: es la descomposición algebraica del predictor lineal $z(x)$. La ventaja pedagógica es que permite comprobar si la explicación que mostramos coincide con los términos que el propio modelo suma.

$$c_j(x) = \beta_j x_j$$

En palabras: si una feature tiene peso positivo y valor alto, empuja el logit hacia arriba; si tiene peso negativo, lo empuja hacia abajo; si vale cero, no aporta en ese caso aunque el peso exista.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$c_j(x)$	Contribución de la feature j al logit.	1,56.
β_j	Peso aprendido o fijado.	0,13.
x_j	Valor del caso.	12 días de espera.

Para `t001`, el cuaderno obtiene:

FEATURE	VALOR	PESO	CONTRIBUCIÓN
`student_wait_days`	12	0,13	1,56
`missing_payment`	1	1,25	1,25
`prior_cases`	3	0,28	0,84

La explicación es clara: el modelo sube prioridad por días de espera, pago pendiente y casos previos. Pero no nos basta con leer esa tabla. Probamos si al quitar la feature superior el score cae de forma relevante.

La prueba de borrado no necesita una ecuación adicional en el texto. Es un procedimiento: calculas el score original, neutralizas una feature, vuelves a calcular el score y miras cuánto cae. En el cuaderno del facsímil, por ejemplo, neutralizar `student_wait_days` en el caso más sensible produce una caída de 0,44. Esa caída no demuestra causalidad por sí sola, pero sí dice: “esta feature sostiene buena parte de esta decisión”.

PASO	QUÉ HACES	QUÉ APRENDES
Score original	Ejecutas el caso tal como llega.	Punto de partida.
Neutralización	Sustituyes una feature por un valor base razonable.	Qué pasa si esa señal desaparece.
Comparación	Miras la caída de score.	Sensibilidad local de la decisión.
Revisión	Compruebas si la feature es legítima y accionable.	Si la explicación se puede defender.

Para importancia global por permutación usamos la idea de romper la asociación entre una feature y la salida, medir cuánto cae el rendimiento y leer esa caída como dependencia del modelo. La importancia por permutación aparece en la tradición de Random Forests de Breiman y se generaliza como *model reliance* en Fisher, Rudin y Dominici.[14][15]

$$I_j = M(D) - M(D_{\pi(j)})$$

En palabras: si al permutar una feature la métrica baja mucho, el modelo dependía de esa feature para rendir. Ojo: esto no prueba causalidad y puede crear combinaciones de datos poco realistas si las features están muy correlacionadas. En ingeniería se usa como señal de dependencia, no como sentencia causal.

SÍMBOLO	SIGNIFICADO	EJEMPLO
I_j	Importancia global de la feature j.	0,25 para `deadline_hours`.
$M(D)$	Métrica original en dataset.	Accuracy 0,75.
$D_{\pi(j)}$	Dataset con la feature j permutada.	`deadline_hours` desordenada.

En el cuaderno, las tres features con mayor caída son:

FEATURE	ACCURACY ORIGINAL	ACCURACY PERMUTADA	CAÍDA
`deadline_hours`	0,75	0,50	0,25
`student_wait_days`	0,75	0,55	0,20
`prior_cases`	0,75	0,60	0,15

Para contrafactuales seguimos la formulación de Wachter, Mittelstadt y Russell: buscar un caso cercano que cambie la decisión.[16]

$$x^{\prime *} = \arg\min_{x'} d(x, x') \quad \text{sujeto a} \quad f(x') \neq f(x)$$

En palabras: queremos el cambio más pequeño que lleve a otra decisión. En producto añadimos una restricción que la fórmula pura no resuelve sola: el cambio tiene que ser accionable, aceptable y compatible con el dominio.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$x^{\prime *}$	Caso contrafactual elegido.	Ticket con documentación adjunta.
$d(x, x')$	Distancia o coste de cambiar de x a x'.	Un cambio accionable.
$f(x')$	Decisión del modelo para el caso modificado.	Pasa de urgente a normal.

Esta fórmula parece limpia, pero en producto tiene una condición escondida: el cambio debe ser accionable y aceptable. No sirve decir “si fueras otra persona” o “si tu historial no existiera”. Sirve decir “si falta documentación, pide la documentación” o “si el pago está pendiente, compruébalo”.

Método no es garantía: cómo elegir bien

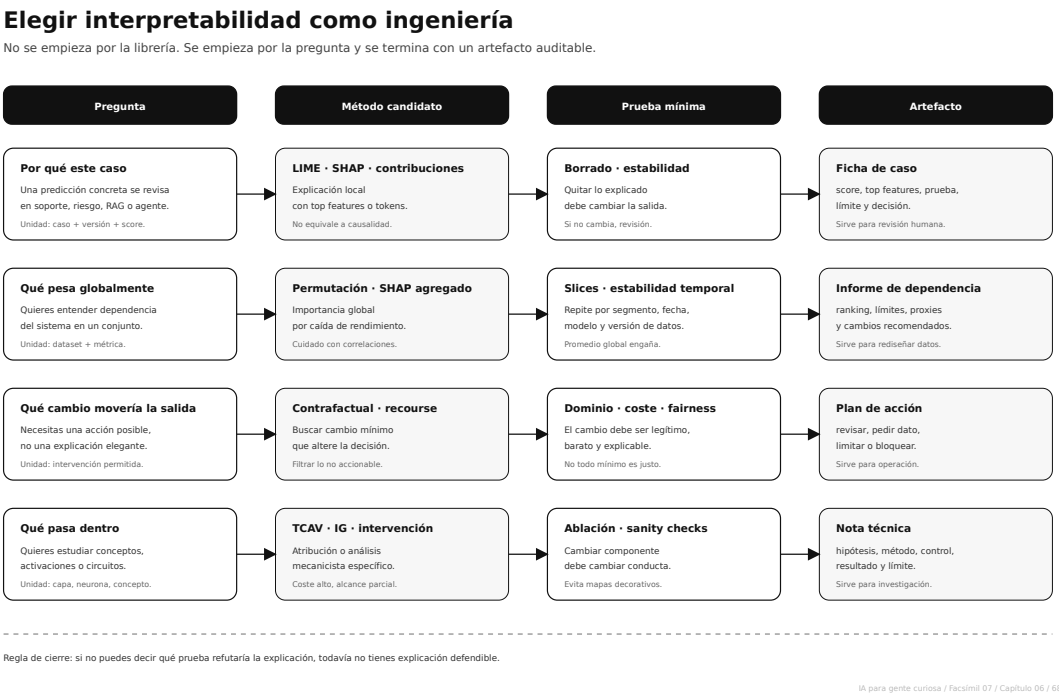
Los métodos de interpretación responden preguntas distintas:

MÉTODO	PREGUNTA QUE RESPONDE	RIESGO
Modelo interpretable	¿Puedo entender el mecanismo completo?	Puede ser demasiado simple para el problema.
LIME	¿Qué modelo simple aproxima esta predicción localmente?	Depende de perturbaciones y vecindario.
SHAP	¿Cómo se reparten contribuciones entre features?	Depende del fondo, correlaciones y coste computacional.
Integrated Gradients	¿Qué entrada aporta al cambio desde una línea base?	La línea base puede cambiar la historia.
Grad-CAM	¿Qué región visual pesa para una clase?	Mapa grueso y sensible a sanity checks.
Contrafactuales	¿Qué cambio produciría otra decisión?	Puede proponer cambios no accionables.
TCAV	¿Qué concepto humano afecta al modelo?	Requiere buenos ejemplos del concepto.
Intervenciones internas	¿Qué componente participa causalmente?	Es costoso, específico y fácil de sobreinterpretar.

La regla práctica:

> No elijas el método por popularidad. Elige el método por la decisión que necesitas tomar.

Si el equipo quiere depurar un clasificador tabular, importancia por permutación y contrafactuales pueden bastar. Si quiere revisar un modelo de visión, Grad-CAM puede ayudar, pero con sanity checks. Si quiere saber si un LLM recupera un hecho por una zona concreta, hacen falta intervenciones más cercanas a interpretabilidad mecanicista. Si la decisión tiene alto impacto, Rudin nos recuerda que quizá el primer debate no sea “cómo explico la caja negra”, sino “por qué no uso un modelo interpretable desde el principio”.



Mapa de decisión para no elegir SHAP, LIME, Grad-CAM o contrafactuales por moda. La pregunta manda; el artefacto final se audita.

Herramientas que verás en equipos

Las herramientas ayudan, pero no sustituyen el contrato. En un equipo real conviene separar tres preguntas: qué tipo de modelo tienes, qué tipo de entrada explicas y qué evidencia necesitas guardar. Un paquete puede generar una visualización excelente y aun así no responder a la pregunta de release.

HERRAMIENTA	ENCAJA MEJOR CUANDO	CUIDADO DE INGENIERÍA
SHAP	Necesitas atribuciones locales o globales y puedes definir bien el fondo de comparación.[17]	El resultado depende del explainer, del background y del coste computacional.
LIME	Quieres aproximar localmente una predicción con un modelo interpretable.[18]	El vecindario y las perturbaciones cambian la explicación.
Captum	Trabajas con PyTorch y quieres Integrated Gradients, saliency, DeepLIFT u otras atribuciones.[19]	Hay que fijar línea base, modalidad y controles.
InterpretML / interpret-community	Trabajas con tabular, glassbox models o explicaciones comparables en notebooks.[20]	Útil para análisis, pero hay que versionar datos y configuración.
Alibi Explain	Necesitas explicaciones locales, contrafactuales o inspección de modelos en flujos Python.[21]	No conviertas un contrafactual sintético en recomendación sin filtro de dominio.
scikit-learn inspection	Quieres importancia por permutación reproducible en modelos clásicos.[22]	La permutación puede crear filas irreales si las variables están correlacionadas.

La herramienta que elegiría para empezar no siempre es la más famosa. Si el problema es tabular y la decisión es sensible, empezaría por un modelo interpretable o por una auditoría de features. Si el sistema es RAG, guardaría trazas y evidencia recuperada antes de pedir una explicación textual. Si el sistema es de visión, usaría mapas visuales solo con sanity checks. Si es un LLM, separaría explicación narrada, traza operativa y evidencia externa.

Cómo evaluar una explicación

Una explicación debe pasar pruebas. No todas son matemáticas sofisticadas; algunas son puro criterio de ingeniería:

PRUEBA	QUÉ COMPRUEBA	SEÑAL MALA
Borrado	Quitar la parte explicada cambia la salida.	La salida no cambia.
Inserción	Añadir features importantes recupera la salida.	Features supuestamente clave no aportan.
Permutación	Desordenar una feature global baja métrica.	Importancia alta sin caída real.
Estabilidad	Perturbaciones pequeñas conservan explicación.	Explicación cambia por ruido menor.
Sanity check	Explicación responde a modelo/datos reales.	Mapa igual con pesos aleatorios.
Revisión de slice	Explicación se sostiene por segmento.	Global bien, segmento mal.
Utilidad humana	La persona decide mejor con la explicación.	Más confianza sin mejor decisión.

Hay una trampa clásica: una explicación muy bonita puede aumentar confianza sin aumentar acierto. Eso es peligroso. En entornos de producto, una explicación debería medirse por decisión: reduce errores, mejora revisión, acelera diagnóstico o permite detectar un problema antes.

En el día a día

En un proyecto de IA, interpretabilidad aparece en cinco momentos:

MOMENTO	PREGUNTA
Diseño	¿Necesitamos modelo interpretable por defecto?
Desarrollo	¿Qué features dominan y cuáles son proxies problemáticos?
Evaluación	¿Las explicaciones se sostienen en fallos y slices?
Producción	¿Podemos explicar una incidencia o una decisión revisada?
Mejora	¿Qué casos se convierten en regresión o cambio de datos?

Un ejemplo cercano: en un asistente RAG, la explicación no debería ser “respondí esto porque el modelo lo consideró relevante”. Debería mostrar:

CAPA	EVIDENCIA
Retrieval	Documentos recuperados, scores, reranker y citas usadas.
Respuesta	Afirmaciones principales y soporte por chunk.
Evaluación	Groundedness, cobertura de cita, abstención y errores.
Calibración	Probabilidad de respuesta aceptada o zona de revisión.
Operación	Modelo, prompt, índice, release y trace id.

En agentes, una explicación útil no es solo el resumen final. Es la trayectoria: qué herramienta eligió, con qué argumentos, qué observó, qué descartó, cuánto costó y dónde pidió revisión.

Por qué debería importarte

La interpretabilidad cambia decisiones concretas de ingeniería. Si una explicación se usa solo para adornar una pantalla, añade ruido. Si se usa bien, permite decidir si una release se publica, si un caso pasa a revisión humana, si una feature debe salir del dataset, si un contrato de salida necesita campos nuevos o si una incidencia de producción se puede depurar sin adivinar.

Esto importa especialmente cuando el sistema empieza a afectar a otras personas. Un score mal explicado puede crear confianza falsa. Un contrafactual mal diseñado puede recomendar una acción imposible. Una explicación global sin slices puede ocultar que el modelo funciona bien en promedio y mal en un grupo concreto de casos. Y una explicación textual de un LLM puede sonar convincente sin ser evidencia del mecanismo que produjo la respuesta.

En un equipo serio, la explicación no vive sola. Vive junto al contrato de datos, la model card, las trazas, los umbrales de calibración, los checks de CI y la política de revisión. Esa es la razón por la que este capítulo no se queda en “qué es SHAP” o “qué es LIME”: el objetivo es que sepas convertir una explicación en un artefacto que alguien pueda revisar, discutir y defender.

Contrato de explicación: quién puede usarla y para qué

Una explicación profesional debería tener contrato. No basta con producir un gráfico o una frase. Hay que declarar para qué sirve, quién puede verla, qué campos son obligatorios, qué versión del modelo la produjo y qué usos no están permitidos.

En el cuaderno del facsímil generamos `output/explanation_contract.json`. Un ejemplo resumido:

```
{
  "model_version": "support-prioritizer-linear-v1",
  "explanation_policy_version": "interp-audit-v1",
  "owner": "equipo-ia",
  "purpose": "diagnostico interno y revision operativa de tickets priorizados",
  "allowed_consumers": ["ingenieria", "soporte_n2", "producto"],
  "not_for": ["decision final sin revision", "comunicacion automatica a usuario"],
  "required_fields": [
    "case_id",
    "model_version",
    "score",
    "prediction",
    "top_features",
    "deletion_test",
    "counterfactual",
    "data_hash_sha256",
    "policy_hash_sha256"
  ]
}
```

La parte importante no es el JSON bonito. Es la disciplina:

CAMPO	QUÉ EVITA
`purpose`	Que una explicación de diagnóstico acabe vendida como verdad final.
`allowed_consumers`	Que cualquier equipo use la explicación sin entender sus límites.
`not_for`	Que se automatice una decisión que exige revisión.
`required_fields`	Que una explicación llegue sin score, versión, prueba o linaje.
`data_hash_sha256`	Que no sepamos con qué datos se generó.
`policy_hash_sha256`	Que cambie el umbral o la política y nadie lo vea.

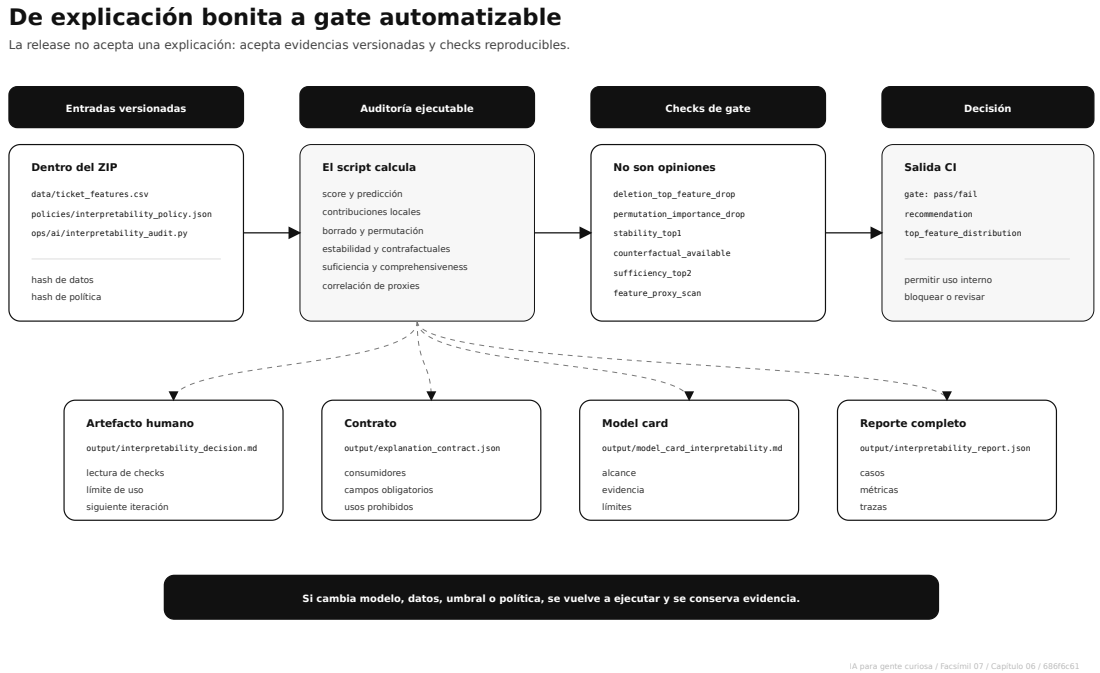
Esto conecta con las model cards, pero baja a operación. Una model card explica el sistema; el contrato de explicación fija cómo se puede consumir una explicación concreta en una run concreta.

Tests de explicación en CI

Si una explicación forma parte de una release, también debería tener tests. No en el sentido de “la explicación es bonita”, sino en el sentido de que pasa checks mínimos antes de publicar una versión.

El cuaderno produce `output/ci_explanation_gate.json`:

```
{
  "gate": "pass",
  "checks": [
    {"name": "deletion_top_feature_drop", "passes": true},
    {"name": "permutation_importance_drop", "passes": true},
    {"name": "stability_top1", "passes": true},
    {"name": "counterfactual_available", "passes": true},
    {"name": "comprehensiveness_top2", "passes": true},
    {"name": "sufficiency_top2", "passes": true},
    {"name": "feature_proxy_scan", "passes": true}
  ],
  "recommendation": "permitir uso interno con monitorización"
}
```



El gate de explicación convierte interpretabilidad en una tubería reproducible: entradas versionadas, checks, contrato, model card y decisión.

Aquí aparecen dos pruebas que conviene conocer bien. En NLP se conocen como *comprehensiveness* y *sufficiency* dentro de trabajos de evaluación de racionales como ERASER.[23] En el cuaderno las adaptamos a features tabulares y a un score probabilístico: no estamos inventando una métrica nueva, estamos usando la misma idea de quitar o conservar la evidencia que la explicación dice que importa.

$$C_K(x) = \hat{p}(x) - \hat{p}(x_{\setminus S_K})$$

En palabras: quitamos las features que la explicación señala como importantes. Si el score apenas cae, la explicación no está capturando señales que sostengan la decisión.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$C_K(x)$	Comprehensiveness para las K features explicadas.	0,210094 para top-2 medio.
S_K	Conjunto de las K features superiores de la explicación.	`student_wait_days`, `missing_payment`.
$x_{\setminus S_K}$	Caso con esas features neutralizadas.	Ticket sin esas dos señales.
$\hat{p}(x)$	Score original del modelo.	0,859603.

Si quitamos las features que la explicación dice que importan y el score no cae, la explicación no está contando algo fuerte.

La suficiencia mira la pregunta contraria:

$$U_K(x) = |\hat{p}(x) - \hat{p}(x_{S_K})|$$

En palabras: dejamos solo las features que la explicación dice que importan. Si con ellas casi reconstruimos el score, la explicación es más suficiente; si no, hay señales importantes que la explicación está escondiendo.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$U_K(x)$	Diferencia entre score original y score usando solo las K features explicadas.	0,063245 para top-2 medio.
x_{S_K}	Caso donde conservamos las features explicadas y neutralizamos el resto.	Ticket reducido a sus dos señales principales.
$\hat{p}(x_{S_K})$	Score usando solo las features explicadas.	Score reconstruido desde top-2.

Si las features explicadas bastan para reconstruir casi todo el score, $U_K(x)$ será bajo. Si no bastan, quizá la explicación ha omitido una señal importante.

En la ejecución actual:

CHECK	RESULTADO	LECTURA
`deletion_top_feature_drop`	0,444993	La feature superior tiene efecto medible.
`permutation_importance_drop`	0,25	Hay features globales con impacto real.
`stability_top1`	0,9667	La explicación local es estable ante perturbación pequeña.
`comprehensiveness_top2`	0,210094	Al quitar las dos features principales, el score cae.
`sufficiency_top2`	0,063245	Las dos features principales aproximan bastante el score.
`feature_proxy_scan`	0,8837	Hay correlación alta entre `prior_cases` y `student_wait_days`; conviene revisarla.

Ese último punto es muy de ingeniería. Una correlación alta no prueba causalidad ni invalida el modelo, pero sí abre una tarea: comprobar si dos features están contando casi lo mismo, si una funciona como proxy de otra o si el dataset necesita rediseño.

Producción: deriva de explicaciones y trazas

En producción no basta con guardar predicciones. Si la explicación influye en revisión, soporte o producto, conviene guardar eventos de explicación:

```
{
  "case_id": "t001",
  "model_version": "support-prioritizer-linear-v1",
  "score": 0.859603,
  "prediction": 1,
  "top_features": ["student_wait_days", "missing_payment", "prior_cases"],
  "data_hash_sha256": "ec7bf...",
  "policy_hash_sha256": "9c167..."
}
```

Con esos eventos podemos medir deriva explicativa. Una forma sencilla es comparar la distribución de feature principal entre dos ventanas mediante distancia de variación total, una distancia estándar entre distribuciones de probabilidad.[24]

$$D_{TV}(P_t, P_{t-1}) = \frac{1}{2} \sum_f |P_t(f) - P_{t-1}(f)|$$

En palabras: sumamos cuánto cambió la proporción de cada razón principal y dividimos por dos para obtener una distancia entre 0 y 1. Si se acerca a 0, las razones dominantes se parecen; si crece, el sistema está explicando por motivos distintos.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$P_t(f)$	Proporción de casos donde la feature f fue la explicación principal en la ventana t.	`deadline_hours = 0,50`.
$P_{t-1}(f)$	La misma proporción en la ventana anterior.	`deadline_hours = 0,40`.
D_{TV}	Distancia de variación total entre distribuciones.	0 significa sin cambio agregado.

En la muestra actual, la distribución de feature principal queda así:

FEATURE PRINCIPAL	PROPORCIÓN
`deadline_hours`	0,50
`student_wait_days`	0,30
`missing_payment`	0,15
`prior_cases`	0,05

Si en la siguiente release `prior_cases` pasa de 0,05 a 0,45, no basta con decir “el accuracy sigue bien”. Algo cambió en la razón operativa de las decisiones. Puede ser un cambio de datos, un cambio de política, un error de feature engineering o una señal nueva real. Hay que investigarlo.

LLMs: explicación textual, traza y mecanismo

En modelos de lenguaje hay una confusión habitual: pedir al modelo que explique su respuesta y tratar esa explicación como mecanismo interno. Son cosas distintas.

NIVEL	QUÉ ES	QUÉ PUEDE APORTAR	QUÉ NO GARANTIZA
Explicación textual	Una justificación generada en lenguaje natural.	Puede ayudar a revisar una respuesta.	No prueba cómo se produjo la salida.
Traza operativa	Prompt, mensajes, herramientas, documentos, scores, coste y eventos.	Permite depurar una run.	No abre las capas internas del modelo.
Evidencia externa	Chunks, citas, resultados de herramientas y validaciones.	Permite comprobar afirmaciones.	No demuestra causalidad interna.
Interpretabilidad mecanicista	Análisis de activaciones, circuitos o intervenciones internas.	Puede estudiar mecanismos concretos.	Es costosa, parcial y no siempre trasladable a producto.

Para ingeniería aplicada, muchas veces la traza vale más que una explicación verbal. Si un agente consulta una herramienta, recupera un documento, cambia una respuesta y pide revisión por score bajo, eso se puede auditar. Si solo dice “he razonado cuidadosamente”, no tenemos suficiente.

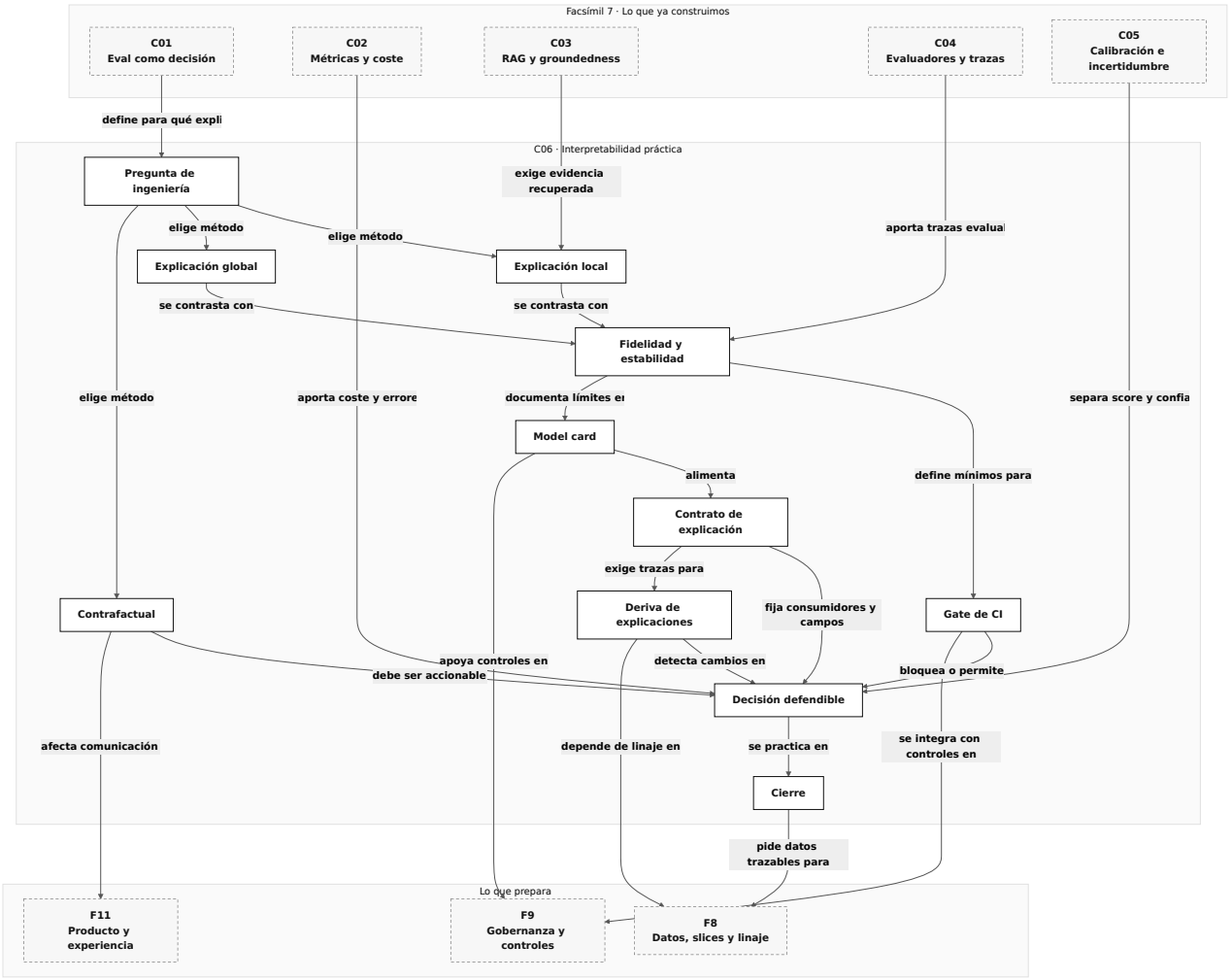
Una regla práctica para alumnos:

> En LLMs, no confundas explicación narrada con evidencia. Guarda trazas, contratos y resultados verificables.

Dónde solía tropezar yo

TROPIEZO	POR QUÉ OCURRE	ANTÍDOTO
Aceptar explicaciones porque suenan bien	La explicación textual parece convincente.	Separar plausibilidad de fidelidad.
Usar un método para todo	LIME, SHAP o saliency parecen universales.	Empezar por la pregunta de ingeniería.
Enseñar atribuciones sin pruebas	La tabla queda elegante.	Añadir borrado, permutación y estabilidad.
Proponer contrafactuales imposibles	La optimización encuentra cambios absurdos.	Filtrar por accionabilidad y aceptabilidad.
Confundir atención con explicación	Un peso de atención parece intuitivo.	Verificar si cambia la salida y si el mecanismo lo sostiene.
Olvidar los slices	La explicación global tapa segmentos malos.	Auditar por slice y por caso frontera.
No declarar quién puede usar la explicación	El mismo artefacto se usa para diagnóstico, soporte y comunicación externa.	Escribir contrato de explicación.
Dejar explicaciones fuera de CI	La release pasa métricas, pero cambia la lógica explicativa.	Añadir gate con borrado, suficiencia, estabilidad y proxies.
No vigilar deriva de razones	El modelo sigue acertando, pero decide por señales distintas.	Monitorizar distribución de top features.
Tratar proxy como causalidad	Una correlación alta parece una explicación causal.	Revisar pares de features y validar con datos o dominio.
Confundir explicación textual de LLM con mecanismo	La respuesta suena razonada.	Pedir trazas, evidencias y contratos de salida.

Cómo encaja todo



Vocabulario aprendido

TÉRMINO	DEFINICIÓN BREVE
Interpretabilidad	Capacidad de entender una decisión o comportamiento con una finalidad concreta.
Explicación local	Explicación de una predicción concreta.
Explicación global	Resumen del comportamiento general del modelo.
Fidelidad	Correspondencia entre explicación y comportamiento real del modelo.
Plausibilidad	Facilidad con la que una persona acepta una explicación como razonable.
Atribución	Reparto de una salida entre features, tokens, regiones o componentes.
LIME	Aproximación local de un modelo complejo mediante un modelo interpretable.
SHAP	Marco de atribución basado en valores de Shapley.
Integrated Gradients	Método de atribución que integra gradientes desde una línea base hasta la entrada.
Grad-CAM	Método visual que localiza regiones relevantes para una clase.
Contrafactual	Cambio mínimo de entrada que produciría otra decisión.
Prueba de borrado	Procedimiento que neutraliza una señal explicada para comprobar cuánto cambia la salida.
Importancia por permutación	Caída de métrica al desordenar una feature.
Estabilidad explicativa	Grado en que una explicación conserva sus señales principales ante perturbaciones pequeñas y razonables.
Sanity check	Prueba para detectar explicaciones que no responden al modelo o datos reales.
TCAV	Técnica que mide sensibilidad a conceptos definidos por personas.
Interpretabilidad mecanicista	Estudio de componentes internos y circuitos del modelo mediante análisis e intervenciones.
Contrato de explicación	Acuerdo técnico que fija finalidad, consumidores, campos obligatorios, linaje y usos excluidos.
Comprehensiveness	Prueba que elimina las features explicadas y comprueba cuánto cae el score.

TÉRMINO	DEFINICIÓN BREVE
Suficiencia	Prueba que conserva solo las features explicadas y comprueba cuánto se parece el score al original.
Deriva de explicaciones	Cambio temporal o entre versiones en las razones principales del modelo.
Proxy	Feature que puede representar indirectamente otra variable o mezclar señales que conviene separar.
Model reliance	Dependencia de un modelo respecto a una variable cuando el rendimiento cae al romper su asociación con la salida.
Distancia de variación total	Distancia estándar entre distribuciones de probabilidad que sirve para comparar cambios agregados de razones principales.
Recourse	Cambio accionable que una persona o equipo puede realizar para mover una decisión o resolver un caso.

Antes de pasar página

Antes de cerrar el facsímil, deberías poder responder:

1. ¿Qué diferencia hay entre interpretabilidad, explicación y transparencia?
2. ¿Por qué una explicación plausible puede ser infiel?
3. ¿Cuándo preferirías un modelo interpretable frente a explicar una caja negra?
4. ¿Qué pregunta responde LIME y qué no responde?
5. ¿Qué aporta SHAP y qué supuestos conviene revisar?
6. ¿Por qué Integrated Gradients depende de una línea base?
7. ¿Qué comprobarías antes de confiar en un mapa visual?
8. ¿Qué hace que un contrafactual sea accionable?
9. ¿Cómo usarías pruebas de borrado y permutación?
10. ¿Qué debería entrar en una model card sobre interpretabilidad?
11. ¿Qué campos mínimos pondrías en un contrato de explicación?
12. ¿Qué diferencia hay entre comprehensiveness y suficiencia?
13. ¿Qué señal te daría una deriva de explicaciones?
14. ¿Por qué una correlación alta entre features puede pedir revisión?
15. ¿Qué artefactos debería producir una auditoría de interpretabilidad?
16. ¿Cómo conecta interpretabilidad con EvalOps y calibración?

En resumen

IDEA	QUÉ TE LLEVAS
Interpretar es responder una pregunta situada.	No existe “explicación general” útil para todo.
Plausibilidad no basta.	Una explicación debe probar fidelidad, estabilidad y utilidad.
Cada método tiene límites.	LIME, SHAP, Grad-CAM, contrafactuales y TCAV responden cosas distintas.
Los contrafactuales necesitan criterio humano.	Mínimo no significa accionable ni aceptable.
Las explicaciones se documentan.	Model card, datos, thresholds, checks y decisión.
Las explicaciones también tienen contrato.	Deben declarar finalidad, consumidores, linaje y campos obligatorios.
Las explicaciones se testean.	CI puede revisar borrado, suficiencia, estabilidad, proxies y contrafactuales.
Las explicaciones pueden derivar.	En producción conviene vigilar qué razones dominan cada versión.
El facsímil se cierra con criterio.	El cierre integra métricas, RAG, evaluadores, calibración e interpretación en una sola decisión.

Cuadernos para practicar

Has evaluado sistemas a lo largo del facsímil; estos cuadernos te dejan tocar las dos decisiones que más se equivocan: fiarte de una probabilidad y elegir un umbral. Son *notebooks* que se abren en Google Colab —gratis, en el navegador— o te puedes descargar. Cada uno, explicado paso a paso, con salidas reales, y anclado al capítulo del que sale.

Calibración: cuando el modelo dice «90%», ¿es un 90%?

Qué practicas: medir si las probabilidades de un modelo significan lo que dicen, y corregirlas. **Dónde encaja:** capítulo 5 (calibración e incertidumbre: de *scores* a decisiones). **Qué necesitas:** un navegador. Corre en CPU; sin claves.

Entrenas un modelo que acierta bastante y luego haces la otra pregunta, la que casi nadie hace: ¿son fiables sus porcentajes? Con un *reliability diagram* comparas lo prometido con lo cumplido y resumes el desajuste en un número, el ECE, que aquí sale

en **0,092** (un modelo optimista que promete más acierto del que cumple). Después lo corriges con *temperature scaling* —dividir los *logits* por una temperatura ajustada— y el ECE baja a **0,025** sin cambiar ni una predicción: solo recalibras la confianza.

Si decides con un umbral («bloquea si supera 0,8»), más te vale que ese 0,8 sea de verdad un 80%. Un modelo descalibrado se equivoca con total aplomo.

► **Abrir en Google Colab** ↓ **Descargar el cuaderno (.ipynb)**

El coste del error: dónde poner el umbral

Qué practicas: elegir el umbral de decisión por dinero, no por defecto. **Dónde encaja:** capítulo 2 (métricas clásicas: matriz de confusión y coste del error). **Qué necesitas:** un navegador. Corre en CPU; sin claves.

Montas un detector de fraude (clase rara, 8%) y le pones precio a cada error: un fraude que se cuela cuesta 100 €, una transacción legítima bloqueada solo 5. Recorres todos los umbrales y dibujas la curva del coste. El umbral por defecto de 0,5 cuesta **7.735 €**; el óptimo está en **0,07** —lejísimos de 0,5— y cuesta **4.780 €**: casi **3.000 € de ahorro** por mover un número. Como dejar pasar un fraude es veinte veces más caro que una falsa alarma, conviene ser desconfiado.

El acierto es mala guía cuando una clase es rara y un error cuesta más que otro. La métrica honesta es el coste, y el umbral es la palanca: casi nunca vale 0,5.

► **Abrir en Google Colab** ↓ **Descargar el cuaderno (.ipynb)**

Recursos para seguir: leer, construir y experimentar

Evaluar es decidir. Has visto métricas clásicas, evaluación de RAG, jueces LLM, calibración e interpretabilidad. Aquí tienes por dónde seguir practicándolo.

Para experimentar sin código. En las cajas «Pruébalo en 5 minutos» lo tocaste: MLU-Explain de Amazon (`mlu-explain.github.io`) para mover el umbral de una matriz de confusión y ver precisión y recall pelearse; y los playgrounds (`platform.openai.com` , `aistudio.google.com` , `console.anthropic.com`) para comprobar a mano la fidelidad de un RAG, el ruido y los sesgos de un juez LLM, y la calibración de las confianzas de un modelo.

Para construir. Para evaluar en serio: OpenAI Evals y Promptfoo para montar suites de casos y comparar prompts y modelos; Ragas para métricas de RAG; y plataformas como LangSmith para datasets, trazas y comparación de runs. Empieza con un runner propio pequeño para entender el contrato antes de adoptar una plataforma.

Para leer. Cada capítulo enlaza sus fuentes en «Para saber más»; las ideas clave son la matriz de confusión y el coste del error, la fidelidad y la abstención en RAG, y la calibración. Convertir todo eso en un gate de release defendible es lo que separa «la métrica salió alta» de «puedo justificar por qué publico o bloqueo».

Para saber más

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M. y Kim, B. (2018). Sanity Checks for Saliency Maps. *Advances in Neural Information Processing Systems*. https://papers.nips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html

Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5-32. https://doi.org/10.1023/A:1010933404324

Captum. (2026). *Model Interpretability for PyTorch*. https://captum.ai/

Cover, T. M. y Thomas, J. A. (2006). *Elements of Information Theory* (2.ª ed.). Wiley.

DeYoung, J., Jain, S., Rajani, N. F., Lehman, E., Xiong, C., Socher, R. y Wallace, B. C. (2020). ERASER: A Benchmark to Evaluate Rationalized NLP Models. *Proceedings of ACL*. https://aclanthology.org/2020.acl-main.408/

Doshi-Velez, F. y Kim, B. (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. arXiv. https://arxiv.org/abs/1702.08608

Fisher, A., Rudin, C. y Dominici, F. (2019). All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, 20(177), 1-81. https://jmlr.org/papers/v20/18-760.html

Hastie, T., Tibshirani, R. y Friedman, J. (2009). *The Elements of Statistical Learning* (2.ª ed.). Springer. https://web.stanford.edu/~hastie/ElemStatLearn/

InterpretML. (2026). *InterpretML documentation*. https://interpret.ml/

Jacovi, A. y Goldberg, Y. (2020). Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness? *Proceedings of ACL*. https://aclanthology.org/2020.acl-main.386/

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viégas, F. y Sayres, R. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors. *ICML*. https://arxiv.org/abs/1711.11279

Lipton, Z. C. (2018). The Mythos of Model Interpretability. *Communications of the ACM*, 61(10), 36-43. https://doi.org/10.1145/3233231

Lundberg, S. M. y Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*. https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions

Meng, K., Bau, D., Andonian, A. y Belinkov, Y. (2022). Locating and Editing Factual Associations in GPT. *Advances in Neural Information Processing Systems*. https://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D. y Gebru, T. (2019). Model Cards for Model Reporting. *FAT*, 220-229. https://doi.org/10.1145/3287560.3287596

Ribeiro, M. T., Singh, S. y Guestrin, C. (2016). Why Should I Trust You? Explaining the Predictions of Any Classifier. *KDD*, 1135-1144. https://doi.org/10.1145/2939672.2939778

Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1, 206-215. https://doi.org/10.1038/s42256-019-0048-x

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. y Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *ICCV*. https://doi.org/10.1109/ICCV.2017.74

SHAP. (2026). *SHAP documentation*. https://shap.readthedocs.io/

scikit-learn. (2026). *Permutation feature importance*. https://scikit-learn.org/stable/modules/permutation_importance.html

Seldon. (2026). *Alibi Explain documentation*. https://docs.seldon.ai/alibi-explain

Sundararajan, M., Taly, A. y Yan, Q. (2017). Axiomatic Attribution for Deep Networks. *ICML*, 3319-3328. https://proceedings.mlr.press/v70/sundararajan17a.html

Tabassi, E. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.100-1

Wachter, S., Mittelstadt, B. y Russell, C. (2017). Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law and Technology*, 31, 841-887. https://arxiv.org/abs/1711.00399

Notas

1. Lipton, Z. C. (2018). The Mythos of Model Interpretability. *Communications of the ACM*, 61(10), 36-43. https://doi.org/10.1145/3233231 ↩

2. Doshi-Velez, F. y Kim, B. (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. arXiv. https://arxiv.org/abs/1702.08608 ↩

3. Jacovi, A. y Goldberg, Y. (2020). Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness? *ACL*. https://aclanthology.org/2020.acl-main.386/ ↩

4. Ribeiro, M. T., Singh, S. y Guestrin, C. (2016). Why Should I Trust You? Explaining the Predictions of Any Classifier. *KDD*, 1135-1144. https://doi.org/10.1145/2939672.2939778 ↩

5. Lundberg, S. M. y Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *NeurIPS*. https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions ↩

6. Sundararajan, M., Taly, A. y Yan, Q. (2017). Axiomatic Attribution for Deep Networks. *ICML*, 3319-3328. https://proceedings.mlr.press/v70/sundararajan17a.html ↩

7. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. y Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *ICCV*. https://doi.org/10.1109/ICCV.2017.74 ↩

8. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M. y Kim, B. (2018). Sanity Checks for Saliency Maps. *NeurIPS*. https://papers.nips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html ↩

9. Wachter, S., Mittelstadt, B. y Russell, C. (2017). Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law and Technology*, 31, 841-887. https://arxiv.org/abs/1711.00399 ↩

10. Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1, 206-215. https://doi.org/10.1038/s42256-019-0048-x ↩

11. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viégas, F. y Sayres, R. (2018). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors. *ICML*. https://arxiv.org/abs/1711.11279 ↩

12. Meng, K., Bau, D., Andonian, A. y Belinkov, Y. (2022). Locating and Editing Factual Associations in GPT. *NeurIPS*. https://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html ↩

.3. Hastie, T., Tibshirani, R. y Friedman, J. (2009). *The Elements of Statistical Learning* (2.ª ed.). Springer. https://web.stanford.edu/~hastie/ElemStatLearn/ ↩

.4. Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5-32. https://doi.org/10.1023/A:1010933404324 ↩

.5. Fisher, A., Rudin, C. y Dominici, F. (2019). All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, 20(177), 1-81. https://jmlr.org/papers/v20/18-760.html ↩

.6. Wachter, S., Mittelstadt, B. y Russell, C. (2017). Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law and Technology*, 31, 841-887. https://arxiv.org/abs/1711.00399 ↩

.7. SHAP Documentation. (2026). https://shap.readthedocs.io/ ↩

.8. Ribeiro, Singh y Guestrin, 2016. ↩

.9. Captum. (2026). Model Interpretability for PyTorch. https://captum.ai/ ↩

:0. InterpretML. (2026). https://interpret.ml/ ↩

:1. Seldon. (2026). Alibi Explain documentation. https://docs.seldon.ai/alibi-explain ↩

:2. scikit-learn. (2026). Permutation feature importance. https://scikit-learn.org/stable/modules/permutation_importance.html ↩

:3. DeYoung, J., Jain, S., Rajani, N. F., Lehman, E., Xiong, C., Socher, R. y Wallace, B. C. (2020). ERASER: A Benchmark to Evaluate Rationalized NLP Models. *Proceedings of ACL*. https://aclanthology.org/2020.acl-main.408/ ↩

:4. Cover, T. M. y Thomas, J. A. (2006). *Elements of Information Theory* (2.ª ed.). Wiley. ↩