Facsímil 07

Evaluar, calibrar e interpretar

Capítulo 02

Métricas clásicas: matriz de confusión y coste del error

686f6c61 · 2026

Capítulo 02: Métricas clásicas: matriz de confusión y coste del error

Entrando en el tema

En el capítulo anterior construimos una eval como expediente: hipótesis, casos, unidad de evaluación, scorecard y decisión. Ahora necesitamos una herramienta mucho más humilde y, precisamente por eso, imprescindible: la matriz de confusión.

La matriz de confusión no es una tabla para rellenar por costumbre. Es la contabilidad mínima de una decisión automática. Si tu sistema marca un ticket como urgente, acepta un documento, bloquea una operación, recomienda revisión humana o deja pasar una alerta, tienes cuatro preguntas antes de presumir de métrica:

PREGUNTA	QUÉ TE OBLIGA A MIRAR
¿Cuántas veces acertó cuando debía actuar?	Verdaderos positivos.
¿Cuántas veces actuó cuando no debía?	Falsos positivos.
¿Cuántas veces dejó pasar algo importante?	Falsos negativos.
¿Cuántas veces descartó bien lo que no importaba?	Verdaderos negativos.

Al terminar este capítulo deberías poder hacer esto:

RESULTADO DE APRENDIZAJE	EVIDENCIA DE QUE LO SABES HACER
Leer una matriz de confusión.	Distingues TP, FP, FN y TN sin memorizar siglas vacías.
Calcular métricas básicas.	Obtienes accuracy, precision, recall, specificity, F1, F-beta y balanced accuracy.
Elegir métrica según decisión.	No optimizas accuracy si el error importante vive en una clase minoritaria.
Traducir métricas a coste.	Asignas coste a FP, FN y revisión antes de elegir umbral.
Comparar umbrales.	Escaneas umbrales y eliges por coste, cobertura y restricciones.
Defender una política.	Produces una scorecard con matriz, coste, slices, zona de revisión y decisión escrita.

La idea central es esta: **una métrica no es una medalla; es una forma comprimida de hablar de consecuencias**.

Conviene insistir en esto porque, en proyectos reales, la métrica suele llegar demasiado pronto a la conversación. Alguien pregunta “¿qué F1 tenemos?” antes de haber fijado qué clase es positiva, qué casos entran en evaluación, qué ocurre con los ambiguos, quién revisa los errores y qué decisión se tomará si el número baja. Esa prisa produce evaluaciones aparentemente rigurosas, pero débiles: llenas de decimales, vacías de criterio.

En este capítulo vamos a usar métricas clásicas, sí, pero no como una lista de definiciones. Las vamos a tratar como instrumentos de lectura. Accuracy te dice una cosa, precision otra, recall otra, specificity otra, F1 otra. Ninguna de ellas sabe por sí sola si debes publicar, bloquear, revisar o cambiar el diseño. Esa decisión aparece cuando conectas la métrica con el coste del error, la capacidad de revisión y el contexto donde vive el sistema.

El problema: una accuracy alta puede ser una mala noticia

Imagina un clasificador que decide si un ticket de soporte debe tratarse como urgente. De cada 100 tickets, solo 10 son realmente urgentes. Un sistema perezoso podría decir “ninguno es urgente” y acertar 90 veces.

Eso le da 90 % de accuracy.

Y aun así sería un desastre operativo, porque no detecta ni un ticket urgente. La cifra global queda bonita, pero la decisión es inútil. Este ejemplo aparece una y otra vez en ingeniería de IA: fraude raro, incidentes raros, documentos peligrosos raros, bugs críticos raros, abuso raro. Lo que importa suele ser minoritario.

Por eso este capítulo no va de coleccionar métricas. Va de aprender a hacer una pregunta mejor:

> ¿Qué error puedo tolerar, cuál no, y qué decisión cambia cuando miro la matriz?

Soporte, prevalencia y clase positiva

Antes de calcular nada hay que fijar tres piezas que parecen pequeñas y no lo son: **qué clase llamas positiva**, cuántos casos tiene cada clase y cuál es la frecuencia base del fenómeno que buscas. En documentación de métricas, `support` suele significar cuántas muestras reales hay de cada clase. Si tienes 1.000 tickets y solo 30 urgentes, el soporte de la clase positiva es 30. Esa cifra condiciona toda la lectura.

La **prevalencia** o frecuencia base es la proporción habitual de positivos en el conjunto que evalúas o en producción. En un clasificador de urgencias, si el 3 % de los tickets son urgentes, un sistema que marque el 40 % como urgente quizá tenga recall alto, pero puede ser inviable para soporte. En detección de fraude, si el positivo real es rarísimo, una tasa pequeña de falsos positivos puede generar miles de revisiones. En moderación, si cambian las campañas de abuso, la prevalencia puede moverse de una semana a otra y dejar obsoleto un umbral que ayer parecía sensato.

También hay que decidir qué significa “positivo”. Parece obvio, pero no siempre lo es. En un filtro de seguridad, positivo puede ser “riesgoso”; en un sistema médico, positivo puede ser “sospecha”; en soporte, positivo puede ser “urgente”; en recuperación de documentos, positivo puede ser “relevante”. Cambiar esa convención cambia la matriz, cambia precision y recall, y cambia la conversación con el equipo. Si no lo escribes, dos personas pueden mirar los mismos resultados y discutir porque están imaginando positivos distintos.

PIEZA	PREGUNTA CONCRETA	ERROR TÍPICO
Clase positiva	¿Qué evento quiero detectar o activar?	Llamar positivo a lo cómodo, no a lo importante.
Soporte	¿Cuántos casos reales tengo de cada clase?	Concluir demasiado con 8 positivos.
Prevalencia	¿Con qué frecuencia aparece el positivo en producción?	Evaluar con una muestra artificial y esperar el mismo comportamiento en tráfico real.
Unidad	¿Evalúo ticket, documento, respuesta, sesión o release?	Mezclar casos que no deberían compartir métrica.

Esta sección parece previa a las métricas, pero en realidad es parte de la métrica. Una precision del 80 % no significa lo mismo si has evaluado 10 positivos que si has evaluado 10.000. Un recall del 95 % no significa lo mismo si el 5 % que pierdes son tickets de baja prioridad que si son incidentes de producción. La matriz de confusión empieza antes de dibujar la tabla.

Qué sí es una matriz de confusión

Una matriz de confusión cruza dos cosas:

1. La realidad revisada.
2. La decisión del sistema.

Para clasificación binaria:

	PREDICE POSITIVO	PREDICE NEGATIVO
Real positivo	TP	FN
Real negativo	FP	TN

En un sistema de tickets urgentes:

SÍMBOLO	SIGNIFICADO	EJEMPLO
TP	Verdadero positivo.	Ticket urgente marcado como urgente.
FP	Falso positivo.	Ticket normal marcado como urgente.
FN	Falso negativo.	Ticket urgente marcado como normal.
TN	Verdadero negativo.	Ticket normal marcado como normal.

La matriz obliga a hacer una pregunta adulta: **qué error duele más**. En spam quizá un falso positivo duele mucho porque pierdes un correo importante. En incidentes de producción quizá duele más un falso negativo porque se te escapa una caída. En moderación, salud, pagos o educación, la respuesta depende del contexto, no de la métrica favorita del equipo.

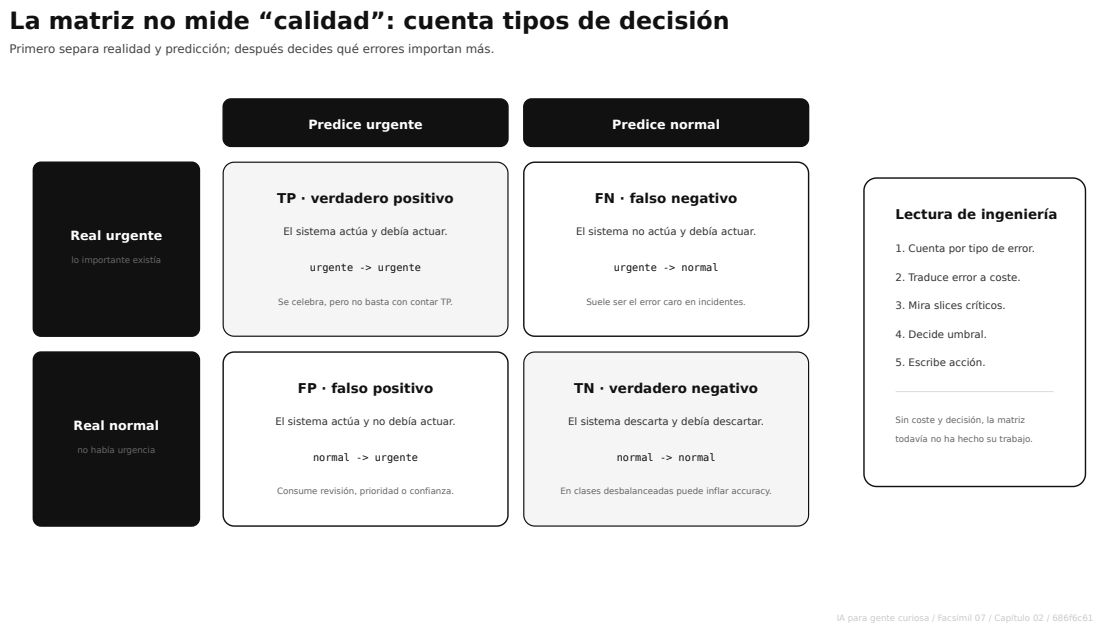
Para leerla con algo de oficio, no mires primero el número grande. Mira la diagonal y luego mira los errores. La diagonal, TP y TN, son aciertos. Los otros dos cuadrantes, FP y FN, son decisiones incorrectas. Pero el peso técnico no está repartido por igual. Si un FN deja pasar

una incidencia grave, ese cuadrante puede mandar más que todos los TN juntos. Si un FP bloquea una transferencia legítima o manda a revisión a demasiadas personas, ese otro cuadrante puede convertirse en coste operativo.

Con números, una matriz podría leerse así:

	PREDICE URGENTE	PREDICE NORMAL
Real urgente	18	4
Real normal	7	71

Una lectura floja diría: “hay 89 aciertos sobre 100”. Una lectura de ingeniería diría algo más parecido a esto: “detectamos 18 de 22 urgentes, perdemos 4 urgentes, generamos 7 falsas urgencias y descartamos bien 71 normales. Ahora hay que saber si perder 4 urgentes es aceptable, cuánto cuesta revisar 7 falsos positivos y si esos 4 falsos negativos pertenecen a un slice crítico”. La segunda lectura es menos cómoda, pero es la que sirve.



> La matriz de confusión convierte una clasificación en una contabilidad de errores. A partir de ahí empieza la decisión de ingeniería.

Fecha de corte del estado del arte

Fecha de corte: 28 de mayo de 2026.

La parte estable de este capítulo viene de métricas clásicas de clasificación y evaluación de clasificadores. scikit-learn documenta métricas como accuracy, balanced accuracy, precision, recall, F-measure, ROC AUC, matriz de confusión y reportes de clasificación con APIs reproducibles.[1] La función `confusion_matrix` cuenta observaciones reales frente a predichas por clase.[2] `classification_report` resume precision, recall, F1 y soporte por clase.[3]

Fawcett presentó ROC como herramienta para visualizar el equilibrio entre tasa de verdaderos positivos y tasa de falsos positivos al mover el umbral.[4] Davis y Goadrich explicaron la relación entre ROC y precision-recall, y por qué ambas vistas no son intercambiables sin más.[5] Saito y Rehmsmeier mostraron que, en datasets desbalanceados, la curva precision-recall suele ser más informativa que ROC para evaluar clasificadores binarios.[6]

Para coste sensible, la idea no es inventar una ecuación decorativa: viene de aprendizaje con costes y teoría de decisión aplicada a clasificación. Elkan formuló los fundamentos de *cost-sensitive learning* y Domingos propuso MetaCost como método general para hacer clasificadores sensibles al coste.[7][8]

La parte que cambia por proyecto es el coste real del error, la capacidad de revisión, el umbral y la decisión que quieres automatizar. Ahí no hay tabla universal: hay que medir, discutir con negocio/operación y dejar evidencia.

Las métricas básicas, con símbolos claros

Partimos de esta identidad básica:

$$N = TP + FP + FN + TN$$

En palabras: el total de casos evaluados es la suma de los cuatro cuadrantes de la matriz.

SÍMBOLO	SIGNIFICADO	EJEMPLO
N	Número total de casos evaluados.	100 tickets.
TP	Positivos reales predichos como positivos.	18 urgentes detectados.
FP	Negativos reales predichos como positivos.	7 normales marcados urgentes.
FN	Positivos reales predichos como negativos.	4 urgentes perdidos.
TN	Negativos reales predichos como negativos.	71 normales bien clasificados.

La **accuracy** mide proporción total de aciertos:

$$accuracy = \frac{TP + TN}{N}$$

En palabras: cuenta todas las decisiones correctas y las divide por todos los casos.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$accuracy$	Aciertos totales sobre casos totales.	$(18 + 71)/100 = 0,89$.
$TP + TN$	Decisiones correctas.	89.
N	Total de casos.	100.

Accuracy es cómoda porque resume todo en una proporción fácil de comunicar. Por eso se usa tanto, y por eso también engaña tanto. Si las clases están equilibradas y FP y FN cuestan parecido, puede ser una primera lectura razonable. Si una clase domina el dataset, accuracy se deja arrastrar por esa clase. En el ejemplo de tickets, 71 TN pesan muchísimo; pueden hacer que el sistema parezca bueno aunque los 4 FN sean justo lo que más te importaba.

Una forma práctica de usar accuracy sin caer en la trampa es leerla siempre junto al soporte de cada clase. No digas “89 % de accuracy” sin decir también cuántos positivos reales había. En revisión técnica, una frase más honesta sería: “89 % de accuracy sobre 100 casos, con 22 urgentes reales y 4 urgentes perdidos”. Esa segunda frase ya no permite esconder el problema detrás del porcentaje.

La **precision** responde: de lo que marqué como positivo, ¿cuánto lo era de verdad?

$$precision = \frac{TP}{TP + FP}$$

En palabras: mide la pureza de las alarmas positivas.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$precision$	Proporción de positivos predichos que eran positivos reales.	$18/(18 + 7) = 0,72$.
TP	Positivos correctos.	18.
FP	Positivos que no lo eran.	7.

Precision es la métrica que mira el ruido de tus positivos predichos. Si el sistema dice “esto es urgente” 25 veces y 18 lo eran, precision es 0,72. En lenguaje operativo: de cada 100 casos que mando a la cola urgente, 72 deberían estar ahí y 28 son trabajo extra. Esto importa

mucho cuando cada positivo predicho dispara una acción: revisión humana, bloqueo, alerta, correo, llamada, escalado, retención de pago o intervención manual.

Precision no te dice cuántos positivos reales se te escaparon. Puedes tener precision perfecta si solo marcas como urgente los casos obvios y dejas pasar todos los difíciles. Por eso una precision alta puede ser buena o puede ser cobarde: depende de si también estás encontrando lo importante.

El **recall** responde: de los positivos reales, ¿cuántos encontré?

$$recall = \frac{TP}{TP + FN}$$

En palabras: mide cobertura de lo importante.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$recall$	Proporción de positivos reales detectados.	$18/(18+4) = 0,82$.
TP	Positivos encontrados.	18.
FN	Positivos perdidos.	4.

Recall es la métrica que mira cobertura. En un sistema de urgencias, incidentes, fraude, seguridad o salud, suele ser la primera métrica que pone nervioso al equipo porque pregunta: “de lo que de verdad importaba, ¿cuánto he encontrado?”. Si el recall es 0,82, encontraste 18 de 22 urgentes y perdiste 4. El número no dice todavía si 4 es tolerable; eso lo decide el coste, la severidad y el slice.

Recall tampoco es gratis. Puedes subirlo bajando el umbral y marcando más casos como positivos. Eso detectará más urgentes, pero probablemente aumentará falsos positivos y revisión. En un sistema real, perseguir recall máximo sin mirar coste puede convertir una automatización en una fábrica de interrupciones.

La **specificity** responde: de los negativos reales, ¿cuántos dejé como negativos?

$$specificity = \frac{TN}{TN + FP}$$

En palabras: mide cobertura de negativos reales, algo muy útil cuando quieres controlar falsas alarmas.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$specificity$	Proporción de negativos reales descartados correctamente.	$71/(71+7) = 0,91$.
TN	Negativos correctos.	71.
FP	Negativos marcados como positivos.	7.

Specificity suele recibir menos cariño que recall, pero en sistemas con muchas falsas alarmas es decisiva. Si el equipo se queja de que “el sistema avisa demasiado”, muchas veces está hablando de specificity sin nombrarla. Una specificity baja significa que muchos negativos reales se convierten en positivos predichos. En soporte, eso llena colas; en seguridad, produce fatiga de alertas; en producto, rompe confianza.

La tensión clásica aparece aquí: subir recall puede bajar specificity, y subir specificity puede bajar recall. No hay una métrica “buena” en abstracto. Hay una frontera de decisión que mueve trabajo entre positivos perdidos, falsas alarmas y revisión.

F1 resume precision y recall con media armónica:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

En palabras: F1 cae si una de las dos piezas, precision o recall, cae mucho. Por eso se usa como resumen cuando quieres equilibrarlas.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$F1$	Equilibrio entre precision y recall.	$2 \cdot 0,72 \cdot 0,82/(0,72+0,82) = 0,77$.
$precision$	Pureza de positivos predichos.	0,72.
$recall$	Cobertura de positivos reales.	0,82.

F1 es útil porque penaliza desequilibrios fuertes entre precision y recall. Si una de las dos cae mucho, F1 cae. Eso evita que alguien enseñe solo la métrica que le favorece. Pero F1 sigue siendo un resumen estadístico, no una política. No sabe si un FN cuesta 12 veces más que un FP. No sabe si tienes personas suficientes para revisar la zona gris. No sabe si el slice de pagos pesa más que el de consultas generales.

Por eso F1 sirve bien como indicador de lectura comparativa, especialmente cuando quieres comparar variantes bajo un mismo protocolo. Sirve peor como criterio único de publicación. Si una release se aprueba solo porque F1 sube, sin mirar matriz, coste y slices, la evaluación está incompleta.

F-beta permite dar más peso a recall o a precision:

$$F_\beta = \frac{(1+\beta^2)\cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$$

En palabras: si $\beta > 1$, recall pesa más; si $\beta < 1$, precision pesa más.

SÍMBOLO	SIGNIFICADO	EJEMPLO
F_β	F-score con peso ajustable.	F_2 prioriza recall.
β	Peso relativo de recall frente a precision.	$\beta = 2$.
$precision$	Pureza de positivos predichos.	0,72.
$recall$	Cobertura de positivos reales.	0,82.

F-beta es una forma más honesta de reconocer una preferencia. Si el problema castiga mucho perder positivos, puedes usar F_2 para dar más peso a recall. Si el problema castiga mucho molestar con falsas alarmas, puedes usar un beta menor que 1 para dar más peso a precision. La clave es no elegir beta porque “queda avanzado”, sino porque expresa una decisión de dominio.

Powers revisa precision, recall, F-measure y medidas relacionadas como informedness, markedness y correlación, útiles para no reducir la evaluación a una sola cifra sin contexto.[9]

Qué aporta y qué no aporta cada métrica

Una forma práctica de no perderse es separar cada métrica por pregunta, utilidad y límite. Esta tabla no sustituye las fórmulas; ayuda a decidir cuándo una fórmula te está contando algo útil y cuándo te está distrayendo.

MÉTRICA	PREGUNTA QUE RESPONDE	ÚTIL CUANDO	NO TE CUENTA
Accuracy	¿Qué proporción total acerté?	Clases equilibradas y errores parecidos.	Si una clase minoritaria crítica está fallando.
Precision	¿Cuánto ruido hay entre mis positivos predichos?	Cada positivo dispara trabajo, bloqueo o alerta.	Cuántos positivos reales se escaparon.
Recall	¿Cuánto de lo importante encontré?	Perder positivos es caro o peligroso.	Cuántas falsas alarmas generaste.
Specificity	¿Cuánto de lo normal descarté bien?	Te preocupa saturar colas o producir fatiga.	Si cubres bien la clase positiva.
F1	¿Cómo equilibran precision y recall?	Comparas variantes bajo el mismo protocolo.	Coste, capacidad, severidad y slices.
F-beta	¿Quiero inclinar el resumen hacia precision o recall?	Hay una preferencia explícita de dominio.	La justificación de esa preferencia.

Si tienes que elegir una sola frase para una revisión técnica, que sea esta: “la métrica que enseño responde a esta pregunta y no responde a estas otras”. Esa frase ahorra muchos malentendidos.

Accuracy, balanced accuracy y clases desbalanceadas

Cuando hay muchas más clases negativas que positivas, `accuracy` puede ser complaciente. Si detectas spam, fraude, tickets urgentes o documentos peligrosos, la clase que te importa puede ser pequeña. En esos casos, un sistema que no hace nada puede parecer bueno.

Una alternativa sencilla es balanced accuracy:

$$balanced\ accuracy = \frac{recall + specificity}{2}$$

En palabras: da el mismo peso a la capacidad de encontrar positivos y a la capacidad de descartar negativos.

SÍMBOLO	SIGNIFICADO	EJEMPLO
$balanced\ accuracy$	Media entre cobertura positiva y negativa.	$(0,82 + 0,91)/2 = 0,865$.
$recall$	Tasa de positivos encontrados.	0,82.
$specificity$	Tasa de negativos bien descartados.	0,91.

Balanced accuracy evita que una clase mayoritaria tape la lectura de la minoritaria, pero sigue sin saber cuánto cuesta cada error. Por eso no cierra el problema: solo lo lee con algo más de justicia estadística.

En ingeniería conviene leer balanced accuracy como una alarma contra el autoengaño. Si accuracy es alta y balanced accuracy cae, probablemente estás beneficiándote de la clase mayoritaria. Si ambas son altas, todavía no has terminado: falta mirar coste, intervalos de confianza, slices y estabilidad temporal. Un buen capítulo de evaluación no termina en “sube la métrica”, sino en “esta métrica sube, bajo estas condiciones, con estos límites, y por eso puedo o no puedo tomar esta decisión”.

Coste sensible: cuando FP y FN no cuestan lo mismo

El paso de métrica a ingeniería empieza cuando escribes una matriz de costes. No porque los costes didácticos sean perfectos, sino porque obligan al equipo a verbalizar sus prioridades.

Un coste no tiene por qué ser dinero exacto. Puede representar minutos de revisión, puntos de riesgo, impacto en SLA, fricción de usuario, carga de soporte o severidad operacional. Lo importante es que sea explícito y discutible. Si el equipo dice “un falso negativo es grave”, todavía no hay política. Si dice “un falso negativo pesa seis veces más que un falso positivo y la revisión cuesta menos que ambos errores”, ya puedes comparar umbrales de una forma auditable.

Esta conversación suele ser incómoda porque saca a la luz desacuerdos reales. Producto quizá quiere automatizar más. Operaciones quizá quiere menos falsas alarmas. Riesgo quizá prefiere revisar cualquier caso dudoso. Datos quizá avisa de que la muestra es pequeña. La matriz de costes no elimina el desacuerdo, pero lo convierte en algo que se puede probar en una scorecard.

En clasificación sensible al coste, una forma estándar de expresar el riesgo empírico de un clasificador es:

$$R(h) = \frac{1}{n} \sum_{i=1}^{n} C(y_i, h(x_i))$$

En palabras: calculas el coste que produce el clasificador h en cada caso evaluado y promedias sobre la muestra. Esta idea pertenece al marco de aprendizaje sensible al coste, no a una ocurrencia del capítulo.[1011]

SÍMBOLO	SIGNIFICADO	EJEMPLO
$\widehat{R}(h)$	Riesgo empírico sensible al coste del clasificador.	Coste medio por ticket evaluado.
n	Número de casos evaluados.	100 tickets.
x_i	Entrada del caso i.	Texto y metadatos de un ticket.
y_i	Etiqueta real revisada del caso i.	Urgente o normal.
$h(x_i)$	Predicción del clasificador.	Urgente o normal.
$C(y_i, h(x_i))$	Coste de predecir $h(x_i)$ cuando la realidad es y_i.	12 si pierdes un urgente; 2 si generas falsa urgencia.

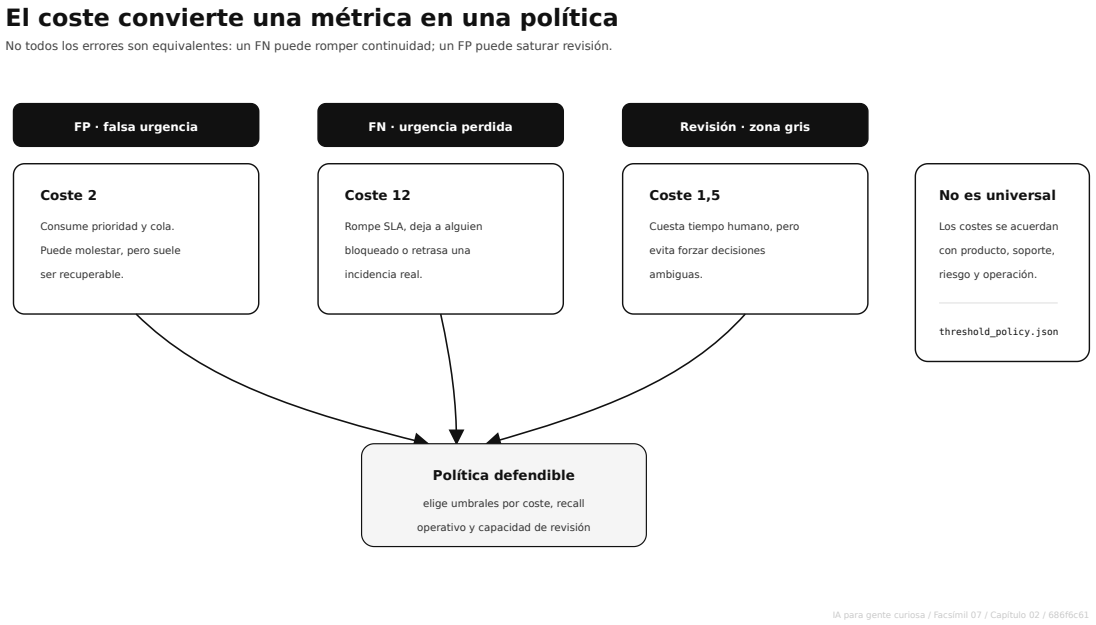
En el caso binario, si solo contamos falsos positivos y falsos negativos, esta lectura se traduce de forma práctica así:

ERROR	PREGUNTA DE NEGOCIO	COSTE DIDÁCTICO
Falso positivo	¿Qué pasa si trato un ticket normal como urgente?	2
Falso negativo	¿Qué pasa si trato un ticket urgente como normal?	12
Revisión	¿Qué cuesta que una persona mire la zona gris?	1,5

No voy a convertir la revisión humana en una fórmula nueva del facsímil. Es una política operativa: cuentas cuántos casos mandas a revisar, multiplicas por el coste acordado y comparas políticas. Lo importante es que esa decisión se pueda auditar.

En el cuaderno del facsímil, los costes son didácticos: falso positivo 2, falso negativo 12 y revisión 1,5. No significan euros reales. Significan relación de prioridades: perder un urgente pesa mucho más que crear una falsa urgencia, y revisar un caso ambiguo es más barato que equivocarse automáticamente. En un proyecto real, esos valores se deberían estimar con datos históricos y conversación de dominio. ¿Cuántos minutos cuesta revisar? ¿Cuánto cuesta incumplir un SLA? ¿Qué pasa si un falso positivo bloquea una acción legítima? ¿Qué coste reputacional o legal tiene cada error?

También conviene separar coste de severidad. Dos errores pueden tener el mismo tipo, pero no la misma gravedad. Un falso negativo en una consulta menor no pesa igual que un falso negativo en acceso durante una entrega. Por eso el dataset de práctica incluye `slice` y `severity`: no para adornar el JSON, sino para recordarte que una matriz global puede mezclar daños muy distintos.



> Coste sensible no significa poner números al azar: significa explicitar qué consecuencias acepta o no acepta tu sistema.

Umbrales: mover la frontera cambia el sistema

Un clasificador suele devolver un score. El umbral convierte ese score en una acción. Para no meter una fórmula innecesaria, pensemos en pseudocódigo:

```
si score >= umbral:
    decidir "urgente"
si no:
    decidir "normal"
```

Si subes el umbral, normalmente marcas menos casos como positivos. Eso puede subir precision, pero baja recall. Si bajas el umbral, detectas más positivos, pero generas más falsos positivos. No hay magia: estás moviendo trabajo entre errores, automatización y revisión.

El umbral no debería elegirse en el test final. Lo normal es usar un conjunto de validación para explorar umbrales y reservar un test final para estimar cómo se comporta la política ya elegida. Si tanteas umbrales mirando el test una y otra vez, conviertes el test en parte del entrenamiento de la decisión. El número final parecerá más sólido de lo que es.

También hay que evitar otro malentendido: un score alto no siempre es una probabilidad calibrada. Un score de 0,80 puede significar “este caso está muy arriba en el ranking”, pero no necesariamente “hay un 80 % de probabilidad real”. Para elegir umbrales puede bastar con que el score ordene bien; para interpretar riesgo como probabilidad necesitas calibración, que veremos en el capítulo 05. Mezclar esas dos cosas es una fuente clásica de errores en sistemas de IA.

En sistemas reales, muchas veces conviene usar dos umbrales:

```
si score <= umbral_bajo:
    decidir "normal"
si score >= umbral_alto:
    decidir "urgente"
si no:
    enviar a revisión
```

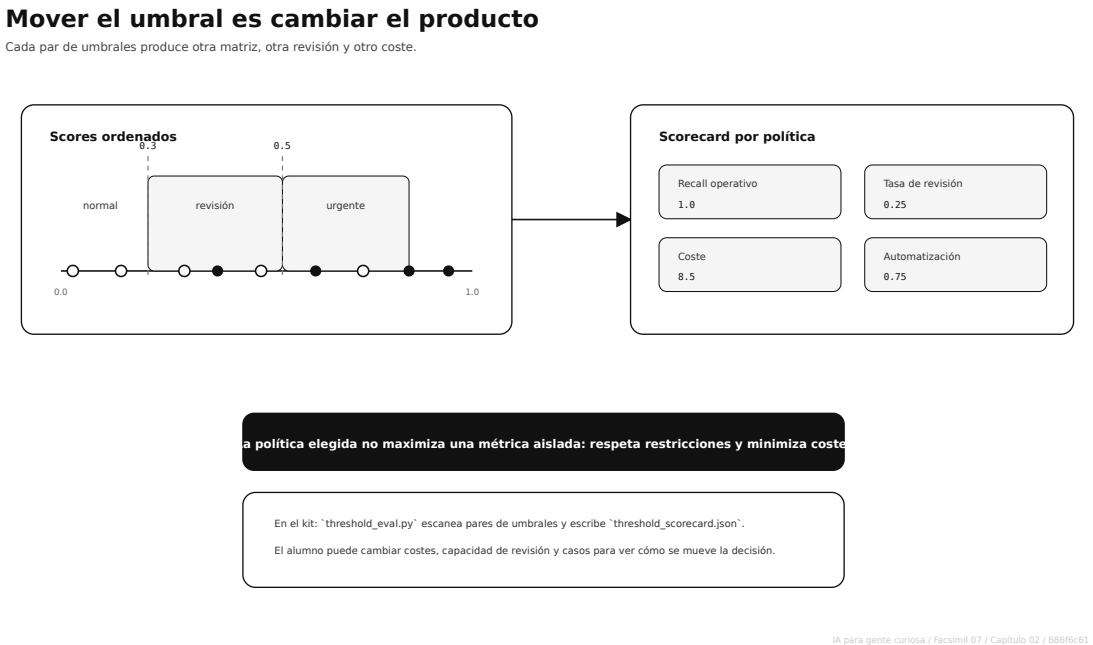
La zona gris no es una derrota. Es una herramienta de ingeniería cuando el coste de equivocarte supera el coste de revisar. El problema es que la revisión también tiene capacidad finita. Si todo cae en la zona gris, no has automatizado: has creado una cola.

La zona gris se diseña con dos restricciones a la vez. La primera es de calidad: quiero que los casos claramente normales salgan como normales y los claramente urgentes salgan como urgentes. La segunda es de capacidad: solo puedo revisar una parte del volumen. Si el equipo puede revisar un 35 % de los casos, una política que manda el 60 % a revisión no es “prudente”; es inoperable.

En el runner de la práctica, cada par de umbrales produce cuatro lecturas: coste, recall operativo, tasa de revisión y tasa de automatización. Esa combinación es más útil que discutir un único número. Si una política tiene gran recall pero revisa demasiado, no escala. Si automatiza mucho pero pierde positivos, no es segura. Si minimiza coste en la muestra pero falla en el slice de acceso, no debería publicarse sin más.

DECISIÓN DE UMBRAL	QUÉ SUELE PASAR	RIESGO QUE VIGILAS
Umbral alto muy exigente	Pocos positivos automáticos, precision alta.	Pierdes positivos reales.
Umbral bajo muy permisivo	Muchos positivos automáticos, recall alto.	Generas demasiadas falsas alarmas.
Dos umbrales con zona gris	Automatizas extremos y revisas ambiguos.	Saturas revisión si la zona gris es grande.
Umbral por slice	Ajustas decisión por segmento.	Puedes introducir reglas difíciles de gobernar.

Esta última fila merece cuidado. A veces tiene sentido usar umbrales distintos por canal, producto o idioma, pero eso aumenta complejidad y exige trazabilidad. Si una persona recibe una decisión distinta por pertenecer a un slice, necesitas justificarlo técnicamente y vigilar impacto. No es solo una mejora de métrica; es una política.



El umbral no se elige por costumbre: se barre, se mide y se defiende con una política de coste y revisión.

ROC, PR y qué mirar primero

ROC compara tasa de verdaderos positivos contra tasa de falsos positivos mientras movemos el umbral:

$$TPR = \frac{TP}{TP + FN}$$

En palabras: TPR es lo mismo que recall; mide qué proporción de positivos reales detectas.

$$FPR = \frac{FP}{FP + TN}$$

En palabras: FPR mide qué proporción de negativos reales conviertes en falsas alarmas.

SÍMBOLO	SIGNIFICADO	EJEMPLO
TPR	True Positive Rate; equivalente a recall.	0,82.
FPR	False Positive Rate.	$7/(7+71) = 0,09$.

Precision-recall mira precision contra recall. En clases muy desbalanceadas, PR suele enseñar mejor si los positivos predichos están llenos de ruido. Por eso Saito y Rehmsmeier recomiendan mirar precision-recall en datasets desbalanceados.[1][2]

La diferencia práctica es importante. Una curva ROC puede verse razonablemente buena aunque el positivo sea rarísimo, porque el número de negativos reales es enorme y la tasa de falsos positivos puede parecer pequeña. Pero una tasa pequeña aplicada a millones de negativos puede producir miles de falsas alarmas. La curva precision-recall te obliga a mirar la pureza de los positivos que realmente vas a tocar.

En un sistema de tickets, ROC responde bien a la pregunta “¿cómo se mueve la tasa de detección frente a la tasa de falsas alarmas?”. PR responde mejor a “cuando digo urgente, ¿cuántas veces estoy metiendo ruido en la cola?”. Si el equipo humano trabaja sobre los positivos predichos, PR suele estar más cerca del dolor diario.

Lectura práctica:

SITUACIÓN	MÉTRICA O CURVA QUE MIRARÍA PRIMERO
Clases equilibradas y costes parecidos.	Accuracy, matriz, F1 y ROC.
Positivo raro y caro de perder.	Recall, PR curve, F-beta con $\beta > 1$, coste de FN.
Positivo marcado genera trabajo humano.	Precision, coste de FP, tasa de revisión.
Decisión con score usado como probabilidad.	Calibración, Brier/log loss y umbrales; lo veremos en el capítulo 05.
Sistema con slices críticos.	Métricas por slice antes que media global.

En el día a día

Supón que un equipo quiere automatizar priorización de tickets. No necesita solo “un modelo que clasifique”. Necesita una política que pueda defender ante soporte, producto y operaciones.

En una demo, el modelo puede parecer útil porque marca como urgente los casos obvios: “no puedo entrar”, “servicio bloqueado”, “pago duplicado”. Pero la decisión real vive en los casos intermedios: mensajes ambiguos, clientes con distinto contexto, incidencias que suenan normales pero ocurren antes de un cierre, o tickets de acceso que parecen rutina hasta que bloquean una entrega.

Ahí las métricas se vuelven operativas:

PREGUNTA	DECISIÓN CONCRETA
¿Qué clase positiva importa?	`urgente`.
¿Qué coste tiene perder un urgente?	Alto: afecta tiempos de respuesta y continuidad.
¿Qué coste tiene marcar normal como urgente?	Medio: consume revisión y altera prioridad.
¿Cuántos casos puede revisar el equipo?	Por ejemplo, 35 % del volumen de la muestra.
¿Qué umbral acepta producto?	El que minimice coste respetando capacidad y recall mínimo.
¿Qué pasa si el volumen cambia?	Se monitoriza base rate, matriz por día y cola de revisión.

El capítulo 01 nos enseñó a crear el expediente de evaluación. Este capítulo añade el motor numérico para defender ese expediente: matriz, métricas, coste, umbral y decisión.

En una reunión real, una buena lectura no sería “F1 sale 0,80”. Sería algo más parecido a esto: “con umbral bajo 0,3 y alto 0,5 automatizamos el 75 % de la muestra, revisamos el 25 %, no perdemos urgentes en esta validación, generamos dos falsas urgencias en consultas y mantenemos el coste en 8,5 unidades didácticas. El slice de acceso queda cubierto por automatización o revisión; el slice de consulta concentra las falsas urgencias. Si el volumen de consultas crece, tendremos que revisar coste y umbral”.

Esa frase es más larga, pero es muchísimo más útil. Da decisión, evidencia, límites y siguiente vigilancia. Ese es el tipo de densidad que buscamos en una evaluación: no más jerga, sino más información accionable por frase.

Por qué debería importarte

Porque estas métricas son el punto donde un sistema de IA deja de ser una respuesta bonita y se convierte en una decisión que afecta trabajo real.

Si eliges mal la métrica, puedes publicar un sistema que parece mejorar y aun así empeora lo importante. Si eliges mal el umbral, puedes saturar una cola humana o dejar pasar casos críticos. Si no miras slices, puedes aprobar globalmente una release que falla en pagos, acceso, idioma, región o tipo de usuario. Y si no escribes el coste, cada reunión se convierte en una discusión de gustos.

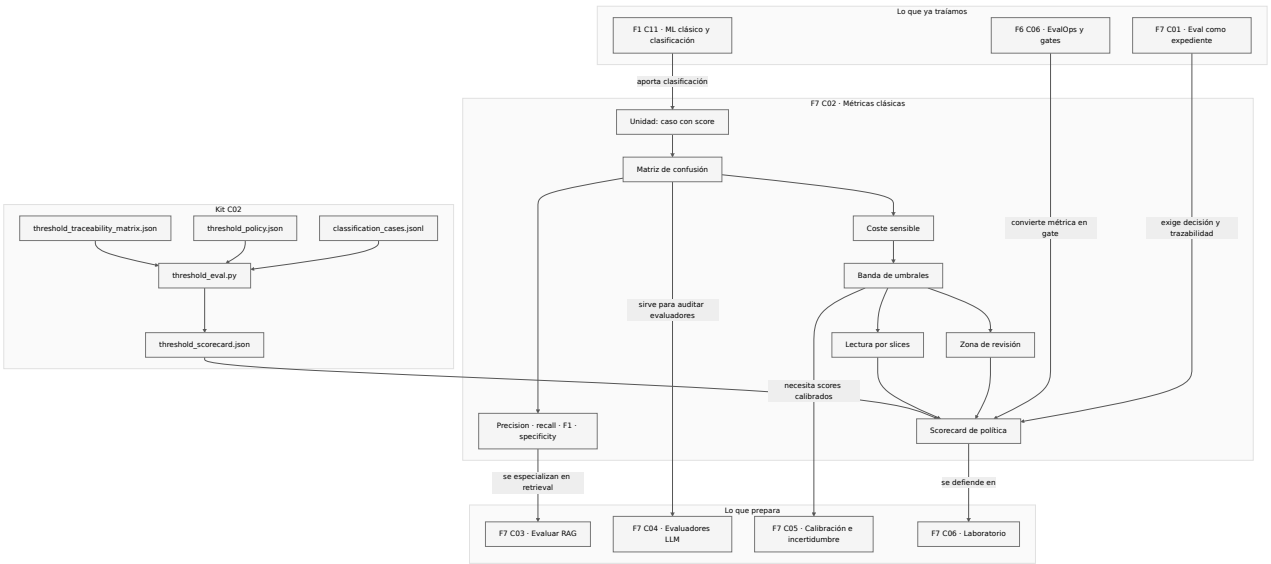
La matriz de confusión te da lenguaje común. Precision y recall te dan lectura. El coste sensible te obliga a priorizar. La zona gris te permite no fingir certeza cuando el sistema no la tiene. Esa combinación es ingeniería básica para evaluar clasificadores, routers, filtros, moderadores, detectores de riesgo, evaluadores de respuestas y gates de release.

También importa porque muchos sistemas modernos disfrazan decisiones binarias bajo interfaces más amables. Un router decide si una consulta va a RAG o a respuesta directa. Un guardrail decide si deja pasar una salida. Un evaluador decide si una respuesta cumple una rúbrica. Un detector decide si manda una conversación a revisión. Aunque por fuera hables de agentes, LLMs o pipelines multimodales, por dentro siguen apareciendo decisiones de clasificación. Si no sabes leer matriz, coste y umbral, acabarás aceptando decisiones automáticas sin entender qué errores compras.

Dónde solía tropezar yo

TROPIEZO	POR QUÉ OCURRE	ANTÍDOTO
Celebrar accuracy sin mirar la clase positiva.	Si el positivo es raro, accuracy puede ser alta aunque el sistema no encuentre lo importante.	Empezar por matriz de confusión, recall y coste del FN.
Creer que F1 elige por mí.	F1 no conoce capacidad de revisión, coste operativo ni daño de cada error.	Usar F1 como resumen, no como jefe.
Mover el umbral en el test final.	Si eliges umbral mirando el test final, contaminas la estimación.	Separar validación para elegir umbral y test para estimar rendimiento final.
No contar la revisión como salida.	Enviar a revisión consume tiempo y cambia el coste.	Medir `review_rate`, `automation_rate` y coste de revisión.
No mirar slices.	Una política puede funcionar globalmente y fallar en un segmento pequeño.	Reportar matriz y métricas por slice antes de publicar.
Tratar un score como probabilidad calibrada.	Muchos scores ordenan casos, pero no expresan probabilidad real.	Dejar la calibración para el capítulo 05 y no prometer más de lo medido.

Cómo encaja todo



Vocabulario aprendido

TÉRMINO	DEFINICIÓN BREVE
Matriz de confusión	Tabla que cruza realidad y predicción por tipos de acierto y error.
TP	Positivo real que el sistema marca como positivo.
FP	Negativo real que el sistema marca como positivo.
FN	Positivo real que el sistema marca como negativo.
TN	Negativo real que el sistema marca como negativo.
Accuracy	Aciertos totales entre casos totales.
Precision	Proporción de positivos predichos que eran positivos reales.
Recall	Proporción de positivos reales que el sistema detecta.
Specificity	Proporción de negativos reales que el sistema deja como negativos.
F1	Media armónica entre precision y recall.
F-beta	Variante de F1 que da más peso a recall o precision.
Balanced accuracy	Media entre recall y specificity.
Coste sensible	Evaluación que pondera errores distintos con consecuencias distintas.
Umbral	Corte que convierte score en acción.
Zona de revisión	Rango de score que se manda a revisión.
Recall operativo	Positivos cubiertos por automatización positiva o revisión.
Tasa de revisión	Proporción de casos enviados a revisión.
Tasa de automatización	Proporción de casos decididos sin revisión.
Slice	Subgrupo donde miramos métricas separadas.
Soporte	Número de casos reales por clase o por slice.
Prevalencia	Frecuencia base de la clase positiva en evaluación o producción.

TÉRMINO	DEFINICIÓN BREVE
Matriz de costes	Tabla que asigna consecuencias a errores y decisiones.

Antes de pasar página

Antes de avanzar al siguiente capítulo, deberías poder responder:

1. ¿Por qué accuracy puede ser engañosa en clases desbalanceadas?
2. ¿Qué diferencia hay entre FP y FN en un sistema de tickets urgentes?
3. ¿Qué pregunta responde precision?
4. ¿Qué pregunta responde recall?
5. ¿Por qué F1 no sustituye una matriz de costes?
6. ¿Qué cambia cuando usamos dos umbrales y zona de revisión?
7. ¿Por qué PR curve suele ser más útil que ROC cuando el positivo es raro?
8. ¿Qué significa elegir umbral por coste sensible?
9. ¿Qué archivos produce la práctica del capítulo?
10. ¿Qué entregarías para defender una política de automatización?

En resumen

IDEA	QUÉ TE LLEVAS
La matriz de confusión es la contabilidad básica de la clasificación.	Sin ella, no sabes qué tipo de error estás cometiendo ni qué cuadrante debería preocupar al equipo.
Soporte y prevalencia condicionan todo.	Un 90 % de accuracy no significa lo mismo con clases equilibradas que con un positivo raro.
Precision y recall responden preguntas distintas.	Precision mide ruido en positivos predichos; recall mide positivos reales encontrados. Ambas necesitan contexto.
F1 resume, pero no decide.	Si FP y FN cuestan distinto, necesitas coste sensible, restricciones y política operativa.
El umbral es una decisión de producto e ingeniería.	Moverlo cambia automatización, revisión, errores y coste; no debería elegirse tanteando el test final.
La zona gris es útil si cabe en operación.	Revisar casos ambiguos puede ser mejor que forzar automatización, pero una cola imposible no es una solución.
Los slices importan.	Una métrica global puede esconder el fallo que más te importa, especialmente en segmentos pequeños o críticos.
El cuaderno no es un adorno.	Practicas con datos, política, runner, tests y salida para tomar una decisión real y defenderla.

Para saber más

Davis, J. y Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning*, 233-240. https://doi.org/10.1145/1143844.1143874

Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 155-164. https://doi.org/10.1145/312129.312220

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1), 1-26. https://doi.org/10.1214/aos/1176344552

Elkan, C. (2001). The foundations of cost-sensitive learning. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 973-978.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861-874. https://doi.org/10.1016/j.patrec.2005.10.010

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1), 103-123. https://doi.org/10.1007/s10994-009-5119-5

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2), 153-157. https://doi.org/10.1007/BF02295996

Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1), 37-63.

Saito, T. y Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

scikit-learn. (2026). *Classification Metrics*. https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

scikit-learn. (2026). *classification_report*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

scikit-learn. (2026). *confusion_matrix*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Notas

1. scikit-learn. (2026). *Classification Metrics*. https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics. Consultado el 28 de mayo de 2026. ↩

2. scikit-learn. (2026). *confusion_matrix*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. Consultado el 28 de mayo de 2026. ↩

3. scikit-learn. (2026). *classification_report*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html. Consultado el 28 de mayo de 2026. ↩

4. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), 861-874. https://doi.org/10.1016/j.patrec.2005.10.010 ↩

5. Davis, J. y Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning*, 233-240. https://doi.org/10.1145/1143844.1143874 ↩

6. Saito, T. y Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. https://doi.org/10.1371/journal.pone.0118432 ↩

7. Elkan, C. (2001). The foundations of cost-sensitive learning. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 973-978. ↩

8. Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 155-164. https://doi.org/10.1145/312129.312220 ↩

9. Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1), 37-63. ↩

.0. Elkan, 2001. ↩

.1. Domingos, 1999. ↩

.2. Saito y Rehmsmeier, 2015. ↩

Cuadernos para practicar

Escanea el código con el móvil para abrir el cuaderno en Google Colab y ejecutarlo sin instalar nada.

El coste del error: dónde poner el umbral

https://colab.research.google.com/github/686f6c61/iaparagentecuriosa-notebooks/blob/main/Facs%C3%ADmil%2007/02-coste-error-umbral.ipynb