Facsímil 08

La ciencia de los datos

Capítulo 05

Slices, sesgos y decisión algorítmica

686f6c61 · 2026

Capítulo 05: Slices, sesgos y decisión algorítmica

Entrando en el tema

En el capítulo anterior convertimos datos en representaciones. Ahora viene una pregunta menos cómoda: **¿esa representación se comporta igual de bien en todas las partes importantes del problema?**

La respuesta casi nunca sale mirando solo la media global. Una accuracy de 0,86, un recall de 0,78 o un coste medio aceptable pueden esconder que el sistema falla en un producto, un canal, un idioma, una fuente de datos, una necesidad de accesibilidad o una combinación pequeña pero importante. A esa partición útil del comportamiento la llamamos slice.

Al terminar deberías poder hacer esto:

RESULTADO DE APRENDIZAJE	EVIDENCIA DE QUE LO SABES HACER
Definir slices útiles.	No partes por columnas al azar: eliges segmentos conectados con la decisión.
Separar atributo de auditoría y feature.	Entiendes que un campo puede servir para medir aunque no deba entrar al modelo.
Calcular métricas por slice.	Obtienes recall, tasa de revisión, falsos positivos, coste y captura segura por grupo.
Leer disparidades sin convertirlas en eslogan.	Sabes qué métrica se compara, qué tamaño de muestra la sostiene y qué acción permite.
Entender criterios clásicos de equidad algorítmica.	Distingues paridad demográfica, igualdad de oportunidad, odds igualadas y calibración por grupo.
Convertir una auditoría en decisión.	Generas un reporte, un CSV de slices, una ficha y una decisión operativa.

La clave es que un slice no es una curiosidad estadística. Es una forma de preguntar si el sistema trata bien las situaciones donde realmente va a usarse. En un producto educativo, puede importar idioma, canal, necesidad de accesibilidad, tipo de trámite, fuente documental o centro. En otro dominio serán otras variables. Lo importante es que cada slice tenga una razón de existir y una acción posible si falla.

La frase central del capítulo:

> Un sistema no se publica porque la media global suene bien; se publica cuando sabes dónde funciona, dónde no y qué harás con esa diferencia.

La escena: la media global llega demasiado tarde

Imagina un sistema que ayuda a priorizar casos académicos. Cada caso recibe un score. Si el score es alto, se prioriza. Si es bajo, sigue el flujo normal. Si cae en medio, se manda a revisión.

La métrica global parece razonable. El sistema acierta muchos casos, revisa una parte manejable y mantiene buena latencia. En una reunión rápida alguien podría decir: “vamos adelante”.

Pero al partir los resultados por slices aparece otra lectura. Los casos con necesidad de adaptación tienen más casos prioritarios enviados al flujo normal. En inglés se revisa mucho más. En un producto concreto, los casos importantes no quedan capturados. La media global no mentía; simplemente comprimía demasiado.

Este capítulo enseña a no dejar que eso pase.

Qué no es auditar por slices

Auditar por slices no es hacer una tabla enorme con todas las columnas del CSV. Si cada valor distinto se convierte en una fila de auditoría, obtienes ruido. Un slice debe existir porque representa una hipótesis: “este canal cambia la forma de escribir”, “este producto tiene más ambigüedad”, “este idioma tiene menos ejemplos”, “este grupo necesita que no confundamos silencio con baja prioridad”.

Tampoco es declarar que una diferencia numérica demuestra una injusticia completa. Una disparidad es una señal medible, no una sentencia universal. Puede venir de datos escasos, etiquetas inconsistentes, política de negocio, diseño del umbral, cobertura desigual, drift o un error real del modelo. La ingeniería consiste en separar esas causas antes de automatizar más.

Y no es lo mismo usar un atributo para decidir que usarlo para auditar. En muchos sistemas, ciertos campos no deben entrar como feature. Aun así, puede ser imprescindible medir resultados agregados por esos campos para detectar si el sistema está fallando justo donde más importa. El contrato debe decirlo: **campo no usado por el modelo, campo permitido para auditoría agregada.**

Qué sí es un slice

Un slice es un subconjunto definido por una condición. Usaremos notación estándar de teoría de conjuntos para nombrarlo. No es una métrica nueva ni un algoritmo: solo una forma compacta de decir “qué filas entran en este segmento de evaluación”. Si D es el conjunto evaluado y A es un atributo de auditoría, el slice asociado al valor a es:

$$D_a = \{(x_i, y_i, \hat{s}_i) \in D : A_i = a\}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
D	Dataset de evaluación.	36 casos de test del cuaderno.
A	Atributo usado para auditar.	`language`.
a	Valor concreto del atributo.	`en`.
D_a	Slice formado por casos con ese valor.	Casos de test en inglés.
x_i	Entrada o representación del caso.	Texto, producto, canal, metadata.
y_i	Etiqueta real.	`true_priority = 1`.
$\hat{s}_i$	Score producido por el sistema.	`0.84`.

La parte importante es que el slice conserva la unidad de decisión. No evaluamos “idioma” en abstracto. Evaluamos **decisiones sobre casos** dentro de un idioma, producto, canal o combinación de condiciones.

En el cuaderno del facsímil, una política de triaje convierte score en decisión. Los umbrales forman parte de la política del ejercicio. Lo importante es que estén congelados antes de mirar test y que cada salida tenga una consecuencia operativa.

REGLA DEL CUADERNO	DECISIÓN	CONSECUENCIA
`score >= 0.78`	Priorizar.	Caso claro de alta prioridad.
`score < 0.38`	Flujo normal.	Caso claro de baja prioridad.
Resto	Revisar.	Caso intermedio que no debería automatizarse sin mirar.

Esta política no se ajusta en test. Test sirve para medir. Si tocamos umbrales mirando el slice que falla, ya no estamos auditando: estamos usando la evaluación como desarrollo. Eso lo vimos en el capítulo 03 del facsímil 08.

Métricas por slice: lo mínimo que un ingeniero debería mirar

Para cada slice conviene calcular una matriz de confusión. En una decisión binaria clásica, tendríamos TP, FP, FN y TN; esa notación es la base habitual para leer clasificadores, curvas ROC, precisión, recall y tasas de error.[1][2] `scikit-learn`, por ejemplo, documenta la matriz de confusión como el conteo de ejemplos reales frente a ejemplos predichos por clase.[3] En una política con revisión, añadimos una tercera salida: `revisar`. Eso cambia la lectura.

Si un caso prioritario se manda a `normal`, tenemos un fallo operativo fuerte. Si se manda a `revisar`, no se ha automatizado bien, pero tampoco se ha dejado pasar sin mirar. Por eso el cuaderno distingue tres métricas:

MÉTRICA	CÁLCULO DEL CUADERNO	LECTURA
Auto-recall	`tp / positives`	De los casos prioritarios, cuántos se priorizan automáticamente. Es recall aplicado solo a la salida `priorizar`.
Miss rate	`fn / positives`	De los casos prioritarios, cuántos se envían a flujo normal. Es la cara operativa del falso negativo.
Captura segura	`(tp + urgent_review) / positives`	De los prioritarios, cuántos quedan priorizados o revisados. Es un cálculo operativo del cuaderno, no un criterio académico de fairness.

Donde:

SÍMBOLO	SIGNIFICADO	EJEMPLO
P	Número de casos positivos reales en el slice.	6 casos prioritarios.
TP	Positivos reales priorizados.	2 casos.
FN	Positivos reales enviados a flujo normal.	4 casos.
`urgent_review`	Positivos reales mandados a revisión.	1 caso.

En palabras: el recall académico pregunta cuántos positivos reales recupera la salida positiva. El cuaderno conserva esa lectura en `auto_recall`, pero añade `captura segura` porque la política no solo decide `priorizar` o `normal`; también puede abstenerse y mandar a revisión. Esa tercera salida no convierte el problema en "justo" por sí misma. Solo evita que ciertos positivos reales se pierdan sin mirar.

Para no olvidar el coste, el cuaderno calcula una lectura operativa separando falsos negativos, falsos positivos y revisiones. Es una tabla de pesos didácticos que el alumno puede cambiar para ver cómo cambia la decisión.

COMPONENTE	QUÉ PENALIZA	PESO DEL CUADERNO	POR QUÉ IMPORTA
`FN`	Enviar un prioritario a flujo normal.	8,0	Es el error más caro del caso.
`FP`	Priorizar un no prioritario.	3,0	Consume capacidad de atención.
`R`	Mandar a revisión humana.	1,2	Protege, pero no escala gratis.

Un coste no tiene que ser euros. Puede ser minutos de equipo, riesgo operativo, saturación de soporte o coste de oportunidad. Lo importante es hacerlo explícito antes de elegir política.

Este párrafo es más importante que la tabla. En IA aplicada, cada decisión desplaza trabajo a algún sitio: al usuario, al equipo de soporte, al equipo legal, al revisor humano, al sistema de colas o al presupuesto de inferencia. Si solo miras `accuracy`, esos desplazamientos desaparecen. Cuando asignas pesos de coste, aunque sean didácticos, estás declarando qué daño estás intentando evitar.

Criterios clásicos: nombres útiles, no dogmas

La literatura de equidad algorítmica distingue varios criterios. Conviene conocerlos porque aparecen en papers, herramientas y auditorías, pero no deben usarse como recetas automáticas.

La **paridad demográfica** compara tasas de selección entre grupos. Es una noción clásica en la literatura de fairness: mira si la salida positiva aparece con tasas parecidas al condicionar por el atributo de grupo, sin comprobar si las tasas base reales son iguales.[4] En nuestro caso sería comparar qué proporción de casos se prioriza en cada slice:

$$P(\hat{Y} = 1 \mid A = a)$$

Sirve para detectar diferencias de tasa de salida. No sabe si los casos positivos reales estaban distribuidos igual. Si un producto recibe más casos realmente prioritarios que otro, exigir la misma tasa de priorización puede ser una mala idea.

La **igualdad de oportunidad** compara la tasa de verdaderos positivos entre grupos, es decir, si los casos positivos reales reciben la salida positiva con tasas parecidas.[5] En el cuaderno se parece al auto-recall por slice:

$$P(\hat{Y} = 1 \mid Y = 1, A = a)$$

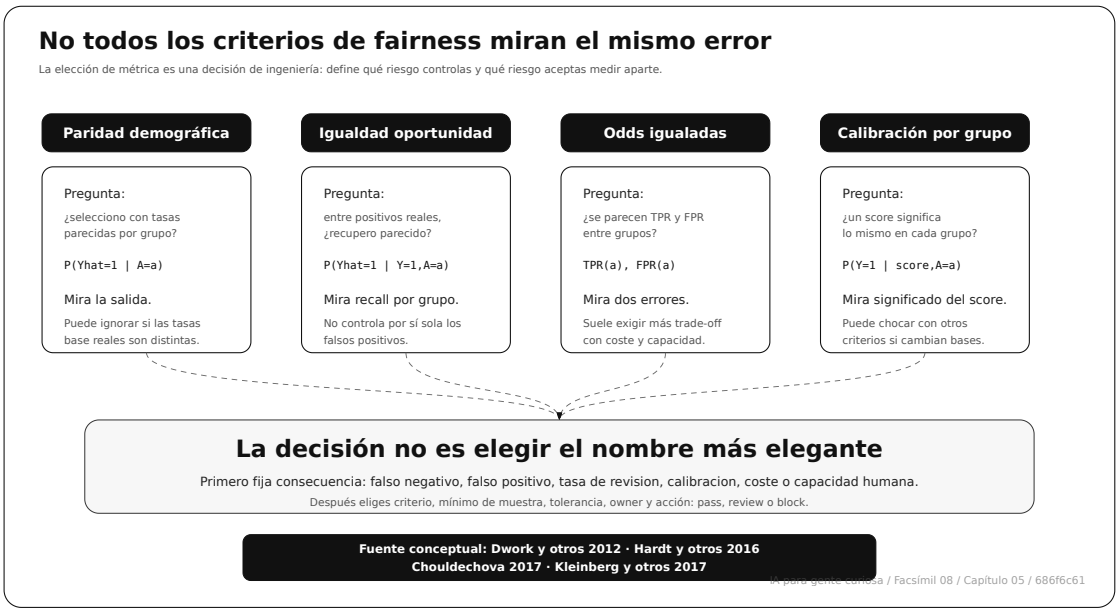
SÍMBOLO	SIGNIFICADO	EJEMPLO
$\hat{Y}$	Decisión predicha o salida positiva del sistema.	`priorizar`.
Y	Etiqueta real.	`true_priority = 1`.
A	Atributo de grupo o auditoría.	`access_need`.
a	Valor concreto del atributo.	`si`.

En palabras: paridad demográfica mira la tasa de salida positiva dentro de cada grupo; igualdad de oportunidad mira esa tasa solo entre los casos que de verdad eran positivos. La primera responde “¿selecciono igual de a menudo?”. La segunda responde “cuando debía seleccionar, ¿fallo igual de poco?”. Son preguntas distintas y pueden empujar decisiones distintas.

Las **odds igualadas** comparan tanto verdaderos positivos como falsos positivos entre grupos. Pide que el sistema tenga comportamiento parecido para positivos reales y negativos reales. Es más exigente que mirar solo recall.

La **calibración por grupo** pregunta si un score significa lo mismo en cada grupo. Si los casos con score 0,8 aciertan el 80 % en un slice y el 55 % en otro, no basta con decir que el score ordena bien globalmente. Esto conecta con calibración e incertidumbre en el facsímil 07.

Hay un punto clave: algunos criterios no se pueden satisfacer todos a la vez salvo en condiciones especiales. Chouldechova y Kleinberg, Mullainathan y Raghavan mostraron incompatibilidades entre nociones de equidad cuando las tasas base difieren entre grupos.[6][7] Traducido a ingeniería: elegir una métrica de equidad es elegir qué error quieres controlar y qué compromiso aceptas.



Los criterios de equidad algorítmica son preguntas distintas sobre la misma política: salida, positivos reales, falsos positivos, score y coste operativo.

Sesgo no siempre significa lo mismo

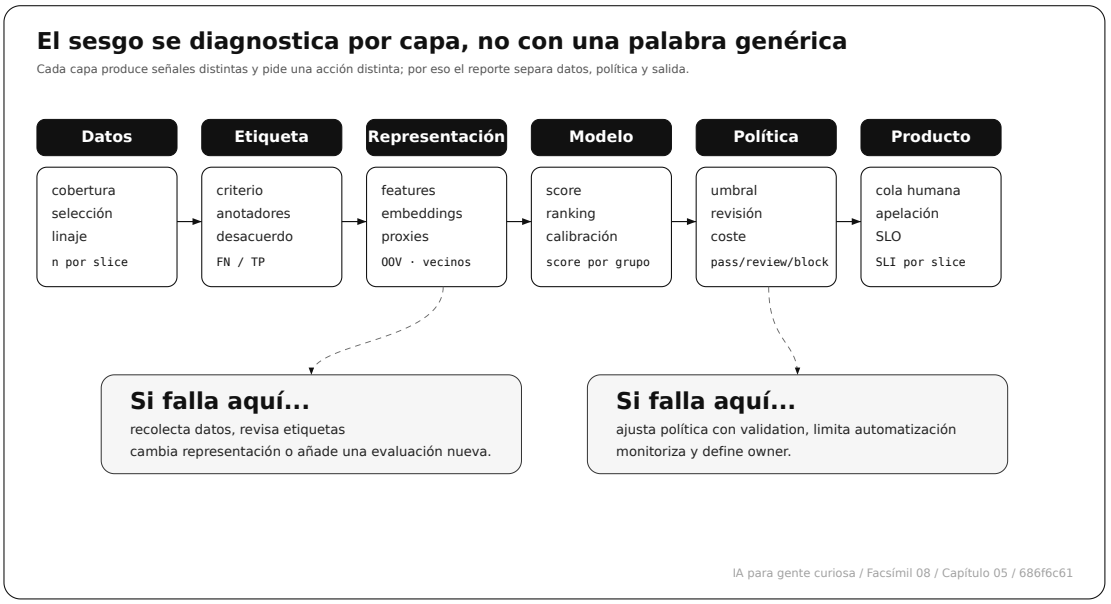
La palabra sesgo se usa demasiado rápido. Para ingeniería, decir “el modelo tiene sesgo” sin precisar de qué tipo es casi no ayuda. Necesitamos localizar **dónde entra**, **cómo se manifiesta**, **qué métrica lo hace visible** y **qué acción permite**.

En este capítulo no usamos sesgo como insulto técnico. Lo usamos como una diferencia sistemática que puede afectar a una decisión. Puede estar en los datos, en el objetivo, en la representación, en el modelo, en el umbral, en la interfaz o en la forma de medir. Si no separas esas capas, acabas arreglando el sitio equivocado.

FUENTE DE SESGO	QUÉ OCURRE	CÓMO SE DETECTA	QUÉ SUELE HACERSE
Cobertura	Un slice tiene pocos ejemplos o no aparece en train.	Conteos, intervalos, unknowns, categorías raras.	Recoger más datos, limitar automatización, declarar cobertura.
Selección	Los datos observados no representan el uso real.	Comparar distribución de train, test y producción.	Rehacer split, muestreo, ponderación o contrato de uso.
Medición	Un campo no mide lo mismo en todos los grupos.	Revisar definición, instrumento de captura y linaje.	Cambiar proxy, añadir variable mejor, documentar límite.
Proxy	Una variable cómoda sustituye mal al concepto real.	Correlación con slices, errores concentrados, revisión de dominio.	Sustituir proxy o cambiar objetivo.
Etiquetado	Las etiquetas se aplican con criterios distintos.	Acuerdo entre anotadores, desacuerdo por slice, revisión de casos frontera.	Reescribir política de anotación y reetiquetar muestra.
Representación	Las features o embeddings capturan peor un segmento.	OOV, vecinos pobres, recall por slice, sensibilidad a idioma o formato.	Cambiar encoder, vocabulario, normalización o datos de entrenamiento.
Agregación	Un único modelo o umbral mezcla subpoblaciones con comportamientos distintos.	Buen promedio global con slices débiles.	Umbrales por contexto, rutas de revisión o modelos especializados con control.
Evaluación	El test no contiene el problema que aparecerá en producción.	Falta de slices críticos, test pequeño, ausencia de intersecciones.	Nueva eval, holdout, monitorización por slice.
Interacción	La interfaz cambia lo que el usuario escribe o aporta.	Diferencias por canal, longitud, idioma, plantilla o formulario.	Rediseñar entrada, instrucciones y validaciones.
Política	El mismo score produce consecuencias distintas según contexto.	Coste, revisión, capacidad operativa y efectos por segmento.	Cambiar umbrales, limitar automatización o exigir revisión.

Un caso clásico de la literatura sanitaria mostró que usar coste sanitario como proxy de necesidad médica podía reducir la identificación de pacientes con necesidades reales en ciertos grupos, porque el gasto histórico no medía necesidad de forma neutral.[8] La lección para este facsímil no es copiar ese dominio; es entender el patrón: **un proxy cómodo puede cambiar la decisión que crees estar midiendo**.

Por eso el capítulo 01 insistía en linaje y uso permitido, el 02 en calidad, el 03 en splits y el 04 en representación. Los slices son donde esas decisiones anteriores se vuelven visibles.



> Un slice ayuda a ubicar la causa probable: no es igual un problema de cobertura que un umbral mal elegido o una cola humana sin capacidad.

Qué sabemos por estudios en modelos reales

Los sesgos no aparecen solo en clasificadores tabulares. También se han observado en embeddings, modelos de lenguaje, clasificadores de texto, sistemas de visión y sistemas de decisión basados en proxies. Ver esos estudios ayuda porque cambia la pregunta: ya no es “¿puede pasar?”, sino “¿cómo lo detectaría en mi caso?”.

ESTUDIO	TIPO DE MODELO O SISTEMA	QUÉ MOSTRÓ	SEÑAL DETECTABLE
Bolukbasi y otros (2016)	Word embeddings	Algunas analogías en embeddings capturaban asociaciones de género no deseadas.[9]	Direcciones en el espacio vectorial y vecinos semánticos.
Caliskan y otros (2017)	Embeddings entrenados con corpus	WEAT mostró asociaciones recuperables desde lenguaje ordinario.[10]	Test de asociación entre conjuntos de términos.
Buolamwini y Gebru (2018)	Clasificación facial comercial	Las tasas de error variaban mucho por intersección de tono de piel y género en sistemas evaluados.[11]	Accuracy por grupos e intersecciones, no solo global.
Dixon y otros (2018)	Clasificación de toxicidad	Ciertos términos de identidad podían disparar predicciones tóxicas aunque el texto no lo fuera.[12]	Falsos positivos asociados a términos concretos.
CrowS-Pairs (2020)	Modelos de lenguaje enmascarados	Comparó pares de frases para medir preferencias por frases con estereotipo.[13]	Probabilidad relativa entre pares mínimos.
StereoSet (2021)	Modelos de lenguaje preentrenados	Midió sesgo estereotípico junto con capacidad de modelado lingüístico.[14]	Relación entre score lingüístico y score de estereotipo.
BBQ (2022)	Pregunta-respuesta	Evaluó si modelos recurren a estereotipos cuando el contexto es insuficiente o ambiguo.[15]	Respuestas bajo contexto ambiguo frente a contexto informativo.

Hay dos lecciones de ingeniería en esa tabla.

La primera: el sesgo no siempre aparece como “métrica baja”. A veces aparece como asociación geométrica, falso positivo lexical, peor cobertura en intersecciones, sensibilidad al contexto ambiguo o score calibrado de forma distinta por grupo.

La segunda: cada tipo de sistema necesita su prueba. Para embeddings haces vecinos, direcciones, WEAT o pares mínimos. Para clasificación haces matriz por slice. Para RAG miras recuperación por fuente, idioma, fecha y permisos. Para generación miras pares contrafactuales, respuestas con contexto insuficiente, abstención y criterios de evaluación.

Detectabilidad: cómo se ve un sesgo antes de producción

Una auditoría útil no pregunta “¿hay sesgo?” en abstracto. Pregunta “¿qué señal observable esperaría si este sistema falla de forma sistemática?”.

SEÑAL DETECTABLE	CÓMO SE CALCULA	QUÉ INDICA	QUÉ NO DEMUESTRA SOLA
Slice con poco n	Conteo por segmento.	Falta evidencia para concluir.	Que el sistema sea malo en ese slice.
Diferencia de recall	`tp / positives` por slice.	Un grupo de positivos reales se detecta peor.	La causa del problema.
Diferencia de falsos positivos	`fp / negatives` por slice.	Un grupo recibe más salidas positivas indebidas.	Que el objetivo esté bien definido.
Captura segura baja	`(tp + urgent_review) / positives`.	Casos importantes pasan sin priorizar ni revisar.	Qué componente lo provocó.
Tasa de revisión dispar	`review / n` por slice.	Un segmento se automatiza menos o consume más revisión.	Que revisar sea necesariamente malo.
Coste por caso dispar	`cost_total / n` por slice.	El impacto operativo se concentra.	Que el coste esté bien ponderado.
OOV alto	Términos fuera del vocabulario por slice.	La representación cubre peor un segmento.	Que un embedding real no lo arregle.
Vecinos pobres	Top-k sin evidencia útil para un slice.	Recuperación débil o índice mal cubierto.	Que el generador sea el único culpable.
Pares contrafactuales inestables	Cambiar solo un atributo y comparar salida.	Sensibilidad indeseada a un cambio controlado.	Que todos los casos reales fallen igual.
Contexto ambiguo decide demasiado	Probar preguntas con evidencia insuficiente.	El modelo rellena huecos con asociaciones aprendidas.	Que falle cuando la evidencia es completa.

La palabra “detectable” importa. Si un sesgo no tiene métrica, muestra, caso de prueba o traza, se convierte en debate interminable. El objetivo no es tener todas las respuestas, sino diseñar señales que permitan revisar.

Por eso conviene escribir la prueba antes de mirar resultados. Si el equipo decide los slices después de ver la tabla, puede acabar contando la historia que más le conviene. Si los decide antes, con una hipótesis clara, la auditoría se parece más a ingeniería y menos a búsqueda de

relato. La pregunta no es “qué diferencia puedo encontrar”, sino “qué diferencia sería peligrosa si existiera”.

Buenas prácticas para no fabricar un problema nuevo

La buena práctica empieza antes de entrenar. Un equipo serio no espera al final para “pasar fairness”. Diseña la evaluación desde el contrato de datos.

BUENA PRÁCTICA	QUÉ HACES EN CONCRETO
Definir la unidad de decisión	No mezclas usuario, caso, documento, chunk y sesión como si fueran lo mismo.
Separar feature y auditoría	Un campo puede no entrar al modelo y aun así medirse de forma agregada.
Escribir slices críticos antes de test	Evitas elegir segmentos solo porque cuentan una historia cómoda.
Medir intersecciones	`language=en` puede parecer aceptable y `language=en + access_need=si` no.
Añadir mínimos de muestra	Sin n, positivos y negativos suficientes, el slice va a revisión.
Guardar política y hashes	Si cambia el dataset o el contrato, cambia la auditoría.
Congelar umbrales antes de test	Ajustar con test convierte auditoría en desarrollo.
Mirar coste, no solo métrica	Un fallo raro puede importar más que diez aciertos baratos.
Dejar una acción escrita	`pass`, `review` o `block` con razones y siguiente paso.
Conectar con monitorización	Los mismos slices se observan después en producción.

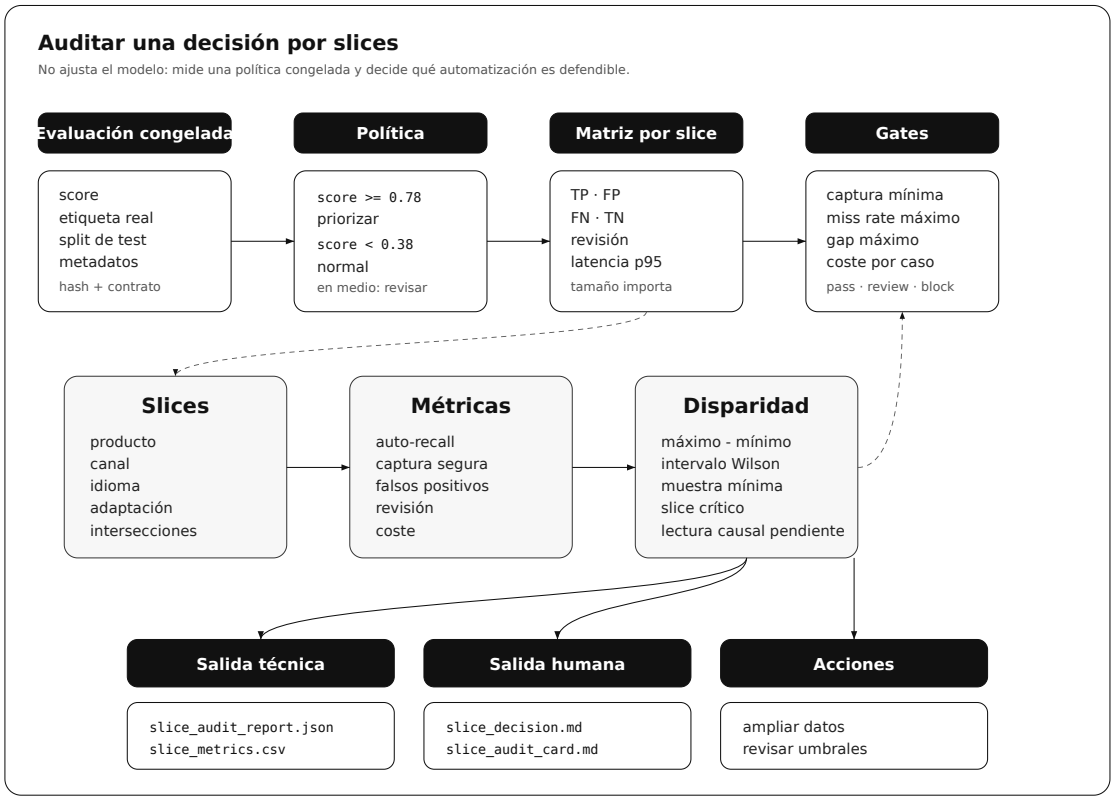
Y hay cosas que conviene evitar:

EVITA	POR QUÉ
“No medimos ese atributo, así que no hay problema”	No usar un campo como feature no implica que el sistema no tenga diferencias por ese campo.
“La muestra es pequeña, pero la media global pasa”	Un slice pequeño no se arregla escondiéndolo dentro del promedio.
“El benchmark público dice que el modelo es bueno”	El benchmark quizá no cubre tu dominio, idioma, interfaz o coste.
“Mitigamos borrando una columna”	Otros campos pueden actuar como proxies.
“Probamos muchos slices y publicamos solo los interesantes”	Eso convierte auditoría en selección de relato.
“Una herramienta lo certifica”	La herramienta calcula; el equipo define uso, consecuencia, coste, límites y acción.
“Arreglamos el test tocando el umbral”	Si el umbral se elige mirando test, necesitas nueva evaluación.
“Todos los slices deben tener la misma tasa”	Algunas tasas base reales pueden diferir; la pregunta es qué error estás controlando.

La buena práctica completa siempre termina en una decisión concreta. Si un slice queda débil por falta de muestra, quizá no publicas, pero tampoco concluyes que el sistema sea injusto: pides más datos o dejas el segmento en revisión. Si un slice falla con muestra suficiente, toca elegir: cambiar representación, revisar etiquetas, ajustar política en validation, añadir revisión humana o limitar automatización. Sin esa acción escrita, la auditoría se queda en una tabla incómoda que nadie sabe convertir en trabajo.

El cuaderno del facsímil incorpora estas prácticas en pequeño: `fields_not_for_model`, `audit_fields`, `critical_slices`, mínimos de muestra, gates, hashes y decisión Markdown. No es una auditoría completa de un sistema real, pero sí tiene la forma profesional que debe tener una primera revisión.

Anatomía de una auditoría de decisión



Auditar por slices convierte una métrica global en una decisión revisable: datos, política, segmentos, métricas, gates y acciones.

Herramientas reales y cómo leerlas

Fecha de corte de esta lectura: **7 de junio de 2026**. El mercado cambia rápido, así que no conviene memorizar logos. Conviene memorizar **qué artefacto produce cada herramienta**: métrica por slice, intervalo, explicación, alerta, comparación de versiones, política de mitigación o evidencia para una revisión.

Una herramienta seria no “quita el sesgo” en abstracto. Hace una de estas cosas: mide una diferencia, ayuda a encontrar causa probable, aplica una técnica de mitigación, deja trazabilidad o vigila si producción se aleja de lo que mediste en validación. Si no produce un artefacto revisable, es una demo bonita, no una práctica de ingeniería.

Herramientas abiertas para auditar y mitigar

Estas herramientas encajan bien cuando el equipo trabaja con notebooks, `pandas`, `scikit-learn`, TensorFlow o reportes reproducibles en CI. Son especialmente útiles para enseñar y para construir un primer estándar interno, porque puedes inspeccionar datos, métricas y código.

HERRAMIENTA	DÓNDE ENCAJA	QUÉ PERMITE VER	QUÉ PUEDE AYUDAR A MITIGAR	QUÉ NO DEBES DELEGARLE
Fairlearn	Modelos tabulares o pipelines compatibles con `scikit-learn`.	Métricas agrupadas con `MetricFrame`, comparación entre grupos y visualización de disparidades.[16]	Reducciones como `ExponentiatedGradient`, búsqueda con restricciones y optimización de umbrales.[17]	La definición de grupos, coste del error y métrica que gobierna la decisión.
AI Fairness 360	Comparar algoritmos de fairness, datasets de referencia y métricas clásicas.	Métricas, detectores, explicadores y API compatible con `scikit-learn` para parte del toolkit.[18]	Preprocesamiento, técnicas durante entrenamiento y postprocesamiento.[19]	El encaje con tu contrato de datos real. Muchos ejemplos académicos no se parecen a tu producto.
Aequitas	Auditoría de scores binarios o continuos con CSV, CLI o interfaz local.	Métricas de grupo, disparidades y criterios sobre scores, etiquetas y atributos.[20]	No mitiga por sí sola: ayuda a decidir si el modelo o la política deben cambiar.	Convertir una tabla de disparidades en decisión sin revisar el dominio.
Fairness Indicators	Ecosistema TensorFlow, TFMA y evaluación por slices.	Visualización de rendimiento por segmentos, intervalos y métricas en pipelines TensorFlow.[21]	No es una varita de mitigación; empuja a localizar slices débiles y a rediseñar datos o entrenamiento.	Pensar que solo aplica a “temas sociales”. También sirve para idioma, canal, dispositivo o fuente.
What-If Tool	Exploración interactiva de ejemplos, umbrales y contrafactuales.	Cómo cambian predicciones al mover ejemplos, atributos o umbrales.[22]	Ayuda a descubrir hipótesis antes de automatizar una prueba.	Sustituir una evaluación versionada. Explorar no es certificar.
Responsibly	Aprendizaje, prototipos y análisis de fairness en clasificación y NLP.	Métricas, visualizaciones y utilidades para enseñar conceptos con código.[23]	Puede acompañar un primer laboratorio o auditoría exploratoria.	Producción sin controles propios, versionado y trazabilidad.

Para un curso de ingeniería, Fairlearn y AIF360 son buenas primeras estaciones porque obligan a escribir `y_true`, `y_pred`, score, atributo sensible o de auditoría y métrica. Ese gesto parece básico, pero es la mitad del aprendizaje: si no sabes pasar esos arrays con nombres correctos, todavía no sabes qué estás auditando.

Plataformas cloud y de ciclo de vida

Cuando el sistema ya vive en una nube concreta, las herramientas integradas aportan algo que una libreta local no tiene: conexión con jobs, registros de modelo, monitorización, permisos, reportes y gobierno de releases.

PLATAFORMA	QUÉ CUBRE	CUÁNDO TIENE SENTIDO	CUIDADO DE INGENIERÍA
Amazon SageMaker Clarify	Sesgo en datos y modelos, explicabilidad, evaluaciones de modelos fundacionales y monitorización integrada con SageMaker.[24]	Si tu entrenamiento, registro o despliegue ya está en SageMaker.	Hay que fijar qué features o atributos se analizan, en qué split, con qué baseline y qué acción dispara una alerta.
Azure Responsible AI dashboard	Paneles de análisis responsable dentro de Azure ML: errores, importancia de variables, contrafactuales, causalidad y fairness según el tipo de modelo.[25]	Si trabajas con Azure ML y quieres que evaluación, registro y explicación vivan en el mismo flujo.	Un panel no sustituye un gate en CI/CD. Debe acabar en una decisión reproducible.
SageMaker Model Monitor, Azure ML monitoring o equivalentes	Vigilancia de datos, predicciones, latencia y cambios por segmento.	Si el riesgo no termina al desplegar.	La métrica por slice debe existir antes de producción; no la inventes cuando salta la alerta.

La pregunta correcta para una nube no es “¿tiene dashboard?”. Es: **¿puedo versionar la política, reproducir el reporte, bloquear un release, abrir una incidencia y comparar producción contra validación por los mismos slices?** Si no, el panel sirve para mirar, pero no para gobernar.

Observabilidad y herramientas de mercado

En producción aparece otra familia: plataformas de observabilidad de ML y GenAI. No suelen ser la primera herramienta para aprender fairness, pero sí son relevantes cuando tienes tráfico real, múltiples modelos, trazas, embeddings, RAG, usuarios, feedback y alertas.

HERRAMIENTA	QUÉ SUELE APORTAR	DÓNDE AYUDA CON SESGOS	QUÉ PEDIR ANTES DE ADOPTARLA
Evidently	Librería y plataforma para calidad de datos, drift, evaluación y reportes.[26]	Comparar referencia frente a producción, crear reportes por segmento y vigilar si cambian distribuciones.	Exportar reportes, integrarlo en CI y definir tus propios slices, no solo métricas por defecto.
Giskard	Evaluación y testing de modelos, agentes y aplicaciones GenAI.[27]	Pruebas de comportamiento en LLMs: toxicidad, estereotipos, robustez de prompts, RAG y regresiones de comportamiento.	Datasets de prueba propios y criterios explícitos. Sin casos de tu dominio, solo pruebas una maqueta.
Fiddler	Observabilidad, explicabilidad y métricas de fairness en producción.[28]	Monitorizar rendimiento por cohortes, explicar predicciones y detectar degradación por grupos.	Acceso a etiquetas o feedback posterior; sin ground truth retrasado, algunas señales serán aproximadas.
Arize	Observabilidad para ML, visión, embeddings y GenAI, con slicing, drift y análisis de rendimiento.[29]	Slices, cohortes, embeddings, drift y degradación de rendimiento en producción.	Que los atributos de auditoría lleguen al sistema con permisos y granularidad correcta.
WhyLabs	Observabilidad con perfiles de datos, drift, calidad, modelos predictivos y GenAI.[30]	Vigilar cambios de datos, problemas de calidad, drift, outputs de LLM y trazas.	Que las alertas estén conectadas con owners, SLOs y acciones, no solo con correos.
Arthur, WhyLabs, Fiddler, Arize u otros proveedores enterprise	Gobierno, monitorización, dashboards, explicabilidad y control operativo.[31]	Cuando hay varios equipos, varios modelos y requisitos de trazabilidad.	Evita comprar “confianza”. Compra integración, auditoría, permisos, exportación y comparabilidad.

El mercado es útil cuando resuelve un problema de operación: logging, trazas, control de versiones, dashboards compartidos, alertas, permisos, exportaciones y soporte. Pero el mercado no conoce por defecto tu coste de error. Tampoco sabe si un falso positivo molesta, si un falso negativo deja a alguien sin atención o si un slice con poca muestra debe bloquear o pedir más datos.

Mitigar no es borrar una columna

Hay una trampa clásica: detectar una diferencia, borrar el atributo sensible y declarar el problema resuelto. En sistemas reales, los proxies aparecen por código postal, idioma, canal, horario, dispositivo, historial, coste, texto libre o fuente documental. Por eso la mitigación

debe elegir **dónde está la causa probable**.

CAPA DE MITIGACIÓN	QUÉ CAMBIA	EJEMPLOS TÉCNICOS	CUÁNDO USARLA	QUÉ VIGILAR
Datos	La distribución, cobertura o calidad de las muestras.	Recolectar más casos, reetiquetar, balancear, reponderar, mejorar instrucciones de anotación, separar fuentes.	Hay slices con poca muestra, etiquetas inconsistentes o proxies claros.	No fabricar un dataset artificial que ya no se parezca a producción.
Representación	Cómo se codifica el caso.	Mejorar features, revisar embeddings, normalizar idioma, añadir metadatos permitidos, cambiar encoder, reducir OOV.	El modelo no ve señales suficientes en ciertos grupos o formatos.	No meter campos que expliquen demasiado bien una condición sensible sin justificación.
Entrenamiento	La función objetivo o restricciones del modelo.	Fairlearn reductions, restricciones de paridad, regularización, pérdidas ponderadas, búsqueda de hiperparámetros con gates.	El modelo aprende una frontera útil pero desigual.	Medir trade-off: una restricción puede mejorar un slice y empeorar otro.
Umbral y política	La conversión de score a acción.	Umbral global, banda de revisión, umbrales condicionados con revisión de dominio, abstención, doble lectura humana.	El score está razonablemente calibrado, pero la acción automática concentra riesgo.	Cambiar umbrales por grupo tiene implicaciones de producto, ética, regulación y soporte. No se improvisa.
Producto	La experiencia y el circuito humano.	Explicación al usuario, apelación, revisión manual, colas por capacidad, feedback posterior, monitorización.	La consecuencia de equivocarse es alta o no hay datos suficientes para automatizar.	Que la revisión humana no sea un cajón sin dueño, SLA ni trazabilidad.

La mitigación buena suele ser aburrida: cambiar datos, contratos, umbrales, monitorización, revisión y documentación. La mala mitigación suele parecer elegante: un algoritmo nuevo aplicado sin explicar qué consecuencia reduce.

Cómo elegir herramienta según tu stack

SI ESTÁS AQUÍ	EMPIEZA POR	AÑADE DESPUÉS
Notebook con CSV y modelo `sklearn`	Fairlearn o Aequitas para métricas por grupo.	AIF360 si necesitas comparar técnicas de mitigación.
Pipeline TensorFlow/TFX	Fairness Indicators y What-If Tool.	Reporte versionado y gate por slice en CI.
SageMaker	SageMaker Clarify.	Model Monitor, Model Cards y job recurrente por slice.
Azure ML	Responsible AI dashboard.	Componente de evaluación que exporte métricas a tu release.
Producto con tráfico real	Evidently, Fiddler, Arize, WhyLabs o plataforma equivalente.	SLO por slice, owner, alerta, runbook y comparación contra validación.
LLM, agente o RAG	Giskard, Evidently, Arize Phoenix u otras evals propias.	Dataset de pares mínimos, prompts de regresión, trazas y revisión humana en casos críticos.
Auditoría académica o de clase	Cuaderno propio + Fairlearn/AIF360/Aequitas.	Informe reproducible, model card, data card y defensa oral de decisiones.

Para decidir, yo haría esta prueba: toma diez filas reales, un contrato de política y una métrica por slice. Si la herramienta no puede decirte **qué fila falló, en qué slice, con qué métrica, contra qué baseline, desde qué versión y qué acción toca**, aún no tienes una herramienta de auditoría; tienes una visualización.

Qué evitar al comprar o adoptar una herramienta

1. Elegir por capturas de pantalla. Hay que pedir exportaciones, API, reproducibilidad y ejemplos con tus datos.
2. Medir fairness solo en validación y olvidarlo en producción. La distribución cambia y los slices también.
3. Usar atributos de auditoría sin política de acceso. Medir requiere permiso, minimización y propósito.
4. Mitigar antes de diagnosticar. No es igual falta de cobertura, proxy, mala etiqueta, mala calibración o umbral mal puesto.
5. Comparar herramientas sin el mismo dataset, split, métrica y política de decisión.
6. Aceptar métricas por defecto sin traducirlas a coste. Paridad, recall, falsos positivos y calibración no responden la misma pregunta.

7. Confundir un panel verde con una decisión publicable. La salida debe ser `pass`, `review` o `block`, con evidencia.

La regla práctica es sencilla: usa herramientas, pero no les delegues el criterio. Una herramienta calcula, visualiza, alerta o ejecuta una técnica. El equipo decide qué métrica representa la consecuencia que quiere controlar, qué slices importan y qué se hace cuando algo no pasa.

Caso completo: de `block` a `review`

Un buen capítulo de sesgos no puede quedarse en "usa una herramienta". Tiene que enseñar una historia completa: una política parece razonable, falla en slices críticos, se propone una mitigación y se vuelve a medir. Eso es lo que hace el cuaderno.

La política base dice:

DECISIÓN	REGLA
`priorizar`	`score >= 0.78`
`normal`	`score < 0.38`
`revisar`	`0.38 <= score < 0.78`

Con esa política, el resultado global queda en `block`: captura segura `0.7778`, pérdida operativa `0.2222`, tasa de revisión `0.3611` y coste por caso `1.4056`. El problema no es solo la media. En los slices críticos aparece algo peor:

SLICE CRÍTICO	N	POSITIVOS	AUTO-RECALL	PÉRDIDA	CAPTURA SEGURA	REVISIÓN	COSTE POR CASO
`language=en`	9	4	0.25	0.25	0.75	0.5556	1.5556
`access_need=si`	12	6	0.0	0.6667	0.3333	0.5	3.2667
`product=practicas	access_need=si`	4	2	0.0	1.0	0.0	0.5

La lectura humana es directa: algunos casos prioritarios con necesidad de accesibilidad o en inglés están yendo a flujo normal. No basta con decir "la accuracy global es aceptable". La política está dejando pasar casos que el contrato considera críticos.

La mitigación candidata no cambia el modelo. Cambia la política de decisión: amplía la banda de revisión bajando el umbral de flujo normal de `0.38` a `0.32`.

DECISIÓN	POLÍTICA BASE	POLÍTICA CANDIDATA
`priorizar`	`score >= 0.78`	`score >= 0.78`
`normal`	`score < 0.38`	`score < 0.32`
`revisar`	`0.38 <= score < 0.78`	`0.32 <= score < 0.78`

El antes/después queda así:

POLÍTICA	ESTADO	CAPTURA SEGURA	PÉRDIDA OPERATIVA	TASA DE REVISIÓN	COSTE POR CASO	FLAGS BLOCK	FLAGS REVIEW
Base	`block`	0.7778	0.2222	0.3611	1.4056	5	7
Banda de revisión	`review`	1.0	0.0	0.5278	0.7167	0	5

Esto no significa “ya está arreglado”. Significa algo más interesante para ingeniería: la candidata elimina los casos prioritarios enviados a flujo normal, pero aumenta la carga de revisión humana. Pasa de `block` a `review`, no a `pass`. Si el equipo no puede revisar aproximadamente el 53 % de los casos, la mitigación mejora una métrica y rompe la operación. Por eso fairness, coste y capacidad deben leerse juntos.

La matemática mínima de la mitigación

En un proyecto serio no se elige mitigación porque “suena mejor”. La literatura de clasificación justa con restricciones, por ejemplo el enfoque de reducciones de Agarwal y otros, formula el problema como minimizar error bajo restricciones de equidad medibles.[32] Una forma académica de leerlo es:

$$\min_{h \in \mathcal{H}} \quad \operatorname{error}(h) \text{ s.t.} \qquad \gamma_j(h) \leq \epsilon_j, \quad j = 1, \ldots, J$$

Donde:

SÍMBOLO	QUÉ SIGNIFICA	EJEMPLO
h	Clasificador o política candidata.	Política base o banda de revisión.
$\mathcal{H}$	Familia de clasificadores considerados.	Políticas que cambian umbrales o modelos candidatos.
$\text{error}(h)$	Error que se quiere minimizar.	Falsos negativos, falsos positivos o coste definido.
s.t.	Abreviatura estándar de *subject to*.	Introduce las restricciones que la solución no puede romper.
$\gamma_j(h)$	Violación de la restricción j, por ejemplo una disparidad de oportunidad.	Gap de recall por encima de lo permitido.
ϵ_j	Tolerancia permitida para esa restricción.	Gap máximo aceptado por el gate.

En palabras: la optimización no busca solo “el modelo que menos se equivoca en promedio”. Busca una política que reduzca error sin violar restricciones medibles. Si una política candidata tiene $\gamma_j(h) = 0{,}18$ para una restricción cuyo límite es $\epsilon_j = 0{,}10$, la violación no es una opinión: supera el margen permitido en `0.08`. En un sistema real eso debería activar `review` o `block`, según el contrato.

La parte importante es la restricción. No queremos solo minimizar error global. Queremos minimizarlo sin romper restricciones de slices, coste y operación. Fairlearn lo expresa con reducciones y restricciones; AIF360 ofrece varias familias de mitigación; nuestro cuaderno lo enseña con algo más humilde pero más transparente: umbrales, gates, slices y coste.

Hay tres familias de respuesta:

FAMILIA	QUÉ OPTIMIZA	EJEMPLO EN EL CUADERNO	CUÁNDO SIRVE
Cambiar datos	Mejorar cobertura o etiquetas.	Recolectar más casos `access_need=si`.	Cuando el slice falla porque casi no hay evidencia fiable.
Cambiar modelo	Modificar entrenamiento o representación.	Reentrenar con features, embeddings o pérdidas mejor controladas.	Cuando el score separa mal en ciertos segmentos.
Cambiar política	Modificar umbrales, revisión o automatización.	Ampliar banda de revisión de `0.38` a `0.32`.	Cuando el score es incierto y la consecuencia de mandar a flujo normal es alta.

La mitigación por política suele ser la primera que un equipo puede probar porque no exige reentrenar. También es peligrosa si se usa sin capacidad: cada caso enviado a revisión debe tener owner, SLA, registro y criterio de cierre.

De validación a producción

La auditoría no termina al generar `slice_metrics.csv`. En producción pueden cambiar los canales, idiomas, productos, prompts, documentos, colas humanas o criterios de etiqueta. Por eso los mismos slices deben convertirse en señales operativas.

SEÑAL EN PRODUCCIÓN	QUÉ MIDE	QUÉ ACCIÓN DEBERÍA DISPARAR
`review_rate` por slice	Carga humana que genera la política.	Ajustar capacidad, revisar umbral o limitar automatización.
`miss_rate` con etiqueta retrasada	Casos importantes que acabaron en flujo normal.	Abrir revisión de política y bloquear aumento de automatización.
Drift de distribución por slice	Cambio en canales, idiomas, productos o perfiles.	Comparar contra validación y revisar datos de entrenamiento.
Latencia p95 por slice	Si ciertos casos tardan más en resolverse.	Revisar pipeline, herramientas o colas específicas.
Tasa de apelación o corrección	Si usuarios o revisores corrigen más un segmento.	Revisar etiqueta, interfaz, rúbrica o explicación.

Esta es la conexión natural con DataOps: un SLI por slice no es una frase bonita, es una medida que se calcula. Un SLO por slice no es “queremos hacerlo bien”, es un objetivo que decide si se mantiene, se revisa o se detiene una política.

En el día a día

En un proyecto de IA aplicada, los slices deben definirse antes de mirar el resultado final. Puedes añadir nuevos slices cuando aparece una señal nueva, pero no deberías probar veinte cortes hasta encontrar el que cuenta la historia que querías contar.

Una plantilla mínima de decisión sería:

PREGUNTA	RESPUESTA QUE DEBE EXISTIR
¿Cuál es la unidad de decisión?	Caso, usuario, documento, consulta, turno, sesión.
¿Qué campos se usan como features?	Lista permitida y lista prohibida.
¿Qué campos se usan solo para auditoría?	Segmentos agregados, con permiso y propósito claro.
¿Qué slices son críticos?	Los que no pueden fallar aunque la media global pase.
¿Qué métrica gobierna cada slice?	Recall, miss rate, revisión, coste, latencia, calibración.
¿Qué gate decide?	Umbral medible de pass, review o block.
¿Qué acción sigue si falla?	Más datos, revisión de etiqueta, cambio de umbral, desautomatización, monitorización.

Para un ingeniero, lo más peligroso no es que el reporte diga `block`. Lo peligroso es que no exista reporte. Sin slices, la decisión parece técnica porque tiene números, pero no tiene trazabilidad.

Por qué debería importarte

Una auditoría por slices no es una capa estética de “IA responsable”. Es una herramienta de ingeniería para no publicar una política que falla justo donde el coste del error es más alto. Si el sistema prioriza bien en promedio, pero deja pasar casos críticos en un segmento pequeño, la media global te está contando una verdad incompleta.

También importa por operación. Cada mitigación cambia alguna carga: más revisión humana, más datos, más latencia, más coste de etiquetado, menos automatización o más complejidad de gobierno. Por eso este capítulo no trata solo de detectar diferencias: trata de convertir esas diferencias en decisiones defendibles. A veces la decisión será publicar. A veces será publicar con monitorización. A veces será ampliar datos. Y a veces será bloquear.

La pregunta profesional no es “¿hay sesgo?”. La pregunta es: **qué slice falla, con qué métrica, con qué muestra, con qué coste, qué acción dispara y quién se hace cargo**. Si no puedes contestar eso, todavía no tienes una auditoría; tienes una tabla.

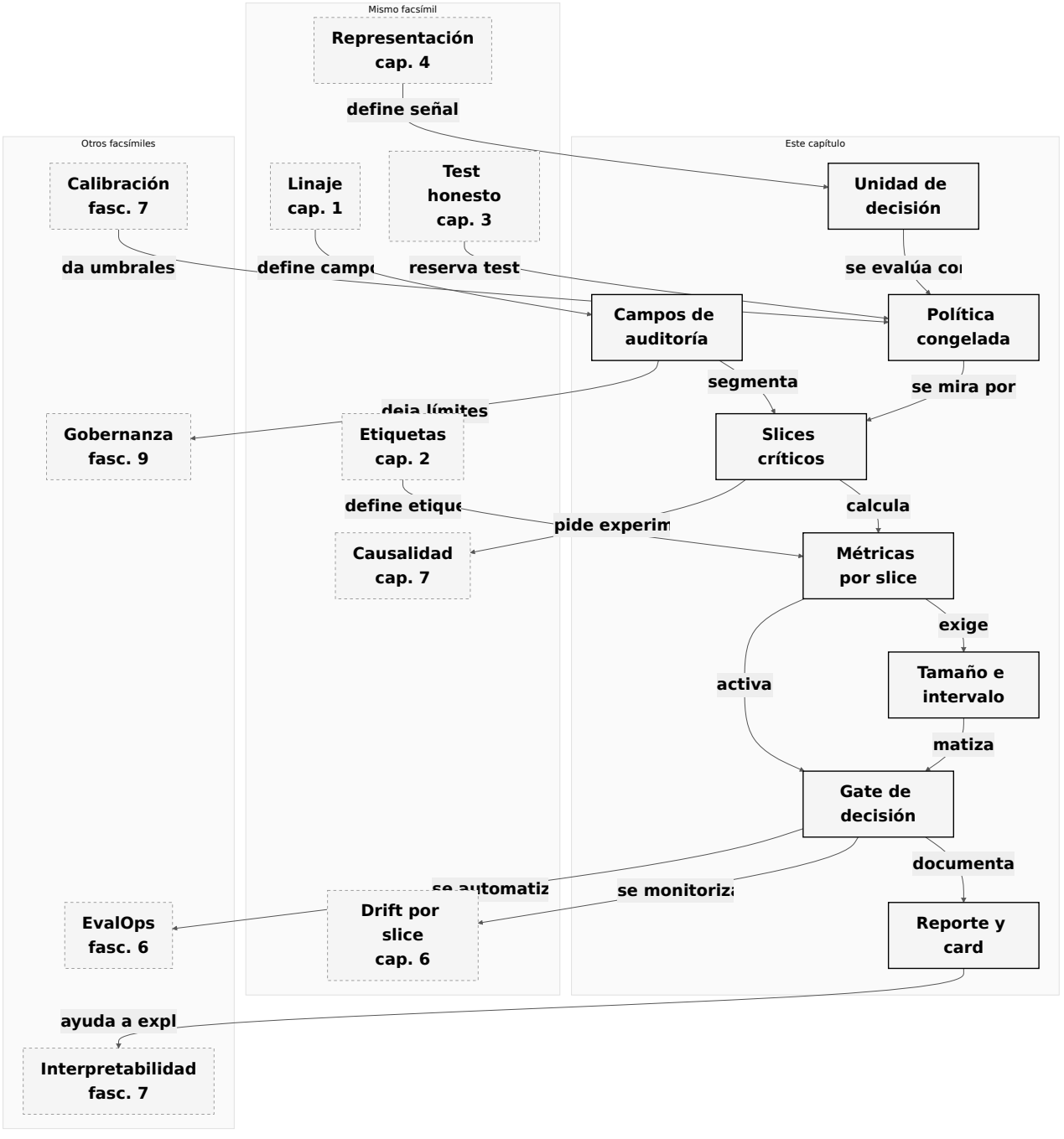
Dónde solía tropezar yo

TROPIEZO	POR QUÉ OCURRE	ANTÍDOTO
Mirar solo la media global	Es cómoda y fácil de explicar.	Exigir slices críticos antes de aprobar.
Partir por demasiadas columnas	Parece más completo.	Empezar por hipótesis de decisión y coste.
Confundir auditoría con feature	Si tengo el campo, parece que puedo usarlo.	Separar `fields_not_for_model` y `audit_fields`.
Ajustar umbrales mirando test	Es tentador arreglar el slice que falla.	Congelar umbrales desde validation y versionar la política.
Ignorar tamaños pequeños	Una tabla con números da sensación de rigor.	Añadir mínimos, intervalos y estado `review`.
Elegir una métrica de equidad sin contexto	La palabra suena suficiente.	Preguntar qué error controla y qué compromiso acepta.
No dejar acción concreta	El reporte se queda en diagnóstico.	Escribir decisión, dueño, gate y siguiente paso.

Cómo encaja todo

Este mapa debe leerse como continuidad de ingeniería. Los capítulos anteriores construyeron el suelo: linaje, calidad, split honesto y representación. Este capítulo pregunta si la política resultante se comporta de forma defendible en los segmentos que importan. Después, el capítulo 06 usará estos mismos slices como señales de monitorización: si producción cambia, no miraremos solo drift global.

La decisión central aquí es pasar de “mi métrica global mejora” a “sé qué slices sostienen o impiden automatizar”. Esa decisión conecta directamente con evaluación, calibración, interpretabilidad y gobernanza.


Mismo facsímil
Representación cap. 4
define señal
Otros facsímiles
Este capítulo
Calibración fasc. 7
Linaje cap. 1
Test honesto cap. 3
Unidad de decisión
da umbrales
define campo
reserva test
se evalúa co
Campos de auditoría
Política congelada
segmenta
se mira por
deja límites
Gobernanza fasc. 9
Etiquetas cap. 2
Slices críticos
define etique
pide experim
calcula
Causalidad cap. 7
Métricas por slice
exige
Tamaño e intervalo
activa
matiza
Gate de decisión
documenta
se automatiz
se monitoriza
EvalOps fasc. 6
Drift por slice cap. 6
Reporte y card
ayuda a expl
Interpretabilidad fasc. 7

Vocabulario aprendido

TÉRMINO	DEFINICIÓN BREVE
Slice	Subconjunto de datos definido por una condición útil para evaluar una decisión.
Atributo de auditoría	Campo usado para medir comportamiento por segmentos.
Atributo no usado por el modelo	Campo que no entra como feature, pero puede usarse para auditoría agregada.
Paridad demográfica	Comparación de tasas de selección entre grupos.
Igualdad de oportunidad	Comparación de verdaderos positivos entre grupos positivos reales.
Odds igualadas	Comparación simultánea de verdaderos positivos y falsos positivos entre grupos.
Calibración por grupo	Comprobación de que un score significa algo parecido en cada grupo.
Disparidad	Diferencia medible entre slices para una métrica concreta.
Captura segura	Proporción de casos importantes priorizados o enviados a revisión.
Gate	Regla medible que convierte métricas en decisión de release.
Proxy	Variable que sustituye a otra más difícil de medir, con riesgo de medir otra cosa.
OOV	Término fuera del vocabulario usado para construir una representación textual.
Par mínimo	Dos entradas casi iguales que solo cambian una pieza controlada.
WEAT	Test de asociación para medir relaciones entre grupos de palabras en embeddings.
Intersección	Slice definido por la combinación de varios atributos.

Antes de pasar página

Antes de avanzar, deberías poder responder:

1. ¿Qué es un slice y por qué no equivale a cualquier columna?

2. ¿Qué diferencia hay entre feature y atributo de auditoría?
3. ¿Por qué una métrica global puede esconder un problema importante?
4. ¿Qué mide la paridad demográfica?
5. ¿Qué mide la igualdad de oportunidad?
6. ¿Por qué odds igualadas mira verdaderos positivos y falsos positivos?
7. ¿Qué significa que algunas nociones de equidad sean incompatibles en ciertos escenarios?
8. ¿Por qué conviene separar auto-recall y captura segura?
9. ¿Qué coste asignarías a un falso negativo operativo y por qué?
10. ¿Por qué `access_need=si` debería bloquear una política de triaje?
11. ¿Qué archivo contiene la tabla plana de métricas por slice?
12. ¿Qué harías si un slice crítico tiene muestra insuficiente?
13. ¿Por qué no se deben ajustar umbrales mirando test?
14. ¿Cómo conectarías esta auditoría con una model card?
15. ¿Qué slices monitorizarías en producción en el capítulo siguiente?
16. ¿Qué diferencia hay entre sesgo de cobertura, medición, proxy y representación?
17. ¿Por qué los estudios de embeddings no se detectan igual que los de clasificación?
18. ¿Qué señal usarías para detectar falsos positivos asociados a términos concretos?
19. ¿Qué significa probar pares mínimos o contrafactuales?
20. ¿Qué parte del playbook adaptarías primero a tu proyecto?
21. ¿Por qué la política candidata pasa de `block` a `review`, pero no a `pass`?
22. ¿Qué trade-off aparece al ampliar la banda de revisión?
23. ¿Qué representa $\gamma_j(h)$ en una mitigación con restricciones?
24. ¿Qué pedirías a una herramienta antes de incorporarla al ciclo de release?
25. ¿Qué SLI por slice monitorizarías cuando el sistema llegue a producción?

En resumen

IDEA	QUÉ TE LLEVAS
La media global no decide sola.	Hay que mirar slices conectados con consecuencias reales.
Un campo puede auditar sin decidir.	No todo atributo útil para medir debe entrar al modelo.
La métrica depende del error.	Paridad, oportunidad, odds y calibración responden preguntas distintas.
El tamaño importa.	Un slice pequeño pide revisión o más datos, no conclusión fuerte.
La auditoría debe producir acción.	Reporte, CSV, card y decisión son artefactos de ingeniería.
Los sesgos tienen mecanismos distintos.	Cobertura, proxy, etiqueta, representación y política requieren pruebas diferentes.
Las buenas prácticas se escriben antes de test.	Si eliges slices y gates después, la auditoría pierde fuerza.
Mitigar implica trade-offs.	La banda de revisión reduce pérdida, pero aumenta carga humana.
Las herramientas no deciden solas.	Fairlearn, AIF360, Clarify o Evidently ayudan si hay contrato, slices y gates.
Producción cambia la pregunta.	Los mismos slices deben convertirse en SLI, SLO, alerta y runbook.

Para saber más

Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J. y Wallach, H. (2018). A Reductions Approach to Fair Classification. *ICML,* 60-69. PMLR

Amazon Web Services. (2026). *Amazon SageMaker Clarify*. Documentación

Arize AI. (2026). *ML Observability Platform*. Producto

Arthur AI. (2020). *Product Update - Bias Monitoring v2.1*. Blog

Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K. N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R. y Zhang, Y. (2019). AI Fairness 360: An Extensible Toolkit for Detecting and Mitigating Algorithmic Bias. *IBM Journal of Research and Development,* 63(4/5), 4:1-4:15. DOI

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V. y Kalai, A. T. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. *Advances in Neural Information Processing Systems 29*, 4349-4357. Paper

Buolamwini, J. y Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. *Proceedings of Machine Learning Research*, 81, 77-91. Paper

Caliskan, A., Bryson, J. J. y Narayanan, A. (2017). Semantics Derived Automatically from Language Corpora Contain Human-Like Biases. *Science*, 356(6334), 183-186. DOI

Center for Data Science and Public Policy. (2026). *Aequitas documentation*. Documentación

Chouldechova, A. (2017). Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data*, 5(2), 153-163. DOI

Dixon, L., Li, J., Sorensen, J., Thain, N. y Vasserman, L. (2018). Measuring and Mitigating Unintended Bias in Text Classification. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 67-73. Google Research

Dwork, C., Hardt, M., Pitassi, T., Reingold, O. y Zemel, R. (2012). Fairness Through Awareness. *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 214-226. DOI

Evidently AI. (2026). *Evidently documentation*. Documentación

Fairlearn. (2026). *Assessment: Performing a Fairness Assessment*. Documentación

Fairlearn. (2026). *Mitigations*. Documentación

Fawcett, T. (2006). An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8), 861-874. DOI

Fiddler AI. (2026). *Fairness*. Documentación

Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H. y Crawford, K. (2021). Datasheets for Datasets. *Communications of the ACM*, 64(12), 86-92. DOI

Giskard. (2026). *Giskard documentation*. Documentación

Hardt, M., Price, E. y Srebro, N. (2016). Equality of Opportunity in Supervised Learning. *Advances in Neural Information Processing Systems 29*, 3323-3331. Paper

IBM Research. (2026). *AI Fairness 360 documentation*. Documentación

Kleinberg, J., Mullainathan, S. y Raghavan, M. (2017). *Inherent Trade-Offs in the Fair Determination of Risk Scores*. arXiv

Microsoft. (2026). *Use the Responsible AI dashboard in Azure Machine Learning studio*. Documentación

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. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 220-229. DOI

Nadeem, M., Bethke, A. y Reddy, S. (2021). StereoSet: Measuring Stereotypical Bias in Pretrained Language Models. *ACL-IJCNLP 2021*, 5356-5371. DOI

Nangia, N., Vania, C., Bhalerao, R. y Bowman, S. R. (2020). CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. *EMNLP 2020*, 1953-1967. DOI

Obermeyer, Z., Powers, B., Vogeli, C. y Mullainathan, S. (2019). Dissecting Racial Bias in an Algorithm Used to Manage the Health of Populations. *Science*, 366(6464), 447-453. DOI

Parrish, A., Chen, A., Nangia, N., Padmakumar, V., Phang, J., Thompson, J., Htut, P. M. y Bowman, S. R. (2022). BBQ: A Hand-Built Bias Benchmark for Question Answering. *Findings of ACL 2022*, 2086-2105. DOI

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. arXiv

Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D. y Barnes, P. (2020). *Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing*. arXiv

Responsibly. (2026). *Responsibly documentation*. Documentación

Scikit-learn. (2026). *Classification metrics*. Documentación

Scikit-learn. (2026). *confusion_matrix*. Documentación

TensorFlow. (2026). *Fairness Indicators*. GitHub

Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F. y Wilson, J. (2019). The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics*. Google Research

WhyLabs. (2026). *WhyLabs documentation*. Documentación

Notas

1. 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 ↩

2. 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. https://arxiv.org/abs/2010.16061 ↩

3. Scikit-learn. (2026). *confusion_matrix*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. Consultado el 7 de junio de 2026. ↩

4. Dwork, C., Hardt, M., Pitassi, T., Reingold, O. y Zemel, R. (2012). Fairness Through Awareness. *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 214-226. https://doi.org/10.1145/2090236.2090255 ↩

5. Hardt, M., Price, E. y Srebro, N. (2016). Equality of Opportunity in Supervised Learning. *Advances in Neural Information Processing Systems 29*, 3323-3331. https://papers.nips.cc/paper/6374-equality-of-opportunity-in-supervised-learning ↩

6. Chouldechova, A. (2017). Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data*, 5(2), 153-163. https://doi.org/10.1089/big.2016.0047 ↩

7. Kleinberg, J., Mullainathan, S. y Raghavan, M. (2017). *Inherent Trade-Offs in the Fair Determination of Risk Scores*. https://arxiv.org/abs/1609.05807 ↩

8. Obermeyer, Z., Powers, B., Vogeli, C. y Mullainathan, S. (2019). Dissecting Racial Bias in an Algorithm Used to Manage the Health of Populations. *Science*, 366(6464), 447-453. https://doi.org/10.1126/science.aax2342 ↩

9. Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V. y Kalai, A. T. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. *NeurIPS 2016*, 4349-4357. https://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings ↩

10. Caliskan, A., Bryson, J. J. y Narayanan, A. (2017). Semantics Derived Automatically from Language Corpora Contain Human-Like Biases. *Science*, 356(6334), 183-186. https://doi.org/10.1126/science.aal4230 ↩

11. Buolamwini, J. y Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. *Proceedings of Machine Learning Research*, 81, 77-91. https://proceedings.mlr.press/v81/buolamwini18a.html ↩

12. Dixon, L., Li, J., Sorensen, J., Thain, N. y Vasserman, L. (2018). Measuring and Mitigating Unintended Bias in Text Classification. *AIES 2018*, 67-73. https://research.google/pubs/measuring-and-mitigating-unintended-bias-in-text-classification/ ↩

13. Nangia, N., Vania, C., Bhalerao, R. y Bowman, S. R. (2020). CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. *EMNLP 2020*, 1953-1967. https://doi.org/10.18653/v1/2020.emnlp-main.154 ↩

.4. Nadeem, M., Bethke, A. y Reddy, S. (2021). StereoSet: Measuring Stereotypical Bias in Pretrained Language Models. *ACL-IJCNLP 2021*, 5356-5371. https://doi.org/10.18653/v1/2021.acl-long.416 ↩

.5. Parrish, A. y otros (2022). BBQ: A Hand-Built Bias Benchmark for Question Answering. *Findings of ACL 2022*, 2086-2105. https://doi.org/10.18653/v1/2022.findings-acl.165 ↩

.6. Fairlearn. (2026). *Assessment: Performing a Fairness Assessment*. https://fairlearn.org/main/user_guide/assessment/. Consultado el 7 de junio de 2026. ↩

.7. Fairlearn. (2026). *Mitigations*. https://fairlearn.org/main/user_guide/mitigation/index.html. Consultado el 7 de junio de 2026. ↩

.8. IBM Research. (2026). *AI Fairness 360 documentation*. https://aif360.readthedocs.io/en/stable/index.html. Consultado el 7 de junio de 2026. ↩

.9. Bellamy, R. K. E. y otros (2019). AI Fairness 360: An Extensible Toolkit for Detecting and Mitigating Algorithmic Bias. *IBM Journal of Research and Development*, 63(4/5), 4:1-4:15. https://doi.org/10.1147/JRD.2019.2942287 ↩

0. Center for Data Science and Public Policy. (2026). *Aequitas documentation*. https://dssg.github.io/aequitas/. Consultado el 7 de junio de 2026. ↩

1. TensorFlow. (2026). *Fairness Indicators*. https://github.com/tensorflow/fairness-indicators. Consultado el 7 de junio de 2026. ↩

2. Wexler, J. y otros (2019). The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE TVCG*. https://research.google/pubs/the-what-if-tool-interactive-probing-of-machine-learning-models/ ↩

3. Responsibly. (2026). *Responsibly documentation*. https://docs.responsibly.ai/. Consultado el 7 de junio de 2026. ↩

4. Amazon Web Services. (2026). *Amazon SageMaker Clarify*. https://aws.amazon.com/sagemaker/ai/clarify/. Consultado el 7 de junio de 2026. ↩

5. Microsoft. (2026). *Use the Responsible AI dashboard in Azure Machine Learning studio*. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-responsible-ai-dashboard. Consultado el 7 de junio de 2026. ↩

6. Evidently AI. (2026). *Evidently documentation*. https://docs.evidentlyai.com/docs/library/overview. Consultado el 7 de junio de 2026. ↩

7. Giskard. (2026). *Giskard documentation*. https://docs.giskard.ai/. Consultado el 7 de junio de 2026. ↩

8. Fiddler AI. (2026). *Fairness*. https://docs.fiddler.ai/observability/fairness. Consultado el 7 de junio de 2026. ↩

29. Arize AI. (2026). *ML Observability Platform*. https://arize.com/capabilities/. Consultado el 7 de junio de 2026. ↩

30. WhyLabs. (2026). *WhyLabs documentation*. https://docs.whylabs.ai/docs/. Consultado el 7 de junio de 2026. ↩

31. Arthur AI. (2020). *Product Update - Bias Monitoring v2.1*. https://www.arthur.ai/blog/product-update-bias-monitoring-v21. Consultado el 7 de junio de 2026. ↩

32. Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J. y Wallach, H. (2018). A Reductions Approach to Fair Classification. *ICML*, 60-69. https://proceedings.mlr.press/v80/agarwal18a.html ↩

Cuadernos para practicar

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

Slices y sesgo: el promedio miente

https://colab.research.google.com/github/686f6c61/iaparagentecuriosa-notebooks/blob/main/Facs%C3%ADmil%2008/05-slices-sesgo-el-promedio-miente.ipynb