Parte 7 — Ética, Fairness y Privacidad

6 clases · Parte 7 del programa

Parte 7 — Ética, Fairness y Privacidad

> *6 clases · bundle consolidado del currículo v3.*

Índice de clases

Clase 223 — Clase 223 — Tipos de sesgo algorítmico y orígenes

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: Suresh & Guttag, A Framework for Understanding Sources of Harm throughout the ML Life Cycle (EAAMO 2021) + Barocas, Hardt, Narayanan, Fairness and Machine Learning (2023), caps. 1-2. Duración estimada: 75 min.*

Objetivo

Aprender a nombrar y diagnosticar el origen del sesgo en un sistema ML antes de intentar mitigarlo. Un modelo "sesgado" no es un bug: es el resultado de decisiones tomadas en cada fase del life cycle (recolección, medición, modelado, evaluación, despliegue). Si no sabemos dónde entró el sesgo, no podemos elegir la mitigación correcta (Clases 225-227).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Distinguir los 6 tipos del framework Suresh-Guttag: histórico, representación, medición, agregación, evaluación, despliegue.
- Identificar el origen probable de un sesgo dado evidencia empírica (gap de accuracy entre subgrupos, drift, proxy-target gap).
- Reproducir el patrón de Gender Shades (Buolamwini & Gebru 2018): accuracy alta global, accuracy baja en subgrupo minoritario.
- Reconocer Simpson's paradox y por qué un modelo único puede ser peor que un modelo por subgrupo.
- Justificar por qué fairness no es solo un problema del modelo — empieza en la definición de la tarea.

Temas

#	Tema	Por qué importa
1	Sesgo histórico	El dato es "correcto" pero refleja un mund
2	Sesgo de representación	Subgrupos sub-muestreados → modelo aprende
3	Sesgo de medición	El proxy ≠ el target real (re-arresto ≠ re
4	Sesgo de agregación	Un modelo único asume que todos los subgru

5	Sesgo de evaluación	El benchmark de test no representa la pobl
6	Sesgo de despliegue	El modelo se usa fuera del contexto en que

Definiciones y características

- Sesgo histórico: la realidad social que se midió ya era injusta. El dato es preciso, pero codifica desigualdad. Ej.: Amazon entrenó su hiring tool con 10 años de CVs (~mayoría hombres en tech) → penalizaba la palabra "women's" en CVs.
- Sesgo de representación: el dataset bajo-representa un subgrupo. No es que el modelo sea injusto: nunca vio suficientes ejemplos. Ej.: IJB-A face dataset era 79% piel clara → Gender Shades mostró error 34.7% en mujeres de piel oscura vs 0.8% en hombres de piel clara (Buolamwini & Gebru, 2018).
- Sesgo de medición (proxy bias): el y que medimos no es el y que queremos. Recidiva (queremos predecir) ≠ re-arresto (lo que medimos) — el arresto depende del policing, que ya está sesgado.
- Sesgo de agregación: un modelo único asume una única función óptima f(x) → y para toda la población. Si los subgrupos tienen distintas P(y | x), el modelo único es peor que k modelos por subgrupo (instancia del Simpson's paradox).
- Sesgo de evaluación: el test set no representa la población objetivo. Ej.: evaluar un detector de melanoma sobre un benchmark 90% piel clara y reportar "97% accuracy".
- Sesgo de despliegue: el modelo se aplica a una distribución distinta a la de entrenamiento (covariate shift, label shift) — o a un contexto socio-técnico donde su salida significa otra cosa. Un score que era "una sugerencia para el juez" se vuelve "la decisión".
- Framework Suresh-Guttag (2021): mapea los 6 sesgos a las fases del ML life cycle (data collection → preparation → modeling → evaluation → deployment). Cada fase introduce su propio harm; mitigar en la fase equivocada no funciona.
- Harm allocacional vs representacional (Barocas et al., cap. 1): el sistema deniega recursos (préstamo, libertad bajo fianza) vs refuerza estereotipos (autocomplete de Google asocia "CEO" con foto de hombre).

Dataset / recursos

- Dataset: sintético (préstamos con sesgo histórico inyectado). Auto-generado en el notebook con numpy seed 42.
- Librerías: numpy, pandas, scikit-learn. Sin descargas externas.

Ejercicios

1. Sesgo histórico: generar un dataset de préstamos donde P(aprobado | grupo=A) = 0.70 y P(aprobado | grupo=B) = 0.30 por razones históricas (no por capacidad de pago). Entrenar LogisticRegression sin la feature grupo y mostrar que el modelo igual reproduce el gap vía proxies (código postal, ingreso, etc.).
2. Selection rate disparity: calcular P(ŷ=1 | grupo=A) vs P(ŷ=1 | grupo=B) — la métrica más simple de demographic parity (Clase 224).
3. Sesgo de representación (Gender Shades): re-muestrear el dataset al 10% del grupo B. Reportar accuracy global vs accuracy por subgrupo. Mostrar el patrón "97% global, 60% en B".
4. Sesgo de medición: definir y_proxy = y_true XOR ruido_correlacionado_con_grupo. Entrenar sobre y_proxy. Mostrar que el modelo aprende el patrón del ruido, no del target real.
5. Sesgo de agregación (Simpson): comparar AUC de un modelo único vs un modelo por subgrupo. Mostrar que el modelo único es subóptimo en ambos subgrupos.

Homework verificable

Notebook con:

1. Reproducir el patrón Gender Shades sobre un dataset tabular (sintético o UCI Adult con sex y race).
2. Calcular gap de accuracy entre subgrupos para 3 niveles de sub-muestreo del grupo minoritario (50%, 20%, 5%).
3. Aplicar el framework Suresh-Guttag a un caso real (COMPAS, Amazon Hiring, o un modelo del trabajo) — escribir 1 párrafo por cada uno de los 6 tipos: ¿está presente? evidencia.
4. Implementar Simpson's paradox: dataset donde la correlación global es opuesta a la correlación intra-grupo.
5. Discutir: ¿qué mitigación corresponde a cada tipo? (representación → re-sampling; medición → re-definir target; agregación → modelo por subgrupo).

Criterio de aceptación: el alumno identifica los 6 tipos en al menos un caso real con evidencia cuantitativa, y propone una mitigación específica por tipo (no genérica "más datos").

Errores comunes

Síntoma	Causa y cómo arreglar
"Saqué la variable sensible y el modelo si	Fairness through unawareness no funciona —
Accuracy global alta pero el cliente repor	Sesgo de representación o evaluación. Fix:
El modelo es "objetivo, solo aprendió del	El dato no es neutro — refleja decisiones
"Agregamos más datos del grupo minoritario	Probablemente sesgo de medición — el y par
Modelo único performa peor que k modelos p	Sesgo de agregación. Fix: modelo por subgr
Modelo cae en producción pero pasó test	Sesgo de evaluación/despliegue. Fix: test

Preguntas frecuentes

¿Por qué no simplemente sacar la variable sensible (sex, race)?

Eso es fairness through unawareness y casi nunca funciona. Las features que sí usás (ZIP, nombre, historial crediticio, escuela) son proxies correlacionados. Quitar la variable sensible te impide medir el sesgo pero no lo elimina. Lo que sí se hace: usar la variable sensible para auditar disparidades, no como input del modelo.

¿Sesgo histórico vs sesgo de representación — cuál es la diferencia?

Histórico = el dato es preciso pero el mundo medido era injusto (mujeres ganan menos → modelo de salarios predice salarios más bajos para mujeres). Representación = el dato no representa bien a un subgrupo (pocas caras oscuras en ImageNet → mala accuracy ahí). El primero requiere reescribir el target o el objetivo; el segundo, recolectar/re-muestrear.

¿Simpson's paradox es realmente común en ML?

Sí, especialmente en datasets con subgrupos heterogéneos (Berkeley admissions 1973 es el clásico). Si tu accuracy global mejora pero la accuracy por subgrupo empeora, es Simpson. Por eso siempre se reporta métrica stratificada.

¿Esto se resuelve con un algoritmo de fairness?

No solo. Los algoritmos (re-weighting, adversarial debiasing, Clase 226) atacan principalmente representación y agregación. El sesgo histórico y de medición requieren decisiones humanas: ¿qué estamos prediciendo? ¿es el target correcto? Ningún optimizador resuelve "el target estaba mal definido".

¿Cómo encaja todo esto con Clases 224 (métricas) y 225-227 (mitigación)?

223 = diagnóstico (qué tipo de sesgo y dónde nace). 224 = medición cuantitativa (demographic parity, equalized odds, calibration). 225-227 = mitigación (pre-procesamiento, in-procesamiento,

post-procesamiento). El orden importa: sin diagnóstico, la mitigación es a ciegas.

Referencias

- Suresh, H., Guttag, J. A Framework for Understanding Sources of Harm throughout the ML Life Cycle (EAAMO 2021) — el framework canónico de los 6 tipos.
- Buolamwini, J., Gebru, T. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification (FAT* 2018) — el paper que disparó la auditoría algorítmica.
- Barocas, S., Hardt, M., Narayanan, A. Fairness and Machine Learning: Limitations and Opportunities (MIT Press, 2023) — caps. 1-2 (libre online).
- Mehrabi, N. et al. A Survey on Bias and Fairness in Machine Learning (ACM Computing Surveys, 2021).
- Angwin, J. et al. Machine Bias (ProPublica, 2016) — la investigación sobre COMPAS.
- Dastin, J. Amazon scraps secret AI recruiting tool that showed bias against women (Reuters, 2018).

Material descargable

- Guía explicativa (PDF) — versión imprimible con todo el contenido de la clase.
- Presentación (PPTX) — deck PowerPoint listo para proyectar en clase.
- Notebook ejecutable (.ipynb) — abrilo desde el laboratorio del programa o desde Jupyter.

Clase 224 — Clase 224 — Métricas de fairness: demographic parity, equalized odds, calibration

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: Barocas, Hardt, Narayanan — Fairness and Machine Learning cap. 3 + Hardt, Price, Srebro (NeurIPS 2016) Equality of Opportunity in Supervised Learning. Duración estimada: 75 min.*

Objetivo

Pasar de "el modelo es injusto" a medirlo con un número. Implementar las tres familias de métricas grupales que dominan la literatura — demographic parity, equalized odds, calibration — sobre un dataset binario con atributo protegido, y demostrar numéricamente el teorema de imposibilidad de Kleinberg-Mullainathan-Raghavan / Chouldechova (2017): salvo casos triviales, no se pueden satisfacer las tres a la vez.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Calcular demographic parity gap = $|P(\hat{Y}=1|A=0) - P(\hat{Y}=1|A=1)|$ sobre predicciones de cualquier clasificador binario.
- Calcular equal opportunity (TPR por grupo) y equalized odds (TPR y FPR por grupo) — Hardt, Price, Srebro 2016.
- Verificar calibración por grupo con reliability curves: $P(Y=1|\hat{S}=s, A=a)$ debe ser igual entre grupos para un mismo score s.
- Demostrar el teorema de imposibilidad: ajustar threshold por grupo para forzar demographic parity rompe calibración.
- Aplicar mitigación post-processing con thresholds por grupo (Hardt 2016) y reportar el trade-off accuracy vs fairness gap.

Temas

#	Tema	Por qué importa
1	Atributo protegido A y notación Y / Ŷ / Ŝ	Sin notación común no se discute fairness;
2	Demographic parity (statistical parity)	La métrica más antigua (regla del 80%, US
3	Equal opportunity y equalized odds (Hardt	Condicionan en Y — corrigen el defecto de
4	Calibration por grupo (Chouldechova 2017)	El score debe significar lo mismo en cada
5	Teorema de imposibilidad (KMR / Chouldech	Si las base-rates difieren, DP + equalized
6	Post-processing: threshold por grupo	Mitigación más simple; revela explícitamen

Definiciones y características

- Atributo protegido A: variable demográfica sensible (sexo, raza, edad). En la práctica nunca está sola — hay proxies (zip code, nombre, historial).
- Demographic parity (DP): P(Ŷ=1 | A=0) = P(Ŷ=1 | A=1). Métrica: DP_gap = |selection_rate(A=0) − selection_rate(A=1)|. Regla del 80%: ratio ≥ 0.8 para que la US EEOC no lo considere discriminación.
- Equal opportunity: P(Ŷ=1 | Y=1, A=0) = P(Ŷ=1 | Y=1, A=1). O sea, TPR igual. Solo a los positivos reales se les exige misma tasa de aceptación.
- Equalized odds (Hardt-Price-Srebro 2016): equal opportunity + FPR igual. EO_gap = max(|TPR_diff|, |FPR_diff|).
- Calibration (predictive parity, Chouldechova 2017): P(Y=1 | Ŝ=s, A=a) es igual entre grupos para todo score s. Un score 0.7 significa "70% chance" tanto para A=0 como para A=1.
- Impossibility theorem (Kleinberg-Mullainathan-Raghavan 2017 + Chouldechova 2017): si P(Y=1|A=0) ≠ P(Y=1|A=1) (base rates diferentes) y el clasificador no es perfecto, no existe clasificador que sea simultáneamente calibrado por grupo y con equalized odds. Es matemático, no técnico — no se resuelve con más datos.
- Accuracy-fairness trade-off: forzar cualquier paridad sobre un clasificador óptimo de Bayes baja accuracy. La pregunta política es cuánta accuracy estamos dispuestos a sacrificar.
- Tooling: fairlearn (Microsoft) — MetricFrame, ThresholdOptimizer. aif360 (IBM) — 70+ métricas y mitigadores. Ambas open source.

Dataset / recursos

- Dataset notebook: sintético binario con un atributo protegido A{0,1} y base rates diferentes (60% vs 40%) — necesario para que el teorema de imposibilidad se active.
- Dataset real recomendado para tarea: Adult / Census Income (UCI) con sex como atributo protegido, o COMPAS (ProPublica) con race.
- Librerías: numpy, pandas, scikit-learn. Opcional: fairlearn.

Ejercicios

1. Selection rate por grupo: entrenar LogisticRegression baseline. Calcular P(Ŷ=1|A=0) y P(Ŷ=1|A=1) y el DP_gap. ¿Cumple regla del 80%?
2. TPR y FPR por grupo: armar confusion_matrix separada por grupo. Calcular equal_opportunity_gap = |TPR_0 − TPR_1| y equalized_odds_gap = max(|TPR_diff|, |FPR_diff|).
3. Calibration curves por grupo: binning de scores en 10 bins. Para cada bin y cada grupo, graficar mean(y_true) vs mean(y_score). ¿Las curvas coinciden?
4. Romper calibración: ajustar threshold por grupo (t_0, t_1) tal que se cumpla DP exacta. Recalcular calibración — debe degradarse. (Demostración numérica del teorema.)
5. Post-processing Hardt: buscar (t_0, t_1) que minimicen equalized_odds_gap y reportar el costo en accuracy global. Tabla: baseline vs DP-fixed vs EO-fixed.

Homework verificable

Notebook con:

1. Cargar Adult Census (UCI). Atributo protegido: sex. Target: income > 50K.
2. Entrenar baseline LogisticRegression + reportar accuracy, AUC.
3. Calcular las tres métricas: DP_gap, equal_opportunity_gap, equalized_odds_gap y calibration_gap (max |calibration(A=0) − calibration(A=1)| sobre bins).
4. Implementar post-processing con threshold por grupo que minimice EO_gap sujeto a accuracy_drop ≤ 3pp.
5. Tabla final: 3 modelos (baseline, DP-mitigated, EO-mitigated) × 5 métricas (accuracy, AUC, DP_gap, EO_gap, calibration_gap). Discutir cuál elegirías y por qué — no hay respuesta universal.

Criterio de aceptación: las 4 métricas implementadas a mano (no llamando a fairlearn), el EO_gap mitigado < 0.05, y un párrafo justificando la elección de métrica en función del dominio de aplicación (crédito vs admisión universitaria vs medicina).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
DP_gap = 0 pero el modelo es claramente in	DP ignora el ground truth — si las base ra
Calibration "perfecta" pero TPR diferente	Es el resultado natural cuando las base ra
Quitar el atributo protegido del input "so	Fairness through unawareness — no funciona
Threshold único 0.5 para todos los grupos	Asume que los scores significan lo mismo e
Usar accuracy global para comparar fairnes	Accuracy global puede subir y empeorar al
MetricFrame de fairlearn devuelve nan	Algún grupo tiene 0 positivos predichos o

Preguntas frecuentes

¿Cuál de las tres métricas uso?

Depende del dominio. Crédito o contratación: equal opportunity (no negarle empleo a quien sí calificaría). Justicia penal / riesgo de reincidencia: calibration + FPR equal (el debate ProPublica vs COMPAS giró exactamente sobre cuál priorizar). Publicidad: DP suele alcanzar. Pero la elección es política, no técnica.

¿Demographic parity no es siempre lo que queremos?

No. Si la base rate real difiere entre grupos (ej. distribución de ingresos), DP puede forzar al modelo a aceptar candidatos peores de un grupo y rechazar mejores de otro. Equal opportunity suele ser más defendible.

¿El teorema de imposibilidad significa que fairness es imposible?

No — significa que no se pueden satisfacer las tres simultáneamente cuando las base rates difieren. Hay que elegir cuál priorizar, y documentarlo. El paper de Chouldechova (2017) es lectura obligada.

¿fairlearn o aif360?

fairlearn para empezar (API más limpia, integrada con sklearn). aif360 cuando se necesite catálogo amplio de mitigadores (pre/in/post-processing) y métricas exóticas. Ninguna sustituye entender las definiciones.

¿Y la fairness individual?

Otra familia (Dwork et al. 2012): "individuos similares deben recibir predicciones similares". Más difícil de operacionalizar (requiere métrica de similaridad justificable) y queda fuera del alcance de esta clase. Ver Clase 225-227 para privacy y Clase 228 para causal fairness.

Referencias

- Hardt, M., Price, E., Srebro, N. Equality of Opportunity in Supervised Learning, NeurIPS 2016. https://arxiv.org/abs/1610.02413
- Chouldechova, A. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments, Big Data 2017. https://arxiv.org/abs/1610.07524
- Kleinberg, J., Mullainathan, S., Raghavan, M. Inherent Trade-Offs in the Fair Determination of Risk Scores, ITCS 2017. https://arxiv.org/abs/1609.05807
- Barocas, S., Hardt, M., Narayanan, A. Fairness and Machine Learning (fairmlbook.org), cap. 3 — Classification.
- fairlearn documentation — Microsoft, MIT license.
- AIF360 documentation — IBM, Apache 2.0.

Material descargable

- Guía explicativa (PDF) — versión imprimible con todo el contenido de la clase.
- Presentación (PPTX) — deck PowerPoint listo para proyectar en clase.
- Notebook ejecutable (.ipynb) — abrilo desde el laboratorio del programa o desde Jupyter.

Clase 225 — Clase 225 — Privacidad diferencial: intro

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: Dwork & Roth, The Algorithmic Foundations of Differential Privacy (2014) caps. 2-3 + Dwork, McSherry, Nissim, Smith (TCC, 2006) Calibrating Noise to Sensitivity. Duración estimada: 75 min.*

Objetivo

Entender privacidad diferencial (DP) como la única definición formal de privacidad con garantías matemáticas — no "anonimización" heurística que se rompe con un join. Implementar el mecanismo de Laplace desde cero, observar el trade-off privacy-utility vía el presupuesto ε (epsilon), y mirar conceptualmente DP-SGD (Abadi et al. 2016): cómo se entrena un modelo sin que un atacante pueda inferir si tu registro estuvo en el training set.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Enunciar la definición de (ε, δ)-DP: P[M(D) S] ≤ e^ε · P[M(D') S] + δ para datasets vecinos D, D'.
- Calcular la sensibilidad Δf de funciones típicas (conteos, sumas acotadas, medias) y elegir ruido Laplace o Gaussiano calibrado.
- Implementar laplace_mechanism(value, sensitivity, epsilon) y verificar que ε chico → más ruido → menos utilidad.
- Aplicar composición básica: k consultas con ε cada una gastan k·ε del presupuesto total.
- Reconocer la idea de DP-SGD: per-sample gradient clipping + ruido gaussiano → entrenamiento DP (Opacus, TF-Privacy).

Temas

#	Tema	Por qué importa
1	Anonimización falla (Netflix Prize, AOL se	k-anonymity / pseudonimización son rotas p
2	Definición (ε, δ)-DP y datasets vecinos	El ε es la garantía; sin él, "privacidad"
3	Sensibilidad Δf	Calibra cuánto ruido hace falta. Conteo: Δ

4	Mecanismos Laplace y Gaussiano	Laplace para ε-DP puro; Gaussiano para (ε,
5	Composición y post-processing	Cada query gasta presupuesto; cualquier f(
6	DP-SGD (Abadi 2016)	Clip per-sample + ruido gaussiano. Es el e

Definiciones y características

- (ε, δ)-Differential Privacy (Dwork 2006): mecanismo aleatorizado M es (ε, δ)-DP si para todo par de datasets vecinos D, D' (que difieren en 1 registro) y todo conjunto de salidas S: P[M(D) S] ≤ e^ε · P[M(D') S] + δ. Si δ = 0 se llama ε-DP puro.
- Privacy budget ε: chico = más privacidad, menos utilidad. Valores típicos en la práctica: ε=0.1 (fuerte), ε=1 (estándar de la US Census 2020), ε=10 (débil — más marketing que garantía).
- δ: probabilidad de fallar la garantía. Regla: δ << 1/n (con n = tamaño del dataset).
- Sensibilidad Δf (L1): Δf = max_{D,D' vecinos} |f(D) − f(D')|. Conteo de registros: Δf=1. Suma de valores en [0, B]: Δf=B. Media sobre n fijo y valores en [0, B]: Δf = B/n.
- Mecanismo de Laplace: M(D) = f(D) + Lap(0, Δf/ε). Cumple ε-DP puro.
- Mecanismo Gaussiano: M(D) = f(D) + N(0, σ²) con σ ≥ sqrt(2 ln(1.25/δ)) · Δf / ε. Cumple (ε, δ)-DP.
- Composición básica: k mecanismos ε_i-DP componen a (Σ ε_i)-DP. Composición avanzada da sqrt(2k ln(1/δ)) · ε — mejor escala.
- Post-processing: si M es (ε, δ)-DP, entonces g(M(D)) también — no podés "des-privatizar" mirando la salida.
- DP-SGD (Abadi et al. 2016): en cada paso (1) calcular gradiente per-sample, (2) clipearlo a norma C, (3) promediar el batch, (4) sumar ruido gaussiano N(0, σ²C²). Tracking del ε vía moments accountant / RDP.

Dataset / recursos

- Dataset: sintético — un dataframe de salarios n=10_000 con valores en [0, 200_000]. Suficiente para Laplace, mean privado, histograma y DP-SGD demo. Sin descarga externa.
- Librerías: numpy, pandas, scikit-learn. En producción real: opacus (PyTorch), tensorflow-privacy, diffprivlib (IBM).

Ejercicios

1. Laplace básico: implementar laplace_mechanism(value, sensitivity, epsilon) y verificar empíricamente sobre 10_000 corridas que la varianza es 2·(Δf/ε)².
2. Conteo privado: contar empleados con salario > 100k con ε {0.1, 1.0, 10.0}. Reportar error medio absoluto y discutir el trade-off.
3. Mean privado con clipping: clip salarios a [0, B], sumar con Laplace (Δf=B/n, ε=1), dividir por n. Mostrar bias vs varianza al variar B.
4. Histograma privado: 10 bins de salario, ruido Laplace independiente por bin (sensibilidad = 1 por bin). Comparar con histograma no privado.
5. Composición: hacer 10 conteos con ε=0.1 cada uno → presupuesto total ε=1.0. Mostrar acumulación empírica del ruido.

Homework verificable

Notebook con:

1. Cargar Adult / Census Income (UCI, ~32K filas).
2. Publicar un dashboard DP con 5 estadísticas (count, mean age, mean hours-per-week, count por género, count por education) bajo presupuesto total ε=1.0. Repartir el presupuesto entre queries y justificar.

3. Entrenar un LogisticRegression clásico para predecir income > 50k, reportar accuracy.
4. Re-entrenar con DP-SGD manual: clip per-sample gradient norm a C=1.0, sumar N(0, σ²) con σ=1.0. Reportar accuracy y comparar.
5. Discutir: ¿cuánta utilidad perdés? ¿el modelo DP es publicable sin riesgo de membership inference?

Criterio de aceptación: el dashboard cumple ε=1.0 total (verificable sumando los ε_i), el modelo DP-SGD entrena sin error y la pérdida de accuracy vs no-DP es < 10 pp.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
"Privatizo la salida pero el atacante reco	Olvidaste clippear la entrada — un outlier
Calculo Δf de una media como B (no B/n)	Confundís sensibilidad de suma vs media. F
Hago 100 queries con ε=1 y digo "es ε=1-DP	Sin tracking, gastaste ε=100 por composici
ε=10 o ε=20 "porque así da mejor utility"	ε≥10 da garantía prácticamente nula (e^10
DP-SGD sin clippear per-sample	Sin clipping, la sensibilidad del gradient
Reutilizar el dataset privado para "valida	El proceso de validación también gasta bud

Preguntas frecuentes

¿Qué ε es "seguro"?

No hay un número universal. La US Census 2020 usó ε ≈ 19.6 (TopDown algorithm, criticado por flojo). Apple iOS reporta ε por feature (típicamente 2-8 por día). Recomendación práctica: empezar en ε=1, justificar cualquier valor mayor. ε≥10 es difícil de defender ante un comité de ética.

¿DP me protege de TODO ataque?

Te protege contra membership inference y reconstruction bajo el modelo de atacante con conocimiento auxiliar arbitrario. NO te protege contra: ataques al modelo no-DP entrenado paralelamente, side-channels (timing), o si el atacante tiene el dato original (no es encriptación).

¿Local DP vs Central DP?

Central DP: confiás en el curador (el servidor agrega ruido). Más utilidad. Local DP: cada usuario agrega ruido antes de mandar (Apple, RAPPOR de Google). Menos utilidad, pero no confiás en nadie. La elección depende del modelo de amenaza.

¿Vale la pena en deep learning?

Sí, con caveats. DP-SGD penaliza accuracy (5-15 pp típicos), y necesita batches grandes para que el ruido se promedie. Opacus / TF-Privacy automatizan todo. Es obligatorio si vas a publicar el modelo o usar datos médicos/financieros bajo regulación.

¿Y federated learning?

Federated learning (Clase 226) por sí solo NO es DP — el server ve gradientes que filtran información. Se combina con secure aggregation + DP para garantías reales (lo que hace Google Gboard).

Referencias

- Dwork, McSherry, Nissim, Smith. Calibrating Noise to Sensitivity in Private Data Analysis (TCC 2006) — el paper original que define DP.
- Dwork, C., Roth, A. The Algorithmic Foundations of Differential Privacy (Foundations and Trends, 2014) — el libro de referencia.

- Abadi et al. Deep Learning with Differential Privacy (CCS 2016) — DP-SGD + moments accountant.
- Opacus — DP-SGD en PyTorch (Meta).
- TensorFlow Privacy — DP-SGD en TF/Keras (Google).
- IBM diffprivlib — mecanismos básicos sklearn-compatible.

Material descargable

- Guía explicativa (PDF) — versión imprimible con todo el contenido de la clase.
- Presentación (PPTX) — deck PowerPoint listo para proyectar en clase.
- Notebook ejecutable (.ipynb) — abrilo desde el laboratorio del programa o desde Jupyter.

Clase 226 — Clase 226 — Federated learning: intro

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data (AISTATS 2017) + Kairouz et al. Advances and Open Problems in Federated Learning (FnTML 2021). Duración estimada: 75 min.*

Objetivo

Entrenar un modelo central sin centralizar los datos. Cada cliente (móvil, hospital, banco) entrena local sobre su data, sube solo pesos o gradientes al servidor, que agrega vía FedAvg. Implementar FedAvg manual sobre regresión logística, ver cómo degrada con datos non-IID, y demostrar el ataque básico de gradient leakage (los gradientes filtran data).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar el setup FL: server + K clientes, rondas, partial participation, comunicación de pesos en vez de data.
- Implementar el algoritmo FedAvg (McMahan 2017): w_{t+1} = Σ_k (n_k / n) * w_k^t.
- Distinguir cross-device (millones de móviles, intermitentes) vs cross-silo (decenas de hospitales, estables).
- Reconocer el efecto de datos non-IID: FedAvg degrada cuando cada cliente ve una distribución distinta.
- Identificar los riesgos: model poisoning, gradient leakage (Zhu 2019), y las defensas (secure aggregation, DP-FedAvg, Krum).

Temas

#	Tema	Por qué importa
1	Setup FL: server + clientes + rondas	La data nunca sale del cliente; solo viaja
2	FedAvg: muestreo, local epochs, agregación	Es el baseline; todo lo demás se compara c
3	Cross-device vs cross-silo	Define el régimen: K=10⁶ intermitentes vs
4	Heterogeneidad: non-IID + system + partial	El paper asume IID; la realidad no lo es.
5	Ataques: model poisoning, gradient leakage	Compartir gradientes ≠ proteger data (Zhu
6	Defensas: secure aggregation, DP, Krum/Mec	Lo que se usa en producción real (Google G

Definiciones y características

- Federated learning (FL): paradigma de entrenamiento donde K clientes con data local entrenan un modelo conjunto coordinados por un servidor, sin compartir data cruda.

- FedAvg: el algoritmo base (McMahan 2017). Cada ronda: server envía pesos w_t; un subset de clientes corre E épocas de SGD local; server promedia: w_{t+1} = Σ (n_k / n) * w_k.
- Cross-device FL: K = millones de teléfonos, clientes intermitentes (sin conexión, batería baja), participación parcial obligatoria. Ejemplo: Gboard de Google.
- Cross-silo FL: K = decenas de organizaciones (hospitales, bancos), clientes estables y siempre online. Ejemplo: consorcios médicos contra cáncer.
- Non-IID data: la distribución P_k(x, y) cambia por cliente. Caso típico: cada hospital ve una mezcla distinta de patologías. FedAvg degrada y oscila.
- Gradient leakage: dado un gradiente compartido (especialmente con batch chico), un atacante puede reconstruir el input original vía optimización inversa (Zhu, Liu, Han, Deep Leakage from Gradients, NeurIPS 2019).
- Secure aggregation (Bonawitz 2017): protocolo criptográfico donde el server solo ve la suma de pesos de N clientes, nunca los individuales — anula el gradient leakage contra el server.
- DP-FedAvg: agregar ruido gaussiano calibrado a los pesos antes de subirlos → garantías formales de differential privacy (Clase 225) sobre la presencia de un cliente.

Dataset / recursos

- Dataset: sintético tabular (clasificación binaria, 2000 muestras × 10 features), split en K=10 clientes. Self-contained, sin descargas.
- Librerías: numpy, scikit-learn. Implementamos FedAvg manualmente — sin flower, PySyft ni TensorFlow Federated, para que el algoritmo quede claro.

Ejercicios

1. Particionado IID: generar 2000 muestras, split aleatorio uniforme en K=10 clientes. Verificar que cada cliente tiene ~200 muestras con clases balanceadas.
2. Local training: implementar local_train(X, y, w, epochs=5, lr=0.05) — regresión logística con SGD manual. Devuelve nuevos pesos w_k.
3. FedAvg loop: 20 rondas. En cada ronda, samplear 5 de 10 clientes; entrenar local; agregar vía fedavg(weights, sizes). Trackear loss global.
4. Centralizado vs federado: entrenar la misma logística sobre todo el data junto. Verificar que FL converge a una accuracy comparable (gap < 0.03 en setting IID).
5. Non-IID: re-partir asignando a cada cliente solo 1-2 clases (cliente 0 ve mayoritariamente clase 0, etc.). Correr FedAvg. Mostrar que la accuracy global degrada y oscila más.

Homework verificable

Notebook con:

1. Implementar FedAvg + variante FedProx (agrega regularización μ/2 · ||w_k - w_t||² al loss local — Li et al. 2020) sobre el mismo dataset non-IID.
2. Comparar curvas de loss FedAvg vs FedProx vs central (3 curvas).
3. Implementar DP-FedAvg: ruido gaussiano σ=0.01 sobre los pesos agregados. Medir pérdida de accuracy.
4. Reproducir gradient leakage simple sobre 1 muestra: dado el gradiente de regresión lineal con batch=1, recuperar x vía minimización de ||_w L(w; x_hat, y) - g||².
5. Discutir (≤200 palabras): cuándo FL paga el costo de comunicación vs entrenamiento central + acuerdo de data sharing.

Criterio de aceptación: FedProx supera a FedAvg en setting non-IID (loss final menor); DP-FedAvg con σ=0.01 pierde <0.05 de accuracy; reconstrucción de x con MSE < 0.01 respecto al original.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
FedAvg diverge en non-IID	Demasiadas épocas locales → client drift.
Loss global oscila ronda a ronda	Sampleo de pocos clientes por ronda. Fix:
Cliente con n_k=0 rompe la agregación	División por cero en pesos. Fix: filtrar c
Promedio simple en vez de weighted	Clientes con poca data pesan igual que los
"FL = privado por default"	Falso. Los pesos filtran data (Zhu 2019).
Pesos cargados como listas python en el se	Costo de comunicación se dispara. Fix: env

Preguntas frecuentes

¿FL es differential privacy?

No, son ortogonales. FL no centraliza data, pero los pesos compartidos pueden filtrar muestras individuales. DP da garantías formales (Clase 225). Producción real usa FL + secure aggregation + DP combinados.

¿Cuándo elegir FL en vez de entrenar centralizado?

Cuando la data no puede moverse: regulación (HIPAA, GDPR), data en edge (teclados, wearables), competidores que quieren modelo conjunto sin compartir clientes. Si podés centralizar, centralizá — es 10-100× más barato en comunicación y converge mejor.

¿Cuánto cuesta la comunicación?

Es el cuello de botella real. Cada ronda envía O(|w|) bytes por cliente — para una red de 10M params en float32 son 40 MB por cliente, por ronda. Por eso existen técnicas de compresión, quantization y top-k sparsification.

¿FedAvg está obsoleto?

Sigue siendo el baseline. Variantes modernas: FedProx (regularización local, 2020), SCAFFOLD (control variates para non-IID, 2020), FedOpt (Adam/Yogi del lado server, 2021). Todas se comparan contra FedAvg.

¿Qué framework usar en producción?

Flower es agnóstico al ML framework y el más adoptado en investigación reciente. TensorFlow Federated si ya estás en TF. PySyft si querés combinar con SMPC/HE. Para móvil: TFF Lite y la stack interna de Google (no abierta).

Referencias

- McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data (AISTATS 2017) — el paper original de FedAvg: <https://arxiv.org/abs/1602.05629>.
- Kairouz, P. et al. Advances and Open Problems in Federated Learning (FnTML 2021) — survey canónico, 50+ autores: <https://arxiv.org/abs/1912.04977>.
- Zhu, L., Liu, Z., Han, S. Deep Leakage from Gradients (NeurIPS 2019) — el ataque de reconstrucción desde gradientes: <https://arxiv.org/abs/1906.08935>.
- Bonawitz, K. et al. Practical Secure Aggregation for Privacy-Preserving Machine Learning (CCS 2017) — el protocolo de Google.
- Flower framework — librería FL agnóstica al framework ML.
- Li, T., Sahu, A., Talwalkar, A., Smith, V. Federated Learning: Challenges, Methods, and Future Directions (IEEE Signal Processing, 2020).

Material descargable

- Guía explicativa (PDF) — versión imprimible con todo el contenido de la clase.
- Presentación (PPTX) — deck PowerPoint listo para proyectar en clase.
- Notebook ejecutable (.ipynb) — abrilo desde el laboratorio del programa o desde Jupyter.

Clase 227 — Clase 227 — GDPR y AI Act (EU)

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: Reglamento UE 2016/679 (GDPR) + Reglamento UE 2024/1689 (AI Act). Duración estimada: 75 min.*

Objetivo

Entender qué exige la regulación europea a un sistema de ML que toca datos personales o decisiones automatizadas. GDPR (en vigor desde 25-may-2018) regula el dato: bases legales, derechos del titular, DPIA. AI Act (Reglamento UE 2024/1689, escalonado 2024-2027) regula el sistema de IA por nivel de riesgo: prohibido / alto / limitado / mínimo. Aterrizamos ambas normas en un mini-toolkit programático que un equipo de datos puede ejecutar antes de poner un modelo en producción.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Identificar la base legal del Art. 6 GDPR aplicable a un tratamiento (consentimiento, contrato, interés legítimo, etc.) y distinguir las categorías especiales del Art. 9 (salud, biometría, raza).
- Implementar los derechos del titular más comunes: acceso, rectificación, supresión (Art. 17 — derecho al olvido) y portabilidad.
- Reconocer cuándo el Art. 22 GDPR (decisiones automatizadas con efectos significativos) exige supervisión humana y combinarlo con el AI Act 14 del AI Act.
- Clasificar un caso de uso de IA en el nivel de riesgo del AI Act (prohibido, alto — Anexo III, limitado, mínimo) y listar las obligaciones que aplican.
- Generar una model card mínima y un checklist DPIA programático como artefactos de compliance.

Temas

#	Tema	Por qué importa
1	Bases legales (Art. 6) y categorías especi	Sin base legal válida, el tratamiento es i
2	Derechos del titular (acceso, supresión, p	El "derecho al olvido" obliga a poder borr
3	DPIA (Art. 35) — Data Protection Impact As	Obligatoria para alto riesgo (perfilado ma
4	AI Act: pirámide de riesgo	Prohibido → alto → limitado → mínimo. Defi
5	Sistemas de alto riesgo (Anexo III)	CV-screening, crédito, biometría, educació
6	GPAI y modelos fundacionales	Transparencia + resumen de datos de entren

Definiciones y características

- Dato personal (GDPR Art. 4.1): toda información sobre una persona física identificada o identificable — incluye IP, cookie ID, hashes débiles.
- Categorías especiales (Art. 9): origen racial, opiniones políticas, religión, datos genéticos, biométricos para identificar, salud, orientación sexual. Requieren base reforzada (consentimiento explícito, interés público en salud, etc.).

- Base legal Art. 6: una de seis: (a) consentimiento, (b) contrato, (c) obligación legal, (d) interés vital, (e) interés público, (f) interés legítimo.
- DPIA (Art. 35): análisis previo de riesgos para los derechos del titular; obligatorio en perfilado sistemático a gran escala, categorías especiales, vigilancia pública.
- Decisión automatizada significativa (Art. 22): el titular tiene derecho a no ser sujeto a una decisión basada solo en tratamiento automatizado con efectos jurídicos o significativos — se levanta con consentimiento explícito, ejecución contractual o ley, siempre con derecho a intervención humana.
- Sistema de IA de alto riesgo (AI Act Anexo III): biometría, infraestructura crítica, educación, empleo, servicios esenciales (crédito, seguros), aplicación de la ley, migración, justicia. Obligaciones: gestión de riesgos, gobernanza de datos, documentación técnica, registros (logs), transparencia al usuario, supervisión humana (Art. 14), robustez, conformidad CE.
- GPAI (General Purpose AI): modelos fundacionales reutilizables. Transparencia + resumen público de datos de entrenamiento; si superan 10^{25} FLOPs entran en "systemic risk" con red-teaming y reporte de incidentes.
- Multas: GDPR hasta 20 M€ o 4% revenue global; AI Act hasta 35 M€ o 7% (prácticas prohibidas).

Dataset / recursos

- Dataset sintético de decisiones de crédito (1000 filas) con campos personales (email, dni, age, salary, score, is_minority) generado en el notebook con np.random.default_rng(42) — sin datos reales.
- Librerías: numpy, pandas, scikit-learn, re (regex para PII).

Ejercicios

1. Clasificador de riesgo AI Act: is_high_risk_use_case("CV screening") → "alto"; "juego móvil de match-3" → "mínimo". Cubrir los 4 niveles con lookup table basada en Anexo III.
2. DPIA checklist: dado {"sensitive_categories": True, "automated_decisions": True, "scale": "large"}, listar las obligaciones GDPR aplicables (DPIA, DPO, consentimiento reforzado, etc.).
3. Right to be forgotten: implementar right_to_be_forgotten(df, user_id) que elimine al usuario y devuelva un registro de auditoría con timestamp + columnas afectadas.
4. Data minimization audit: detectar columnas que parecen email (r"[\w\.-]+@[\w\.-]+") o DNI (r"\d{8}[A-Z]"); sugerir hash o remoción.
5. Compliance report: pipeline de scoring de crédito que ejecuta los 7 chequeos del notebook (clasificación de riesgo, DPIA, model card, supervisión humana, minimización, auditoría de PII, registro de borrado) e imprime un reporte único.

Homework verificable

Notebook con:

1. Tomar un caso de uso real propio o público (ej.: recomendador de empleo, scoring de seguros) y clasificarlo en el AI Act con la función del ejercicio 1.
2. Producir una model card completa (intended use, training data summary, performance global y por grupo sensible, limitaciones, fecha, owner).
3. Ejecutar el DPIA checklist y listar las obligaciones GDPR aplicables al caso.
4. Implementar el right to be forgotten sobre un dataset de >10K filas y verificar que tras la deleción el usuario no aparece en ninguna columna (incluido model.predict).
5. Generar un compliance report Markdown con todas las secciones (riesgo AI Act, DPIA, model card, PII audit, human-in-the-loop).

Criterio de aceptación: el reporte ejecuta sin errores, identifica correctamente el nivel de riesgo del caso, lista al menos 5 obligaciones GDPR justificadas, y demuestra el borrado de un usuario con auditoría.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
"Tengo consentimiento, ya puedo hacer todo	Consentimiento debe ser libre, específico,
Borrar al usuario de la tabla users pero s	El derecho al olvido aplica a todo el ecos
"Es solo un prototipo, GDPR no aplica"	GDPR aplica desde el primer dato personal
Clasificar un CV-screener como "riesgo lim	Anexo III lo lista explícitamente como alt
"Anonimicé los datos" pero quedan IPs y us	Pseudoanonimización ≠ anonimización. Si se
Modelo en producción sin human override pa	Art. 22 GDPR + Art. 14 AI Act exigen super

Preguntas frecuentes

Si mi empresa está fuera de la UE, ¿GDPR/AI Act aplican?

Sí, si tratás datos de personas en la UE (GDPR Art. 3, alcance extraterritorial) o si ofrecés un sistema de IA cuyo output se usa en la UE (AI Act Art. 2). Es el mismo principio del CCPA / LGPD.

¿GDPR me obliga a explicar mi modelo?

Hay debate. Wachter, Mittelstadt & Floridi (2017) argumentan que el Art. 22 + Recitales 71 garantizan información significativa sobre la lógica, no una explicación post-hoc tipo SHAP. En la práctica: documentación de la lógica + derecho a intervención humana + posibilidad de contestar la decisión.

¿Qué hago si entreno un LLM con datos scrapeados de internet?

Riesgo legal real (caso Clearview AI multado por la AEPD y otras DPAs). El AI Act prohíbe el scraping indiscriminado de imágenes faciales (Art. 5). Para texto, depende de base legal e intereses; minimizá, documentá fuentes, ofrecé opt-out.

¿Cuándo debo registrar un DPO (Data Protection Officer)?

Obligatorio si: autoridad pública, tratamiento a gran escala de categorías especiales, o monitoreo sistemático a gran escala (Art. 37). En la práctica, casi cualquier producto de ML con datos personales a escala lo requiere.

¿Cuándo entran en vigor las obligaciones del AI Act?

Escalonado: prohibiciones desde feb-2025, GPAI desde ago-2025, alto riesgo desde ago-2026/2027 según anexo. Sanciones empiezan a aplicarse con cada hito.

Referencias

- Reglamento UE 2016/679 (GDPR) — texto consolidado en EUR-Lex.
- Reglamento UE 2024/1689 (AI Act) — texto consolidado en EUR-Lex.
- European Data Protection Board — guidelines y decisiones.
- Council of Europe Framework Convention on AI (2024) — primer tratado internacional vinculante sobre IA.
- Wachter, S., Mittelstadt, B., Floridi, L. Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation (IDPL, 2017). <https://academic.oup.com/idpl/article/7/2/76/3860948>.
- Mitchell, M. et al. Model Cards for Model Reporting (FAT* 2019) — base de la model card que implementamos.

Material descargable

- Guía explicativa (PDF) — versión imprimible con todo el contenido de la clase.
- Presentación (PPTX) — deck PowerPoint listo para proyectar en clase.
- Notebook ejecutable (.ipynb) — abrilo desde el laboratorio del programa o desde Jupyter.

Clase 228 — Clase 228 — Reproducibilidad: seeds, lock files, versionado de datasets

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: Pineau et al., Improving Reproducibility in ML Research (JMLR 2021) + Gebru et al., Datasheets for Datasets (CACM 2021) + Mitchell et al., Model Cards for Model Reporting (FAT* 2019). Duración estimada: 75 min.*

Objetivo

Cerrar la Parte 7 con el problema que atraviesa todo lo anterior: si un experimento no es reproducible, no es auditable, no es comparable y no es ciencia. Aprender a controlar las tres fuentes de no-determinismo (código, datos, ambiente) con seeds, lock files, hashes de datasets, model cards y manifiestos de pipeline. Entender por qué Hutson (Science, 2018) habló de "crisis de reproducibilidad" en ML y qué piden hoy NeurIPS/JMLR como mínimo.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Sembrar correctamente random, numpy, sklearn (y comentar torch) con una función seed_everything() y PYTHONHASHSEED.
- Distinguir requirements.txt (top-level) de un lock file (uv.lock, poetry.lock, conda-lock) que pinea toda la transitive tree.
- Calcular un hash estable de un DataFrame (sha256_of_df) que sobreviva a reorden de columnas e índice y sirva como dataset_id.
- Producir un manifest JSON con {data_hash, code_hash, seed, package_versions} y validar reproducibilidad antes de re-entrenar.
- Redactar una model card mínima (intended use, training data hash, metrics, limitations) según Mitchell et al. 2019.

Temas

#	Tema	Por qué importa
1	Fuentes de no-determinismo	GPU atomicAdd, dict/set ordering, num_work
2	Seeds en stack Python	random.seed, np.random.default_rng, torch.
3	Lock files vs requirements	uv.lock/poetry.lock pinean toda la transit
4	Ambiente reproducible	Docker + base image pineada por digest (py
5	Versionado de datasets	DVC, Git LFS, lakeFS, S3 versioning, hash
6	Documentación: datasheets + model cards	Gebru 2018 (dataset) y Mitchell 2019 (mode

Definiciones y características

- Determinismo: misma entrada + mismo código + mismo ambiente → mismo output bit-a-bit. En ML rara vez se logra al 100% sobre GPU; se aspira a reproducibilidad estadística.
- Seed: entero que inicializa el estado de un PRNG. Cambiar el seed cambia la trayectoria — reportar un solo seed sin promediar varios es una mala práctica (Pineau 2021).
- PYTHONHASHSEED: variable de entorno que controla el hash de strings/objetos. Desde Python 3.3,

por default es aleatorio por proceso — afecta orden de iteración de set/dict con keys hashables custom.
- Float associativity: (a + b) + c ≠ a + (b + c) en float32/64 por redondeo. Sumar en otro orden (BLAS multi-threaded, GPU reductions) da resultados que difieren en 1 ULP — suficiente para divergir tras muchas iteraciones.
- Lock file: archivo que pinea la versión exacta de cada dependencia transitiva con hash del wheel. requirements.txt con pandas==2.2.0 no pinea numpy, pytz, etc. — uv.lock sí.
- DVC (Data Version Control): git para datos. Guarda un .dvc pointer en git (hash MD5 + tamaño) y el blob real en storage remoto (S3/GCS). dvc pull re-hidrata.
- Datasheet for Datasets (Gebru et al. 2018): documento que acompaña al dataset con motivación, composición, recolección, uses, distribución. Equivalente al spec sheet de un componente electrónico.
- Model Card (Mitchell et al. 2019): documento que acompaña a un modelo con intended use, factores, métricas desagregadas, training/eval data, ethical considerations, caveats.

Dataset / recursos

- Dataset sintético generado in-notebook (no externo) — la clase es sobre el proceso, no sobre el dato.
- Librerías: stdlib (hashlib, json, importlib.metadata, os, random), numpy, pandas, scikit-learn.
- Opcional para homework: DVC, uv.

Ejercicios

1. Seed everything: implementar seed_everything(seed=42) que cubra random, numpy y PYTHONHASHSEED. Verificar que np.random.rand(5) da el mismo vector en dos corridas consecutivas.
2. Hash de DataFrame: escribir sha256_of_df(df) que ordene columnas alfabéticamente, resetee el índice y serialice a CSV bytes antes de hashear. Mostrar que dos df con columnas en distinto orden dan el mismo hash.
3. Manifest de experimento: dict con {data_hash, code_hash, seed, sklearn_version, numpy_version, pandas_version, python_version} serializado a experiment.json.
4. Validación de reproducibilidad: dado un manifest guardado, re-leer el dataset, recomputar su hash, comparar — abortar con RuntimeError si difiere.
5. Model card mínima: función build_model_card(model, X, y, intended_use, limitations, date_trained) que devuelve dict con campos de Mitchell et al. y lo dumpea a JSON.

Homework verificable

Notebook (o script) que:

1. Construya un pipeline load → train → evaluate con seed_everything(42) al inicio.
2. Calcule dataset_hash con sha256_of_df y code_hash con sha256 sobre el .py (o el source string vía inspect.getsource).
3. Guarde manifest.json con data_hash, code_hash, seed, package_versions y model_card.json con métricas + limitations.
4. Re-corra el mismo pipeline en otra ejecución; verifique que accuracy reportada coincide bit-a-bit y que ambos hashes coinciden.
5. Cree requirements.lock con uv pip compile requirements.in -o requirements.lock (o pip freeze > requirements.lock como fallback) y commitee ambos al repo.

Criterio de aceptación: en dos máquinas distintas (o dos venvs limpios) con uv pip sync requirements.lock, el pipeline produce el mismo accuracy y el mismo dataset_hash. El manifest.json está commiteado.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Resultados cambian entre corridas a pesar	Hay otra fuente: random stdlib sin seedear
Hash del DataFrame cambia y los datos "son	Estás hasheando df.to_csv() directo — incl
requirements.txt con pandas==2.2.0 rompe e	No pinea numpy (transitive). Fix: usar uv.
Torch da resultados distintos en GPU con s	atomicAdd en GPU no es determinista. Fix p
Iteración sobre set da orden distinto cada	PYTHONHASHSEED aleatorio. Fix: setearla an
sum(arr) ≠ np.sum(arr) con float32	Float associativity + orden de reducción.

Preguntas frecuentes

¿Es realmente posible reproducir bit-a-bit un entrenamiento en GPU?

En general no sin esfuerzo. Hay ops no deterministas (atomicAdd, cuDNN convolutions con autotuner). Con torch.use_deterministic_algorithms(True) + CUBLAS_WORKSPACE_CONFIG + num_workers=0 + cudnn.deterministic=True te acercás, a costa de 1.5-3× más slow. En CPU es mucho más fácil.

¿requirements.txt no alcanza si pineo todo con ==?

No del todo. Aunque pinees pandas==2.2.0, pip puede resolver numpy a una versión distinta entre máquinas si el resolver tiene libertad. Un lock file congela el resultado del resolver con hashes de wheels. uv.lock, poetry.lock, Pipfile.lock, conda-lock son equivalentes funcionales.

¿DVC, Git LFS o S3 versioning?

- DVC: si tu workflow es git-céntrico y querés dvc repro (pipelines con cache por inputs). Storage backend a elección.
- Git LFS: si los archivos son binarios chicos (<2GB) y querés transparencia total con git. No tiene pipelines.
- S3 versioning: si ya vivís en AWS y solo necesitás "ver versiones anteriores del objeto". Sin metadata de pipeline.
- lakeFS / Quilt / Pachyderm: para escala enterprise con branching real sobre data lakes.

¿Hashear el .csv original o el DataFrame cargado?

El DataFrame normalizado (columnas ordenadas, sin índice). El .csv puede tener BOM, line endings distintos (\r\n vs \n), encoding distinto y romper el hash sin que el contenido cambie. Hashear post-parsing es semánticamente correcto.

¿Una model card es un trámite burocrático?

No si la usás como gate de release. Auditorías regulatorias (EU AI Act 2024, NIST AI RMF) ya exigen documentación equivalente. Una model card bien hecha es además la entrada del próximo experimento: "el baseline al que tengo que ganarle es éste, entrenado con estos datos, evaluado así".

Referencias

- Pineau, J. et al. Improving Reproducibility in Machine Learning Research (JMLR 2021) — el NeurIPS Reproducibility Checklist.
- Gebru, T. et al. Datasheets for Datasets (CACM 2021).
- Mitchell, M. et al. Model Cards for Model Reporting (FAT* 2019).
- Hutson, M. Artificial intelligence faces reproducibility crisis (Science, 2018).
- PyTorch Reproducibility notes — qué controla cada flag.
- DVC docs · uv (lock files modernos).

Material descargable

- Guía explicativa (PDF) — versión imprimible con todo el contenido de la clase.
- Presentación (PPTX) — deck PowerPoint listo para proyectar en clase.
- Notebook ejecutable (.ipynb) — abrilo desde el laboratorio del programa o desde Jupyter.

Cierre de la parte

Fin del bundle consolidado de Parte 7 — Ética, Fairness y Privacidad · 6 clases.