Clase 063 — Métricas: confusion matrix, precision, recall, F1

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 3. Duración estimada: 70 min.

Clase 063 — Métricas: confusion matrix, precision, recall, F1

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 3. Duración estimada: 70 min.

Objetivo

Que el alumno deje de mirar accuracy como métrica única y aprenda a leer una confusion matrix, a elegir entre precision, recall, F1 o F-beta según el costo de los errores, y a interpretar classification_report clase por clase. En particular, entender por qué en problemas desbalanceados (fraude, churn, diagnóstico) accuracy miente y qué hacer al respecto.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Construir e interpretar una confusion matrix con sklearn.metrics.confusion_matrix identificando TP, FP, TN, FN.
2. Calcular y comparar precision, recall, F1 y F-beta a mano y con precision_score, recall_score, f1_score, fbeta_score.
3. Elegir la métrica adecuada según el costo asimétrico de los errores (FP vs FN) del problema.
4. Leer classification_report y diferenciar macro avg vs weighted avg en multiclase.
5. Diagnosticar class imbalance y aplicar class_weight='balanced', SMOTE o threshold tuning según corresponda.

Temas

#	Tema	Por qué importa
1	Confusion matrix (TP/FP/TN/FN)	Base de toda métrica de clasificación.
2	Precision = TP / (TP+FP)	Cuán "puras" son las predicciones positiva
3	Recall = TP / (TP+FN)	Qué fracción de positivos reales capturo.
4	F1 y F-beta	Media armónica; F-beta pondera recall (β>1
5	classification_report y macro/weighted avg	Métricas por clase en multiclase.
6	Accuracy y por qué falla con clases desbal	Paradoja del 99% inútil.
7	Class imbalance: class_weight, SMOTE, thre	Cuándo aplicar cada uno.

Versión profundizada — 2026

El tema moderno que antes vivía como complemento dentro de esta clase ahora tiene su(s) clase(s) propia(s) con patrón completo, ejercicios y homework:

- Clase 056a — Class imbalance: SMOTE, ADASYN, class_weight, threshold tuning

Definiciones y características

TP / FP / TN / FN

: True Positive: positivo real predicho positivo. False Positive: negativo real predicho positivo (error tipo I). True Negative: negativo real predicho negativo. False Negative: positivo real predicho negativo (error tipo II). Toda métrica binaria se deriva de estos cuatro.

Confusion matrix

: Tabla 2×2 (o k×k en multiclase) con counts de cada combinación (real, predicho). En sklearn las filas son la clase real y las columnas la predicha: [[TN, FP], [FN, TP]].

Precision = TP / (TP + FP)

: De todo lo que predije positivo, ¿qué fracción acerté? Importa cuando un FP es caro (ej.: bloquear una transacción legítima, marcar email legítimo como spam).

Recall (sensitivity, TPR) = TP / (TP + FN)

: De todos los positivos reales, ¿qué fracción detecté? Importa cuando un FN es caro (ej.: no detectar un tumor, dejar pasar un fraude).

F1 = 2 · (P · R) / (P + R)

: Media armónica de precision y recall. Penaliza fuerte cuando una de las dos es baja. Default razonable cuando no hay preferencia clara entre FP y FN.

F-beta = (1 + β²) · (P · R) / (β² · P + R)

: F1 generalizada. β > 1 pondera más recall (β=2 → recall pesa 4× más que precision). β < 1 pondera más precision (β=0.5). Útil cuando el costo de FP y FN es asimétrico pero ambos importan.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

: Fracción total de aciertos. Engañosa con clases desbalanceadas: con 99% negativos, predecir "todo negativo" da 99% accuracy con cero recall. Reservala para problemas balanceados.

class_weight

: Hiperparámetro de la mayoría de estimadores sklearn. 'balanced' calcula pesos inversamente proporcionales a la frecuencia. También acepta dict {0: 1, 1: 10} para control manual. Modifica la función de pérdida, no el dataset.

Dataset / recursos

- MNIST 5-vs-not-5 (binarizado): clásico de Géron cap. 3. Imbalance ~10% positivo.
- Credit card fraud (Kaggle, opcional): ~0.17% positivo, ideal para ver SMOTE en acción.
- Sintético con make_classification(weights=[0.99, 0.01]) para experimentar sin descargar.

Ejercicios

1. Confusion matrix a mano. Dado y_true = [0,1,1,0,1,1,0,0,1,0] y y_pred = [0,1,0,0,1,1,1,0,1,0]: calculá TP/FP/TN/FN, precision, recall y F1 con lápiz y papel. Verificá con sklearn.metrics.

2. MNIST 5-detector. Entrená SGDClassifier sobre MNIST binarizado (5 vs no-5). Mostrá confusion matrix con ConfusionMatrixDisplay y reportá precision, recall y F1.

3. classification_report multiclase. Sobre MNIST completo (10 clases) con LogisticRegression, imprimí classification_report. Identificá qué dígito tiene peor recall y por qué (mirá la confusion matrix).

4. Class imbalance con class_weight. Generá un dataset con make_classification(n_samples=10000, weights=[0.99, 0.01], random_state=42). Entrená dos LogisticRegression: una sin class_weight y otra con

class_weight='balanced'. Compará recall de la clase minoritaria.

5. SMOTE dentro de Pipeline. Mismo dataset que el ejercicio 4. Armá un imblearn.pipeline.Pipeline con SMOTE + LogisticRegression, evaluá con cross_val_score(scoring='f1') y compará contra class_weight='balanced'. Verificá que SMOTE solo se aplica al train fold (leelo en docs).

Homework verificable

Notebook sobre el dataset sintético desbalanceado (~1% positivo) que: (a) entrene baseline LogisticRegression y reporte confusion matrix + classification_report; (b) repita con class_weight='balanced'; (c) arme Pipeline con SMOTE; (d) haga threshold tuning con precision_recall_curve sobre val set maximizando F1; (e) tabla comparativa de F1 y recall de la clase minoritaria para las 4 estrategias (baseline, class_weight, SMOTE, threshold tuning).

Criterio de aceptación: las 4 estrategias documentadas, F1 de la clase minoritaria del mejor enfoque ≥ 2× el del baseline, y SMOTE aplicado solo dentro de Pipeline (chequear en código que no aparece SMOTE().fit_resample(X, y) sobre el dataset completo antes del split).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Accuracy 99% pero el modelo "no detecta na	Clases desbalanceadas + métrica equivocada
Aplicar SMOTE().fit_resample(X, y) antes d	Data leakage: vecinos sintéticos se constr
precision_score lanza UndefinedMetricWarni	No hubo predicciones positivas (TP+FP=0).
Confunden ejes de la confusion matrix	En sklearn las filas son la clase real y l
Usar F1 cuando el costo de FP y FN es muy	F1 trata ambos errores como equivalentes.

Preguntas frecuentes

¿SMOTE, class_weight o threshold tuning?

Empezá siempre por class_weight='balanced' — es gratis, no toca el dataset, no genera leakage. Si no alcanza, sumá threshold tuning (también gratis, solo cambia el corte sobre predict_proba). Recurrí a SMOTE/ADASYN cuando los dos anteriores no rinden y sospechás que el modelo no separa bien la frontera de decisión. Ojo: SMOTE no es magia — en datasets tabulares ruidosos a veces empeora.

¿Precision o recall?

Depende del costo asimétrico: si un FN es caro (no detectar un cáncer, dejar pasar un fraude) → priorizá recall. Si un FP es caro (bloquear cliente legítimo, marcar email serio como spam) → priorizá precision. Cuando no hay preferencia clara, usá F1.

¿macro avg o weighted avg en multiclase?

macro avg = promedio simple por clase (todas pesan igual). Útil cuando te importan todas las clases por igual, incluyendo las raras. weighted avg = promedio ponderado por soporte (clases mayoritarias pesan más). Útil cuando refleja el costo real del negocio. En problemas desbalanceados macro avg es más honesto.

¿Por qué F1 es media armónica y no aritmética?

Porque la armónica penaliza fuerte los valores bajos. Si precision=1.0 y recall=0.0, la media aritmética da 0.5

(engañosa); la armónica da 0. F1 te obliga a que ambas sean razonables.

¿El umbral 0.5 es óptimo?

Casi nunca — es el default de predict() pero no tiene base teórica. Con precision_recall_curve sobre un set de validación encontrás el umbral que maximiza F1 (o F-beta, o la métrica de negocio). En problemas desbalanceados moverlo suele dar más mejora que cambiar de modelo.

Referencias

- Géron, cap. 3 — Performance Measures, Confusion Matrix, Precision and Recall, The Precision/Recall Trade-off.
- scikit-learn — Classification metrics
- imbalanced-learn — User guide
- imbalanced-learn — SMOTE
- Chawla et al. (2002), SMOTE: Synthetic Minority Over-sampling Technique, JAIR 16.

Siguiente clase

Clase 064 — Class imbalance: SMOTE, ADASYN, class_weight, threshold tuning

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb