Clase 062 — Clasificación binaria con MNIST

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 3. Duración estimada: 60 min.

Clase 062 — Clasificación binaria con MNIST

> *Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 3. Duración estimada: 60 min.*

Objetivo

Que el alumno arme su primer clasificador binario "de verdad" sobre MNIST (¿este dígito es un 5 o no?), entrenando un SGDClassifier, evaluándolo con cross_val_score sobre StratifiedKFold, y entendiendo por qué la accuracy sola miente cuando las clases están desbalanceadas.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Cargar MNIST con fetch_openml('mnist_784', as_frame=False) y separar train/test respetando el split original (60k/10k).
2. Construir un target binario y_train_5 = (y_train == 5) y entrenar un SGDClassifier sobre él.
3. Predecir y obtener scores con predict() y decision_function(), entendiendo la diferencia entre clase y score continuo.
4. Validar con cross_val_score sobre StratifiedKFold y leer los 3 valores que devuelve.
5. Detectar el accuracy paradox: comparar contra un clasificador trivial "nunca-5" y ver que también saca ~90%.

Temas

#	Tema	Por qué importa
1	fetch_openml('mnist_784')	Dataset estándar de entrada a CV/ML; 70k i
2	Target binario 5 vs no-5	Forma canónica de empezar antes de meterse
3	SGDClassifier(random_state=42)	Lineal, escalable, online; el "hola mundo"
4	cross_val_score + StratifiedKFold	Mantiene la proporción de clases en cada f
5	Accuracy paradox	90% suena bien hasta que ves que Never5Cla
6	predict vs decision_function vs predict_pr	Score continuo es lo que después te deja m

Definiciones y características

MNIST

: Dataset de 70.000 imágenes de dígitos manuscritos (0-9), 28×28 píxeles en escala de grises, aplanadas a vectores de 784 features. Split convencional: 60k train + 10k test, ya barajado. Es el "hello world" de ML — chico, limpio, sin sorpresas.

SGDClassifier

: Clasificador lineal entrenado con Stochastic Gradient Descent. Procesa instancias de a una (o mini-batch), lo que lo hace ideal para datasets grandes y aprendizaje online. Sensible al random_state (fijarlo siempre para reproducibilidad).

decision_function(X)

: Devuelve el score crudo (distancia firmada al hiperplano de decisión), no la clase. Score > 0 → clase positiva, score < 0 → negativa. Es lo que vas a necesitar después para mover el umbral y construir curvas PR/ROC.

Accuracy

: (TP + TN) / total. Métrica obvia pero traicionera: si el 90% de tus instancias son de la clase negativa, predecir "siempre negativo" te da 90% sin haber aprendido nada.

Baseline trivial (dummy classifier)

: Modelo que predice siempre la clase mayoritaria (o aleatorio). Sirve de piso: si tu modelo no le gana al dummy, no aprendió. En 5-vs-no-5, el dummy "nunca-5" saca ~90.9% (porque solo ~9.1% de los dígitos son 5).

StratifiedKFold

: Variante de K-Fold que conserva la proporción de clases en cada fold. Imprescindible con desbalanceo: con K-Fold común podrías terminar con un fold sin un solo 5.

cross_val_score(clf, X, y, cv=3, scoring='accuracy')

: Entrena y evalúa clf en cv folds y devuelve un array de cv scores. Por default usa StratifiedKFold cuando la tarea es clasificación.

Dataset / recursos

MNIST vía fetch_openml (se cachea localmente en ~/scikit_learn_data/ después de la primera descarga):

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', as_frame=False, parser='auto')
X, y = mnist.data, mnist.target.astype(int)
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]
y_train_5 = (y_train == 5)
y_test_5  = (y_test == 5)
```

Ejercicios

1. Cargar y explorar. Cargá MNIST, imprimí X.shape y y.shape, mostrá un dígito con matplotlib.imshow(X[0].reshape(28, 28), cmap='binary') y verificá que y[0] == 5.

2. Target binario + SGD. Construí y_train_5, entrená SGDClassifier(random_state=42) y predecí sobre X[0]. ¿Devuelve True?

3. decision_function. Sobre la misma instancia, llamá a decision_function([X[0]]) y compará el signo con el resultado de predict.

4. Cross-validation. Corré cross_val_score(sgd, X_train, y_train_5, cv=3, scoring='accuracy'). ¿Qué tres valores te da? ¿Cuál es el promedio?

5. Baseline trampa. Implementá un Never5Classifier (clase con fit que no hace nada y predict que devuelve np.zeros(len(X), dtype=bool)) y corré el mismo cross_val_score. Comparalo con el SGD: ¿la diferencia es la que esperabas?

Homework verificable

Notebook que: (a) carga MNIST y arma el target 5-vs-no-5; (b) entrena SGDClassifier(random_state=42) sobre el train completo; (c) reporta accuracy 3-fold con cross_val_score; (d) reporta accuracy 3-fold del Never5Classifier; (e) en una celda markdown, explica en 2-3 líneas por qué ambos modelos están "cerca" en accuracy y por qué eso no significa que sean equivalentes.

Criterio de aceptación: Accuracy del SGD ≥ 0.95 en los 3 folds, accuracy del dummy ≈ 0.909 en los 3 folds, y la celda markdown menciona explícitamente el desbalanceo (~9% de 5s) como causa del paradox.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
"¡Mi modelo saca 90% de accuracy, está gen	Trampa clásica con clases desbalanceadas.
fetch_openml tarda eternidades o falla	La primera vez descarga ~55MB y a veces el
y viene como strings ('5', '0', ...) y la	Por default OpenML devuelve labels como st
Resultados distintos cada vez que entrenás	SGDClassifier es estocástico. Fix: SGDClas
decision_function no existe / AttributeErr	Algunos clasificadores (como RandomForest)

Preguntas frecuentes

¿Por qué arrancar con clasificación binaria si MNIST es multiclase?

Pedagógico: binario aísla los conceptos de score, umbral, precision/recall sin el ruido de "¿es 3 o 8?". Multiclase viene en la clase 059 (One-vs-Rest, One-vs-One).

¿SGDClassifier con loss='hinge' (default) es lo mismo que SVM?

Es un SVM lineal entrenado con SGD en vez de con el optimizador cuadrático de SVC. Misma frontera teórica, distinta forma de llegar. Para datasets grandes, SGD le gana en tiempo.

¿Por qué cv=3 y no cv=10?

Géron usa 3 porque MNIST es grande (60k×784) y 10 folds tarda. En datasets chicos, 5-10 es lo habitual. La regla: más folds = menos varianza en la estimación pero más cómputo.

¿predict_proba está disponible en SGDClassifier?

Solo si entrenás con loss='log_loss' (regresión logística) o loss='modified_huber'. Con el default 'hinge' no — usás decision_function.

¿Hace falta escalar las features de MNIST?

Para SGDClassifier sí, conviene (los píxeles 0-255 son grandes y SGD es sensible a escala). Géron en el cap. 3 lo omite para simplificar; en la práctica StandardScaler o dividir por 255 ayuda a la convergencia.

Referencias

- Géron, cap. 3 § "Training a Binary Classifier" y "Measuring Accuracy Using Cross-Validation".
- sklearn SGDClassifier
- sklearn fetch_openml

- sklearn StratifiedKFold

Siguiente clase

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

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb