🔷 CONJUNTO: **TRAIN**

Elemento	Uso en TRAIN
`fit()`	✅ Se usa para entrenar el modelo con los datos. Ajusta pesos y aprende patrones.
`evaluate()`	✅ Opcional. Se usa para ver cómo va aprendiendo el modelo, detectar si está sobreajustando.
`predict()`	❌ No tiene sentido usarlo aquí. Ya conoces las respuestas. No aporta nada.

Objetivo del conjunto TRAIN:

➡️ Enseñar al modelo. Aquí **sí se cambia el modelo**.

🔷 CONJUNTO: **VALIDATION**

Elemento	Uso en VALIDATION
`fit()`	❌ No se usa aquí. No se debe entrenar con estos datos.
`evaluate()`	✅ Se usa para ver si el modelo generaliza bien. Ayuda a elegir hiperparámetros.
`predict()`	✅ Opcional. Sirve para ver *cómo* está prediciendo y analizar errores.

Objetivo del conjunto VALIDATION:

➡️ Ajustar configuración del modelo (hiperparámetros) **sin entrenar con estos datos**.
Sirve para decidir si tu modelo está bien antes de probarlo con el test.

🔷 CONJUNTO: **TEST**

Elemento	Uso en TEST
`fit()`	❌ Prohibido. No se debe entrenar con test.
`evaluate()`	✅ Se usa para obtener el rendimiento final del modelo, con datos nunca vistos.
`predict()`	✅ Se usa para obtener las predicciones reales del modelo en producción o evaluación final.

Objetivo del conjunto TEST:

➡️ Evaluar el rendimiento **definitivo**. No se toca el modelo.

Sirve para presentar resultados, publicar o tomar decisiones.

✅ RESUMEN RÁPIDO (cuadro final)

Conjunto	¿Se entrena (`fit()`)?	¿Se evalúa (`evaluate()`)?	¿Se predice (`predict()`)?	¿Para qué sirve?
Train	✅ Sí	✅ Opcional	❌ No	Para enseñar al modelo
Validation	❌ No	✅ Sí	✅ Opcional	Para ajustar hiperparámetros
Test	❌ No	✅ Sí	✅ Sí	Para evaluar el rendimiento final

🎯 DIFERENCIA GENERAL ENTRE `predict()` Y `evaluate()`

Función	¿Qué hace?	¿Qué devuelve?	¿Para qué sirve?
`predict()`	Usa el modelo ya entrenado para **hacer predicciones** sobre nuevos datos	Las **salidas del modelo** (ej. clases, valores continuos)	Ver qué resultado da el modelo en cada caso
`evaluate()`	Usa el modelo ya entrenado para **medir su rendimiento** en conjunto de datos	Una o varias **métricas** (ej. precisión, pérdida)	Ver cómo de bien funciona el modelo

---

🧠 EN DEEP LEARNING (ej. Keras, TensorFlow, PyTorch)

✅ `predict()`

- Se usa **después de entrenar** (`fit()`).
- No te dice si el modelo ha acertado o no. Solo te dice **qué cree el modelo**.
- Ejemplo: te da `[0.1, 0.7, 0.2]` → eso significa: 10% clase A, 70% clase B, 20% clase C.
- Tú luego comparas con la etiqueta real si quieres saber si acertó.

✅ `evaluate()`

- También se usa **después de entrenar**, pero a diferencia de `predict()`, compara automáticamente las predicciones **con las etiquetas verdaderas**.
- Calcula métricas como **accuracy**, **loss**, **f1-score**, etc.
- Sirve para evaluar en conjunto: no te dice si cada predicción está bien, sino el resultado general.

📦 EJEMPLO RÁPIDO

Supongamos que tienes un modelo que clasifica imágenes de perros y gatos:

python Copiar Editar

```python
# Evaluar
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Accuracy en test: {accuracy}")
```

python Copiar Editar

```python
# Predecir
predicciones = model.predict(x_test)
print(predicciones[0])  # [0.8, 0.2] → el modelo cree que es 80% perro, 20% gato
```

🐯 En resumen:

- Usa `**predict()**` → si quieres ver **qué cree el modelo** para cada caso.
- Usa `**evaluate()**` → si quieres ver **cómo de bien rinde el modelo** en general.
