Curso base

INTELIGENCIA ARTIFICIAL DESDE CERO

LECCIÓN 18
Clasificación básica

Cómo entrenar un modelo que aprende a decidir entre clases sí/no y categorías simples

Lección	18	**Tema**	Clasificación básica
Nombre del participante	_____	**Fecha**	_____

Actividad central

Actividad principal: construcción de un primer clasificador funcional con dos ejemplos guiados: uno binario y otro multicategoría.
Producto esperado: notebook corto y ordenado con dos ejemplos básicos de clasificación, al menos una predicción nueva en cada caso y una interpretación breve sobre el sentido de las etiquetas generadas.

Propósito de la guía. Comprender qué es un problema de clasificación y construir un clasificador sencillo que prediga etiquetas como sí/no o categorías básicas. Al finalizar, el participante podrá reconocer cuándo una salida es una clase, preparar un ejemplo elemental y leer con criterio las predicciones del modelo.

Idea central

La clasificación se utiliza cuando no queremos predecir un número, sino asignar una categoría. En lugar de estimar ventas o temperatura, el modelo debe decidir a qué grupo pertenece un caso: aprobado o no aprobado, compra o no compra, prioridad alta o baja, entre otros escenarios.

En esta lección no buscamos dominar todos los algoritmos. Lo importante es entender la lógica del proceso: tomar variables de entrada, aprender a partir de ejemplos etiquetados y luego usar ese aprendizaje para asignar una clase a casos nuevos.

¿Qué vamos a hacer en esta guía?

- Distinguir clasificación y regresión con ejemplos sencillos.
- Reconocer los conceptos básicos: clase, etiqueta, clasificación binaria y clasificación multicategoría.
- Desarrollar un primer ejemplo guiado de clasificación binaria: *aprobado / no aprobado*.
- Desarrollar un segundo ejemplo guiado de clasificación multicategoría: prioridad de soporte.
- Usar prompts de IA generativa para comprender, revisar e interpretar resultados.

¿Cuándo usamos clasificación?

Usamos clasificación cuando la variable objetivo representa categorías. A veces esas categorías son dos, por ejemplo *fraude / no fraude* o *compra / no compra*. Otras veces son varias, como *prioridad alta, media o baja*, o tipo de cliente según su comportamiento.

Situación	Clase o etiqueta que se quiere predecir
Educación	Aprobado / no aprobado, o nivel de desempeño del estudiante.
Ventas	Compra / no compra, o tipo de cliente según su interés.
Soporte	Prioridad alta, media o baja según el caso reportado.

Vocabulario mínimo para comenzar

Concepto	Interpretación sencilla
Clase	Categoría final que el modelo debe asignar.
Etiqueta	Valor concreto de la clase para cada registro del conjunto de datos.
Clasificación binaria	Problema con dos respuestas posibles, como sí / no.
Clasificación multicategoría	Problema con más de dos clases posibles.
Variable objetivo	Columna que contiene la respuesta final que queremos predecir.

Idea rápida para recordar

Una forma simple de recordarlo es esta: en **regresión** la salida es un número; en **clasificación** la salida es una etiqueta.

Ejemplo guiado 1: aprobado / no aprobado

Vamos a comenzar con un caso muy cercano al contexto educativo. Tenemos datos sencillos de estudiantes: horas de estudio, tareas entregadas y una etiqueta final que indica si aprobaron o no. El objetivo es entrenar un modelo que aprenda esa relación básica y luego prediga la clase para nuevos casos.

Horas de estudio	Tareas entregadas	Aprobado
1	0	0

1	1	0
2	0	0
2	1	0
3	1	0
3	2	1
4	2	1
5	3	1
6	3	1
7	4	1

Paso 1. Observar la tabla antes de programar

Antes de ejecutar código, conviene mirar el patrón general. A simple vista parece que, cuando aumentan las horas de estudio y las tareas entregadas, es más probable que el estudiante apruebe. Esa es justamente la relación que intentará aprender el modelo.

Paso 2. Preparar el notebook e importar herramientas

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Paso 3. Crear la tabla de datos

```
datos = pd.DataFrame({
    "horas_estudio": [1,1,2,2,3,3,4,5,6,7],
    "tareas": [0,1,0,1,1,2,2,3,3,4],
    "aprobado": [0,0,0,0,0,1,1,1,1,1]
})
datos
```

Paso 4. Separar entradas y etiqueta

```
X = datos[["horas_estudio", "tareas"]]
y = datos["aprobado"]
```

Aquí `X` contiene las variables de entrada y `y` contiene la clase que queremos predecir.

Paso 5. Dividir en entrenamiento y prueba

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)
```

Esta separación permite entrenar con una parte de los datos y verificar después si el modelo funciona con ejemplos que no vio durante el aprendizaje.

Paso 6. Entrenar el clasificador

```
modelo = LogisticRegression(max_iter=200)
modelo.fit(X_train, y_train)
```

Paso 7. Hacer predicciones y calcular una métrica simple

```
pred = modelo.predict(X_test)
acc = accuracy_score(y_test, pred)

print("Predicciones:", pred)
print("Accuracy:", round(acc, 2))
```

Paso 8. Interpretar el resultado

En esta primera experiencia no buscamos una métrica perfecta. Lo importante es verificar que el flujo funciona y que las etiquetas predichas tienen sentido. Si el modelo produce *aprobado / no aprobado* de forma coherente con la lógica de los datos, ya hemos dado un paso importante.

Prompts de IA generativa para acompañar el ejemplo 1

- "Explícame con palabras muy sencillas qué hace un clasificador cuando aprende a decidir entre aprobado y no aprobado."
- "Revisa este código de `LogisticRegression` y dime paso a paso si está correcto para un principiante."
- "Ayúdame a redactar una interpretación corta de las predicciones de un clasificador binario."
- "¿Cuál es la diferencia entre predecir una nota numérica y predecir aprobado / no aprobado?"

Ejemplo guiado 2: prioridad de soporte (alta, media, baja)

Ahora veremos una clasificación multicategoría. Supongamos que en una mesa de ayuda se registra el impacto y la urgencia de un caso, y se desea asignar una prioridad: baja, media o alta. La meta es mostrar que la lógica general sigue siendo la misma, aunque ahora la salida ya no tiene dos clases sino tres.

Impacto	Urgencia	Prioridad
1	1	baja
1	2	baja
2	2	media
2	3	media
3	2	alta
3	3	alta

Paso 1. Crear los datos

```
datos_soporte = pd.DataFrame({
    "impacto": [1, 1, 2, 2, 3, 3],
    "urgencia": [1, 2, 2, 3, 2, 3],
    "prioridad": ["baja", "baja", "media", "media", "alta", "alta"]
})
datos_soporte
```

Paso 2. Separar variables

```
X2 = datos_soporte[["impacto", "urgencia"]]
y2 = datos_soporte["prioridad"]
```

Paso 3. Partir en train y test

```
X2_train, X2_test, y2_train, y2_test = train_test_split(
    X2, y2, test_size=0.33, random_state=42
)
```

Paso 4. Entrenar un clasificador sencillo

```
from sklearn.tree import DecisionTreeClassifier

modelo2 = DecisionTreeClassifier(random_state=42)
modelo2.fit(X2_train, y2_train)
```

Paso 5. Predecir nuevas prioridades

```
nuevos_casos = pd.DataFrame({
    "impacto": [1, 3],
    "urgencia": [2, 3]
})

pred2 = modelo2.predict(nuevos_casos)
print(pred2)
```

Paso 6. Leer la predicción con criterio

Si un caso con impacto alto y urgencia alta se clasifica como prioridad alta, la salida parece coherente. Esa lectura humana sigue siendo muy importante: no basta con ejecutar el código; también hay que juzgar si las etiquetas producidas tienen sentido en el contexto del problema.

Prompts de IA generativa para acompañar el ejemplo 2

- "Explícame la diferencia entre clasificación binaria y clasificación multicategoría con ejemplos cotidianos."
- "Ayúdame a interpretar por qué un caso de soporte con alto impacto y alta urgencia debería salir con prioridad alta."
- "Revisa este código de `DecisionTreeClassifier` y explícame cada bloque con lenguaje para principiantes."
- "Redáctame una conclusión breve sobre lo que aprendí en un ejercicio de clasificación multicategoría."

Buenas prácticas para un primer clasificador

Recomendación	¿Por qué importa?
Usar etiquetas claras	Facilita entender qué significa cada clase y evita confusión.
Separar `train` y `test`	Ayuda a comprobar si el clasificador funciona fuera de los ejemplos de entrenamiento.
Empezar simple	Un modelo pequeño y comprensible es mejor para aprender que uno complejo desde el inicio.

Interpretar antes de complicar	Antes de buscar más rendimiento, conviene entender qué dicen los datos y las predicciones.
Usar IA con criterio	La IA generativa debe apoyar la comprensión, no reemplazar el razonamiento del estudiante.

Mini ruta de trabajo sugerida

- Observar la tabla y describir con palabras qué patrón parece existir.
- Identificar con claridad cuáles son las variables de entrada y cuál es la etiqueta.
- Separar entrenamiento y prueba cuando sea posible.
- Entrenar el clasificador y revisar las predicciones.
- Escribir una interpretación breve con lenguaje claro y prudente.

Producto esperado

El producto esperado en esta versión de la guía es un notebook corto y ordenado que contenga dos ejemplos básicos de clasificación, al menos una predicción nueva en cada caso y una interpretación breve sobre el sentido de las etiquetas generadas.

Lista de verificación

■	Revisión
☐	¿Se reconoció correctamente que la salida del problema era una clase y no un número?
☐	¿Se separaron correctamente las variables predictoras y la etiqueta?
☐	¿Se entrenó al menos un clasificador binario sencillo?
☐	¿Se trabajó también un ejemplo multicategoría?
☐	¿Se generaron predicciones nuevas y se interpretaron con palabras sencillas?
☐	¿Se utilizó al menos un prompt de IA generativa para comprender o revisar el trabajo?
☐	¿El notebook quedó organizado, limpio y comentado?

Cierre

La clasificación básica abre la puerta a muchos problemas reales de inteligencia artificial. En esta etapa lo más valioso es comprender el flujo completo: datos de entrada, etiquetas, entrenamiento, predicción e interpretación. Cuando esa lógica queda clara, el estudiante está listo para avanzar hacia métricas más ricas y análisis más finos de errores y aciertos.