Curso base

# INTELIGENCIA ARTIFICIAL DESDE CERO

## LECCIÓN 21
## Preparación de datos y pipeline

Cómo organizar un flujo básico de preparación, entrenamiento y evaluación para construir modelos más ordenados, reutilizables y confiables

| | | | |
|---|---|---|---|
| **Lección** | 21 | **Tema** | Preparación de datos y pipeline |
| **Nombre del participante** | _____ | **Fecha** | _____ |

**Actividad central**

**Actividad principal:** construir un flujo ordenado de preparación de datos, separación train/test, pipeline, entrenamiento, predicción e interpretación.
**Producto esperado:** mini-proyecto 2 de clasificación con flujo ordenado, reutilizable y explicable.

**Propósito de la guía.** Comprender cómo se organiza un flujo de trabajo básico para preparar datos, entrenar un modelo y evaluar sus resultados de forma clara, reproducible y ordenada. Al finalizar esta guía, el participante deberá reconocer que un buen modelo no depende solo del algoritmo, sino también del orden correcto de los pasos.

## Idea central

En aprendizaje automático no basta con cargar una tabla y entrenar un modelo. También necesitamos revisar qué columnas usaremos, separar entrenamiento y prueba, aplicar transformaciones cuando sean necesarias y evaluar el resultado con honestidad. A esa secuencia organizada de pasos la llamamos **pipeline**.

Trabajar con un pipeline ayuda a evitar errores, hace más claro el notebook y permite repetir el proceso con mayor orden. En esta lección no buscamos complejidad técnica; buscamos que el estudiante entienda el flujo correcto y pueda explicarlo con palabras sencillas.

**¿Qué vamos a hacer en esta guía?**

Primero entenderemos qué significa *pipeline* y por qué ayuda. Después construiremos un primer ejemplo guiado de clasificación con un flujo ordenado. Luego trabajaremos un segundo ejemplo guiado para reforzar la idea. En ambos casos usaremos prompts de IA generativa para apoyar la comprensión, revisar errores y redactar interpretaciones claras.

## Vocabulario mínimo

| Concepto | Explicación sencilla |
|---|---|
| **Datos de entrada (X)** | Columnas que usamos para ayudar al modelo a decidir o predecir. |
| **Variable objetivo (y)** | Columna que contiene la respuesta final que queremos predecir. |
| **Transformación** | Cambio útil sobre los datos, por ejemplo escalar o codificar. |
| **Train** | Conjunto de entrenamiento con el que el modelo aprende. |
| **Test** | Conjunto de prueba reservado para evaluar el modelo con casos nuevos. |
| **Pipeline** | Cadena ordenada de pasos que conecta preparación, entrenamiento y evaluación. |

## ¿Por qué un pipeline ayuda tanto?

| Paso | ¿Qué se hace? | ¿Por qué ayuda? |
|---|---|---|
| **1. Preparar datos** | Revisar columnas, tipos y objetivo del problema. | Evita usar columnas incorrectas o confusas. |
| **2. Separar train/-test** | Reservar datos de prueba antes del ajuste final. | Permite una evaluación más honesta. |
| **3. Transformar** | Escalar o preparar variables si el modelo lo necesita. | Mantiene el proceso consistente. |
| **4. Entrenar** | Ajustar el algoritmo con los datos de entrenamiento. | El modelo aprende relaciones útiles. |
| **5. Evaluar** | Comparar predicciones y valores reales. | Ayuda a decidir si el modelo sirve. |

## Ejemplo guiado 1: clasificar compra / no compra con un flujo ordenado

Vamos a imaginar una tabla de clientes con tres variables sencillas: edad, ingreso y número de visitas a una tienda en línea. La variable objetivo será `compra`, donde 1 significa que compra y 0 significa que no compra.

Antes de programar, conviene responder dos preguntas: *¿qué columnas usaré como entrada?* y *¿qué columna quiero predecir?* Aquí las entradas serán `edad`, `ingreso` y `visitas`; la salida será `compra`.

| Edad | Ingreso | Visitas | Compra |
|---|---|---|---|
| 22 | 1.2 | 1 | 0 |
| 25 | 1.8 | 2 | 0 |
| 28 | 2.5 | 3 | 1 |
| 31 | 2.9 | 4 | 1 |
| 36 | 3.4 | 5 | 1 |
| 42 | 1.9 | 1 | 0 |

### Ruta de lectura del ejemplo 1

1. Cargar o crear una tabla pequeña y mirar sus columnas.
2. Identificar `X` e `y` con claridad.
3. Separar train y test antes de entrenar.
4. Construir un pipeline con escalado y un clasificador sencillo.
5. Entrenar el pipeline con los datos de entrenamiento.
6. Generar predicciones sobre prueba y observar si el flujo tiene sentido.

### Código base del ejemplo 1

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

datos = pd.DataFrame({
    "edad": [22,25,28,31,36,42,24,27,33,39],
    "ingreso": [1.2,1.8,2.5,2.9,3.4,1.9,1.4,2.2,3.0,3.6],
    "visitas": [1,2,3,4,5,1,2,3,4,5],
    "compra": [0,0,1,1,1,0,0,1,1,1]
})

X = datos[["edad", "ingreso", "visitas"]]
y = datos["compra"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

pipe = Pipeline([
    ("escala", StandardScaler()),
    ("modelo", LogisticRegression(max_iter=1000))
])

pipe.fit(X_train, y_train)
pred = pipe.predict(X_test)

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

**¿Cómo interpretar este ejemplo?**

Lo importante aquí no es memorizar cada línea, sino comprender el orden. Primero se definen las columnas. Después se separan entrenamiento y prueba. Luego se construye el pipeline. Finalmente se entrena y se evalúa. Ese orden reduce errores y hace más confiable el experimento.
Si el estudiante mezcla estos pasos o cambia el orden, puede terminar con un resultado engañoso. Por ejemplo, transformar datos sin separar primero train y test puede contaminar la evaluación.

**Prompts de IA generativa para acompañar el ejemplo 1**

- Explícame con palabras sencillas qué es un pipeline en aprendizaje automático y por qué conviene usarlo.
- Revisa este código de `Pipeline` con `StandardScaler` y `LogisticRegression`. Dime si el orden de los pasos es correcto y explícame por qué.
- Ayúdame a redactar un comentario breve para explicar por qué separar train y test hace más confiable el resultado.
- Muéstrame los errores más comunes que puede cometer un principiante al construir un pipeline en Google Colab.

## Errores comunes que el pipeline ayuda a evitar

| Error frecuente | Consecuencia |
|---|---|
| **Transformar antes de separar train/test** | Puede dar una evaluación artificialmente buena. |
| **Cambiar columnas sin dejarlo claro** | Dificulta comparar modelos o repetir el trabajo. |
| **No distinguir entre entrenamiento y prueba** | Hace imposible saber si el modelo generaliza. |
| **Ejecutar código sin explicación escrita** | El notebook pierde valor como evidencia de aprendizaje. |

## Segundo ejemplo guiado: aprobar / no aprobar con el mismo flujo

Ahora repetiremos la idea en un contexto educativo. Esto es importante porque un pipeline no es un ejemplo aislado: es una forma de organizar el trabajo que se puede reutilizar en distintos problemas.

En este segundo ejemplo, las entradas serán horas de estudio, tareas entregadas y asistencia. La variable objetivo será `aprobado`, donde 1 significa aprobado y 0 significa no aprobado.

### Ruta de lectura del ejemplo 2

1. Observar la lógica del problema: más horas, más tareas y mejor asistencia suelen asociarse con una mayor probabilidad de aprobar.
2. Preparar la tabla y nombrar claramente la variable objetivo.
3. Separar train y test.
4. Construir el pipeline.
5. Entrenar, predecir e interpretar.

### Código base del ejemplo 2

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

datos2 = pd.DataFrame({
    "horas_estudio": [1,2,2,3,4,4,5,6,7,8],
    "tareas": [0,1,1,1,2,2,3,3,4,4],
    "asistencia": [60,65,70,72,78,80,85,88,92,95],
    "aprobado": [0,0,0,0,1,1,1,1,1,1]
})

X2 = datos2[["horas_estudio", "tareas", "asistencia"]]
y2 = datos2["aprobado"]

X2_train, X2_test, y2_train, y2_test = train_test_split(
    X2, y2, test_size=0.3, random_state=42, stratify=y2
)

pipe2 = Pipeline([
    ("escala", StandardScaler()),
    ("modelo", LogisticRegression(max_iter=1000))
])

pipe2.fit(X2_train, y2_train)
pred2 = pipe2.predict(X2_test)

print("Predicciones:", pred2)
print("Accuracy:", round(accuracy_score(y2_test, pred2), 2))
```

## Comparación entre los dos ejemplos

| Aspecto | Ejemplo 1 | Ejemplo 2 |
|---|---|---|
| **Problema** | Compra / no compra | Aprobado / no aprobado |
| **Entradas** | Edad, ingreso, visitas | Horas, tareas, asistencia |
| **Objetivo** | Clasificación binaria | Clasificación binaria |
| **Mensaje central** | El pipeline ordena el flujo | El mismo flujo puede reutilizarse |

**Prompts de IA generativa para acompañar el ejemplo 2**

- Compara estos dos ejemplos y explícame qué pasos del pipeline se repiten en ambos.
- Ayúdame a redactar una interpretación corta del resultado del clasificador usando lenguaje para principiantes.
- Explícame por qué un notebook ordenado vale más que un notebook con mucho código pero sin estructura.
- Dime cómo verificar si mis columnas de entrada y mi variable objetivo están bien elegidas.

## Buenas prácticas para esta lección

| Recomendación | ¿Por qué importa? |
|---|---|
| **Separar primero train y test** | Protege la evaluación y evita conclusiones engañosas. |
| **Nombrar con claridad X, y y el pipeline** | Hace más entendible el notebook. |
| **Usar la misma secuencia en varios ejemplos** | Ayuda a fijar la lógica del proceso. |
| **Escribir una conclusión breve al final** | Convierte el trabajo en evidencia de aprendizaje. |

**Producto esperado**

El producto esperado en esta versión de la guía es un notebook corto, claro y ordenado, donde el participante muestre un flujo básico de preparación de datos, separación train/test, construcción del pipeline, entrenamiento, predicción e interpretación final del resultado.

## Lista de verificación

| Revisión | Criterio |
|---|---|
| ☐ | ¿Se explicó con claridad cuál es la variable objetivo? |
| ☐ | ¿Se separaron los datos de entrenamiento y prueba? |
| ☐ | ¿Se construyó un pipeline con pasos identificables? |
| ☐ | ¿Se entrenó el modelo y se generaron predicciones? |
| ☐ | ¿Se redactó una interpretación breve del resultado? |
| ☐ | ¿El notebook puede leerse como una secuencia ordenada de pasos? |

## Cierre

Trabajar con pipeline es dar un paso hacia una práctica más seria y ordenada en aprendizaje automático. El valor de esta lección no está en aprender una palabra nueva, sino en comprender que un buen modelo nace de un buen proceso. Cuando el estudiante entiende el orden de preparación, entrenamiento y evaluación, queda mejor preparado para construir modelos más confiables en las lecciones siguientes.