EJERCICIO PRÁCTICO: APLICACIÓN DE ALGORITMOS ML

Duración: 5 horas - 300 minutos

FECHA: 17/10/2025 ESTUDIANTE: Ignacio Sánchez García

PERFIL: INGENIERO DE SISTEMAS

# Elige tu Camino Profesional

A continuación se presentan cuatro perfiles profesionales. Cada uno tiene un caso de estudio específico para aplicar un modelo de Machine Learning. Elige el que más se ajuste a tus intereses y sigue las fases del proyecto descritas más abajo.

## CAMINO 1: INGENIERO DE SISTEMAS

### CASO: Predicción de Carga de Servidor / Detección de Anomalías

**PROBLEMA:** Un administrador de sistemas necesita predecir cuándo un servidor puede saturarse para escalar recursos proactivamente. Usaremos regresión logística para clasificar "servidor OK" vs "servidor en riesgo".

### DATASET - GENERACIÓN RÁPIDA (1 MINUTO):

from sklearn.datasets import make\_classification import pandas as pd

X, y = make\_classification(n\_samples=500, n\_features=5,

n\_informative=4, random\_state=42)

df = pd.DataFrame(X, columns=['CPU\_%', 'Memoria\_%', 'Conexiones',

'Ancho\_banda\_MB', 'Latencia\_ms'])

df['Servidor\_Saturado'] = y

df.to\_csv('servidor\_data.csv', index=False) print(df.head())

### PREGUNTAS A RESOLVER:

¿Cuál es el umbral de CPU que predice saturación?

¿La memoria es más importante que el ancho de banda?

¿Prefiero falsos positivos o falsos negativos?

## CAMINO 2: TÉCNICO MICROINFORMÁTICA (SMR)

### CASO: Clasificación de Fallas de Hardware

**PROBLEMA:** Un taller de reparación clasifica equipos con problemas. Necesitas predecir si la falla es de SOFTWARE, HARDWARE MENOR o HARDWARE CRÍTICO para derivar a diferentes técnicos. Este es un problema de LOGIT MULTINOMIAL (3 clases).

### DATASET - GENERACIÓN:

from sklearn.datasets import make\_classification import pandas as pd

X, y = make\_classification(n\_samples=400, n\_features=8,

n\_informative=6, n\_classes=3,

random\_state=42)

df = pd.DataFrame(X, columns=['Temp\_CPU', 'Voltaje', 'Fallos\_previos',

'Ruido\_dB'])

'Horas\_uso', 'Test\_diagnostico', 'Humedad\_%', 'Vibracion',

df['Tipo\_Falla'] = [['Software', 'Hardware\_Menor',

'Hardware\_Critico'][i] for i in y] df.to\_csv('hardware\_faults.csv', index=False)

### PREGUNTAS A RESOLVER:

¿Qué variables predicen mejor cada tipo de falla?

¿Es equilibrado el dataset?

¿Qué error sería más costoso misclasificar?

## CAMINO 3: FP SMR - SEGURIDAD EN REDES

### CASO: Detección de Spam / Emails Maliciosos

**PROBLEMA:** Una empresa necesita filtrar emails automáticamente. Usarás LOGIT MULTINOMIAL para clasificar: LEGÍTIMO, SPAM, PHISHING.

### DATASET - DESCARGA O GENERACIÓN:

import pandas as pd df = pd.read\_csv(

"[https://raw.githubusercontent.com/justmarkham/DAT8/master/data/spam.](https://raw.githubusercontent.com/justmarkham/DAT8/master/data/spam) encoding='latin-1')

print(df.head())

df.to\_csv('emails\_spam.csv', index=False)

### PREGUNTAS A RESOLVER:

¿Cuál es la característica más indicativa de phishing?

¿Es más común spam o phishing?

Costo de falso positivo vs falso negativo en seguridad.

## CAMINO 4: INGENIERO AERONÁUTICO

### CASO: Predicción de Mantenimiento de Motores

**PROBLEMA:** Necesitas predecir fallos antes de que ocurran: MOTOR NORMAL, REQUIERE MANTENIMIENTO, RIESGO CRÍTICO.

### DATASET - GENERACIÓN:

import pandas as pd import numpy as np

np.random.seed(42) data = {

'Temperatura\_Entrada': np.random.normal(500, 50, 600),

'Presion\_Combustible': np.random.normal(45, 5, 600),

'RPM': np.random.normal(10000, 1000, 600),

'Vibracion': np.random.exponential(0.5, 600),

'Consumo\_Combustible': np.random.normal(8000, 500, 600),

'Temperatura\_Escape': np.random.normal(700, 80, 600),

}

def clasificar\_motor(row):

score = (row['Temperatura\_Entrada']/500 + row['Vibracion']\*2 + row['Temperatura\_Escape']/800)

if score > 3: return 'Critico'

elif score > 2: return 'Mantenimiento' else: return 'Normal'

df = pd.DataFrame(data)

df['Estado\_Motor'] = df.apply(clasificar\_motor, axis=1) df.to\_csv('motores\_aviacion.csv', index=False)

### PREGUNTAS A RESOLVER:

¿Qué sensor es más predictivo de fallo?

¿Cuánto tiempo antes del fallo se detectan cambios?

Costo de parada innecesaria vs accidente.

# Fases Comunes del Proyecto

## FASE 1: CARGA Y EXPLORACIÓN (30 MINUTOS)

import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

# CARGAR (usa el nombre de tu dataset: 'servidor\_data.csv', 'hardware\_faults.csv', etc.)

df = pd.read\_csv('tu\_dataset.csv') print(f"Shape: {df.shape}") print(df.head())

print(df.info()) print(df.describe())

# EXPLORAR DESBALANCE (reemplaza 'target' por tu columna objetivo) print(df['target'].value\_counts())

sns.countplot(x='target', data=df) plt.show()

# REVISAR FALTANTES

print(df.isnull().sum())

### ANOTAR:

Muestras: 500 Variables: 6

Clases balanceadas (Si/No): sí

Valores faltantes: 0

## FASE 2: PREPARACIÓN DE DATOS (30 MINUTOS)

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler

# Reemplaza 'target' por el nombre de tu columna objetivo

X = df.drop('target', axis=1) y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=42, stratify=y

)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) X\_test\_scaled = scaler.transform(X\_test)

print(f"Train: {X\_train\_scaled.shape}") print(f"Test: {X\_test\_scaled.shape}")

### ANOTAR:

Train size: (400, 5)

Test size: (100, 5)

¿Por qué escalar? Para que todas las variables tengan igual peso en el modelo.

## FASE 3: ENTRENAR MODELO (30 MINUTOS)

from sklearn.linear\_model import LogisticRegression from sklearn.metrics import classification\_report, confusion\_matrix

modelo = LogisticRegression(max\_iter=1000, multi\_class='multinomial',

solver='lbfgs', random\_state=42) modelo.fit(X\_train\_scaled, y\_train)

y\_pred = modelo.predict(X\_test\_scaled) y\_proba = modelo.predict\_proba(X\_test\_scaled)

print("=== EXACTITUD ===")

print(f"Accuracy Train: {modelo.score(X\_train\_scaled, y\_train):.3f}")

print(f"Accuracy Test: {modelo.score(X\_test\_scaled, y\_test):.3f}")

print("\n=== REPORTE DE CLASIFICACIÓN ===")

print(classification\_report(y\_test, y\_pred))

print("\n=== MATRIZ DE CONFUSIÓN ===")

cm = confusion\_matrix(y\_test, y\_pred) print(cm)

### ANOTAR:

Accuracy Train: 0.695

Accuracy Test: 0.680

¿Sobreentrenamiento? (Train >> Test): Si/No Sí

## FASE 4: VISUALIZACIONES (20 MINUTOS)

### MATRIZ DE CONFUSIÓN:

import seaborn as sns

import matplotlib.pyplot as plt

cm = confusion\_matrix(y\_test, y\_pred) sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=modelo.classes\_, yticklabels=modelo.classes\_)

plt.title('Matriz de Confusión - Test Set') plt.ylabel('Real') plt.xlabel('Predicción') plt.tight\_layout()

plt.show()

### IMPORTANCIA DE VARIABLES:

# Asegúrate de que X.columns esté disponible desde la Fase 2 coef\_df = pd.DataFrame(

modelo.coef\_, columns=X.columns, index=modelo.classes\_

)

print("\n=== COEFICIENTES DEL MODELO ===")

print(coef\_df)

coef\_df.T.plot(kind='barh', figsize=(10, 6)) plt.title('Importancia de variables por clase') plt.xlabel('Coeficiente (peso)') plt.tight\_layout()

plt.show()

# FASE 5: REPORTE FINAL (60 MINUTOS)

## REPORTE DE ANÁLISIS DE MACHINE LEARNING

**ESTUDIANTE: Ignacio Sánchez García**

**FECHA: 20251017**

**PERFIL: CAMINO 1: INGENIERO DE SISTEMAS**

### INTRODUCCIÓN

*Describe por qué es importante este problema en tu área profesional (3-4 líneas):*

*Es importante para dimensionar infraestructura con volumen de negocio, ayudando a corregir fallos balanceando las cargas entre servidores o indicando si hay que ampliar el hardware del negocio o las lineas para evitar fallas en el servicio.*

### DATASET

Total de muestras: 500

Variables usadas: 6

Número de clases: 2

Clases: **0:** Servidor OK; **1:** Servidor Saturado.

Dataset balanceado (Si/No): Sí

### MODELO ENTRENADO

Algoritmo: Regresión Logística Multinomial Train/Test split: 80/20

Escalado: StandardScaler

### RESULTADOS PRINCIPALES

Accuracy en Train: 69.5%

Accuracy en Test: 68.0%

Precisión promedio: 0.68

Recall promedio: 0.68

F1-Score promedio: 0.68

### VARIABLES MÁS IMPORTANTES

Top 3 variables:

* 1. CPU\_% (coeficiente: ~0.783 )
  2. Memoria\_% (coeficiente: ~0.450 )
  3. Conexiones (coeficiente: ~0.302 )

*¿Qué significan en tu negocio?*

**CPU es el** **enemigo N.º 1**, seguido de cerca por la **Memoria**, lo que dicta

dónde debe ir cada euro invertido en infraestructura para lograr el

mayor **Retorno de la Inversión (ROI)**en disponibilidad del servicio.

### ANÁLISIS DE LA MATRIZ DE CONFUSIÓN

¿Cuál es la clase mejor predicha? Clase 0 (Servidor OK)

¿Cuál tiene más errores? Clase 1 (Servidor Saturado)

Falsos positivos más comunes: 12 Falsos Positivos (FP)

Falsos negativos más comunes: 20 Falsos Negativos (FN)

*¿Cuál sería más costoso? ¿Por qué?*

Falsos Negativos (FN) porque el costo es asimétrico.

### CONCLUSIONES OPERACIONALES

¿Es el modelo lo suficientemente bueno para usar? (Si/No) No

*¿Por qué?*

principalmente debido al bajo **Recall** de la clase crítica

*¿Qué mejorías harías?*

1. Ajuste del UmbralBajar el umbral de decisión (de $0.5$ a $0.4$ o $0.35$). Esto aumentará el **Recall** (menos FN) a expensas de aumentar los **FP** (más alertas innecesarias), lo cual es el *trade-off* económico preferido.

2. Ingeniería de Características (*Feature Engineering*) Mejorar el poder predictivo

*3.* Probar Modelos Avanzados para aumentar la precision

***Recomendación para implementar en producción:***

*La recomendación operativa debe ser cautelosa y escalonada:*

1. ***NO IMPLEMENTAR EN MODO AUTOMÁTICO:*** *El modelo* ***no debe*** *controlar el autoscaling automáticamente, ya que el riesgo de un FN (caída del servicio) es demasiado alto.*
2. ***IMPLEMENTAR EN MODO ASISTENCIA/ALERTA:***
   * ***Acción 1 (Inmediata):*** *Implementar el modelo solo como un* ***sistema de alerta temprana*** *para el administrador. Si el modelo predice "Saturado" (1), se envía una alerta al equipo de operaciones, que verifica manualmente el servidor.*
   * ***Acción 2 (Ajuste Necesario):******Ajustar el umbral de clasificación*** *a un valor que dé un* ***Recall de al menos $\mathbf{0.85}$ o $\mathbf{0.90}$*** *en el Test Set. Este es el nivel mínimo de seguridad para confiar en la detección de fallos reales.*
3. ***Monitoreo del Desviación (Drift):*** *Si se implementa, se debe monitorear la precisión del modelo diariamente. Las tendencias reales de tráfico y el uso del servidor cambian (drift), y la precisión del modelo puede degradarse rápidamente.*

### REFLEXIÓN FINAL

*¿Qué aprendiste sobre algoritmos logísticos?*

Son muy útiles a la hora de evaluar el riesgo de saturacion de un sistema cayendo en la denegacion de servicio. Ayudan a su vez a elegir donde se ha de invertir el dinero para mejorar la infraestructura.

# CHECKLIST DE TRABAJO

[ x] Dataset cargado y explorado

[x ] Datos limpios y escalados

[x ] Modelo entrenado exitosamente

[ x] Matriz de confusión generada

[x ] Variables importantes identificadas

[ x] Gráficos generados y guardados

[ x] Reporte completado

[ x] Conclusiones redactadas

[ ] Archivo entregado

TIMELINE: 5 HORAS = 300 MINUTOS

**Tiempo**

**Actividad**

0-30 min

Presentación + Descarga/generación dataset

30-60 min Exploración EDA (head, describe, value\_counts)

60-120 min Preparación datos (train/test + scaling)

120-180 min Entrenar modelo + evaluar métricas

180-240 min Visualizaciones + análisis de coeficientes

240-300 min Redactar reporte + conclusiones

Referencias

[1] trabajo 1

[https://static-us-img.skywork.ai/prod/analysis/2025-10-17/3780596342200386924/1979182 369802788864\_e2ef2fbbe9e243ef15a42b976e972bdc.txt](https://static-us-img.skywork.ai/prod/analysis/2025-10-17/3780596342200386924/1979182369802788864_e2ef2fbbe9e243ef15a42b976e972bdc.txt)