

# Clasificación de Diagnósticos Médicos

con Redes Neuronales MLP

---

**Universidad Militar Nueva Granada**

Dataset: Exámenes de laboratorio · Consulta externa Perú

Febrero 2026

# CONTENIDO DE LA PRESENTACIÓN

01

## Descripción del Dataset

Origen, variables y estructura de los datos

02

## Pre-procesamiento

Limpieza, codificación y normalización

03

## Distribución Train / Val / Test

Estrategia 70% / 15% / 15% con stratify

04

## Arquitectura MLP

Capas Dense, BatchNorm, Dropout, He Normal

05

## Selección de Hiperparámetros

Búsqueda manual entre 4 configuraciones

06

## Entrenamiento y Evaluación

Curvas de accuracy/loss, métricas de test

# 01 | DESCRIPCIÓN DEL DATASET



## Información General

**Fuente:** datosabiertos.gob.pe (Perú)

**Tipo:** Exámenes laboratorio · Consulta externa

**Objetivo:** Clasificar diagnósticos médicos (CIE-10)

**Variable Y:** DIAGNOSTICO (multiclasé)

**Split RAM:** 20% del dataset (SAMPLE\_FRACTION=0.2)



## Pipeline de Datos

1 Cargar CSV (auto-separador)

2 Reducir dataset (SAMPLE\_FRACTION)

3 Eliminar duplicados y nulos

4 Codificar variables categóricas

5 One-Hot Encoding features

6 Imputar valores nulos (mediana)

7 Normalizar con StandardScaler

## 02 | PRE-PROCESAMIENTO DE DATOS



### Duplicados

Se detectan y eliminan filas duplicadas para evitar sesgo en el entrenamiento.



### Nulos

Columnas con >50% nulos se eliminan. Las restantes se imputan con la mediana.



### Codificación

Variable objetivo → LabelEncoder. Features categóricos → One-Hot Encoding.



### Normalización

StandardScaler entrenado SOLO en Train, aplicado a Val y Test sin fuga de datos.

# 03 | DISTRIBUCIÓN TRAIN / VAL / TEST

70%

TRAIN

15%

VALIDACIÓN

15%

TEST

## Stratify

Se usa stratify=y en ambos splits para preservar la distribución de clases en cada subconjunto.

## Clases < 2 muestras

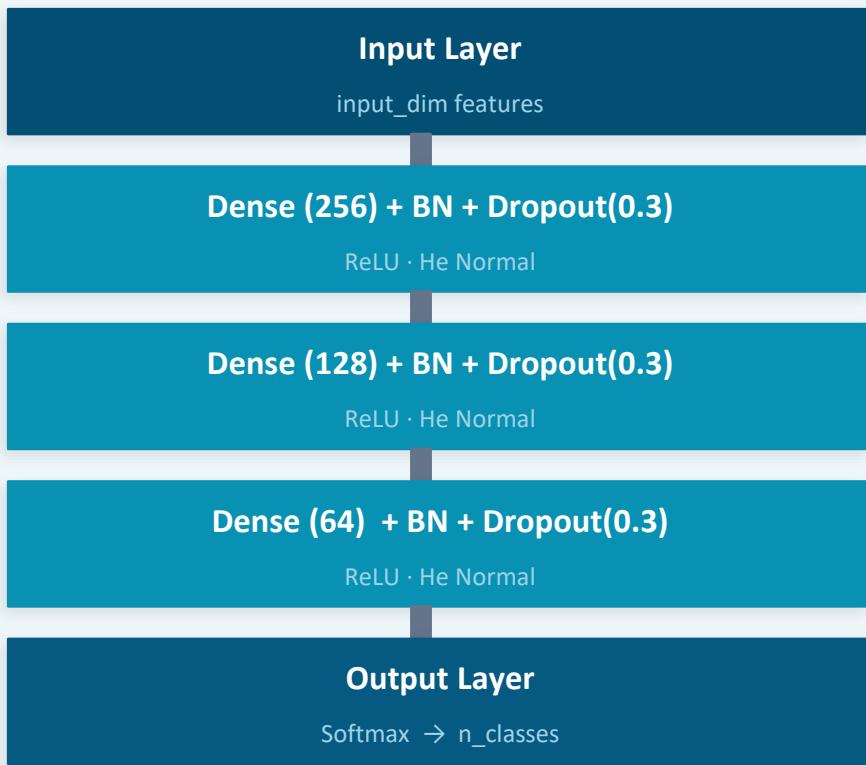
Se filtran automáticamente las clases con menos de 2 instancias, incompatibles con stratify.

## Re-encoding

Tras filtrar, se re-encodean las etiquetas con un segundo LabelEncoder para obtener 0..N-1 sin huecos.

$$test\_size_2 = \text{round}(0.15 / 0.85, 6) \approx 17.647\% \rightarrow 15\% \text{ del total original}$$

# 04 | ARQUITECTURA DEL MODELO MLP



Detalles Técnicos	
<b>Tipo de modelo:</b>	Sequential (MLP)
<b>Inicialización:</b>	He Normal (seed=42)
<b>Activación oculta:</b>	ReLU
<b>Activación salida:</b>	Softmax / Sigmoid
<b>Regularización:</b>	Dropout + BatchNorm
<b>Optimizador:</b>	Adam ( $\text{lr}=0.001$ )
<b>Loss multiclasa:</b>	Sparse Categorical CE

Input(shape=...) en vez de input\_dim= (Keras 3 compatible)

# 05 | SELECCIÓN DE HIPERPARÁMETROS

Config	Capas	Dropout	LR	Batch	Épocas máx
1	[64, 32]	0.2	0.001	256	30
2	<b>[128, 64]</b>	<b>0.3</b>	<b>0.001</b>	<b>256</b>	<b>30</b>
3	[256, 128, 64]	0.3	0.001	64	150
4	[256, 128, 64, 32]	0.4	0.0005	64	150

Callbacks aplicados en TODAS las configuraciones

## EarlyStopping

```
monitor=val_loss  
patience=5  
restore_best_weights=True
```

## ReduceLROnPlateau

```
monitor=val_loss  
factor=0.5 · patience=3  
min_lr=1e-5
```

## clear\_session()

Se llama entre cada config para liberar RAM de Keras  
+ gc.collect()

# 06 | ENTRENAMIENTO Y EVALUACIÓN

## 🏆 Configuración Final

**Arquitectura:** [256, 128, 64]

**Dropout:** 0.3

**Learning Rate:** 0.001 (Adam)

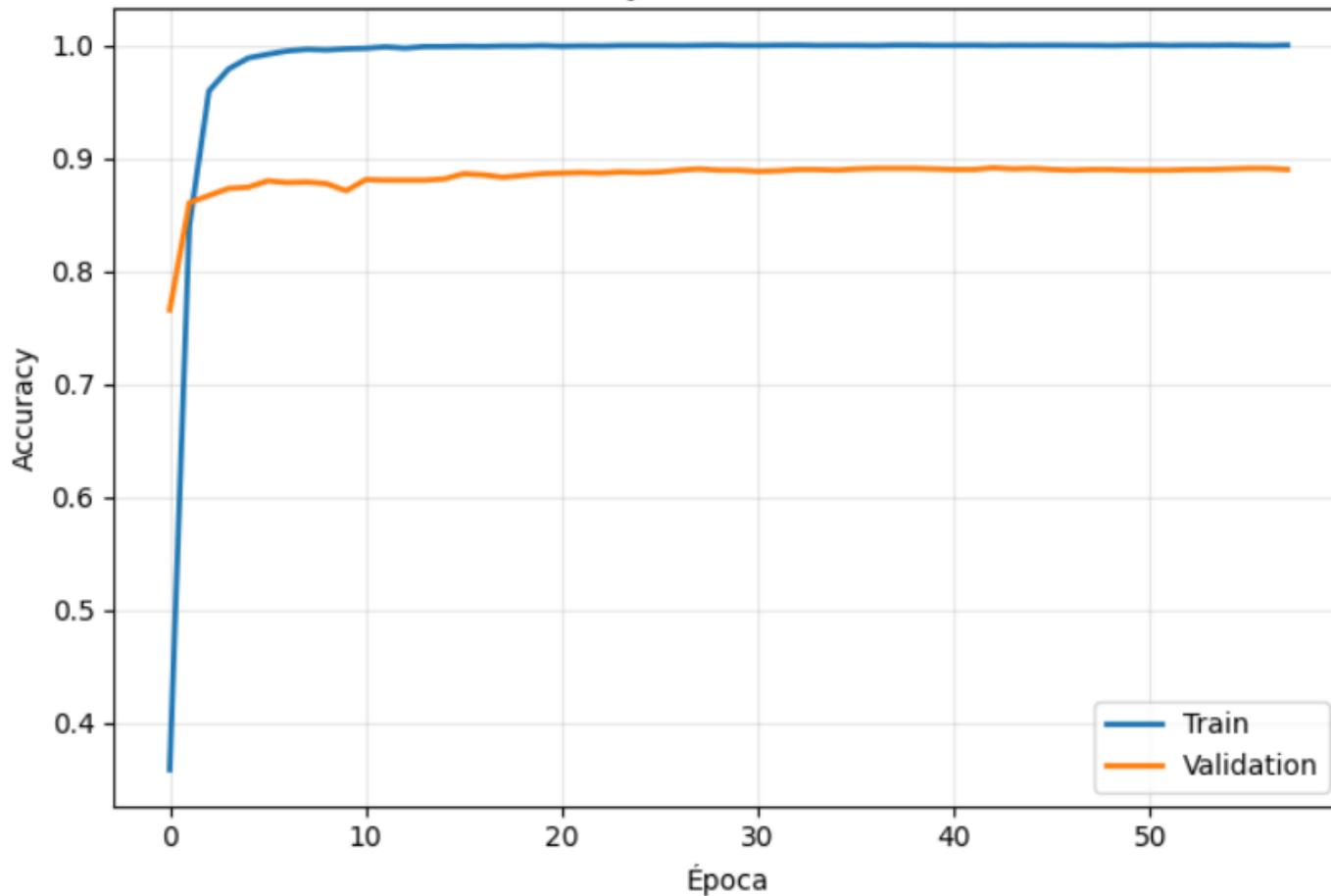
**Batch Size:** 64

**Épocas máx:** 150 (EarlyStopping p=15)

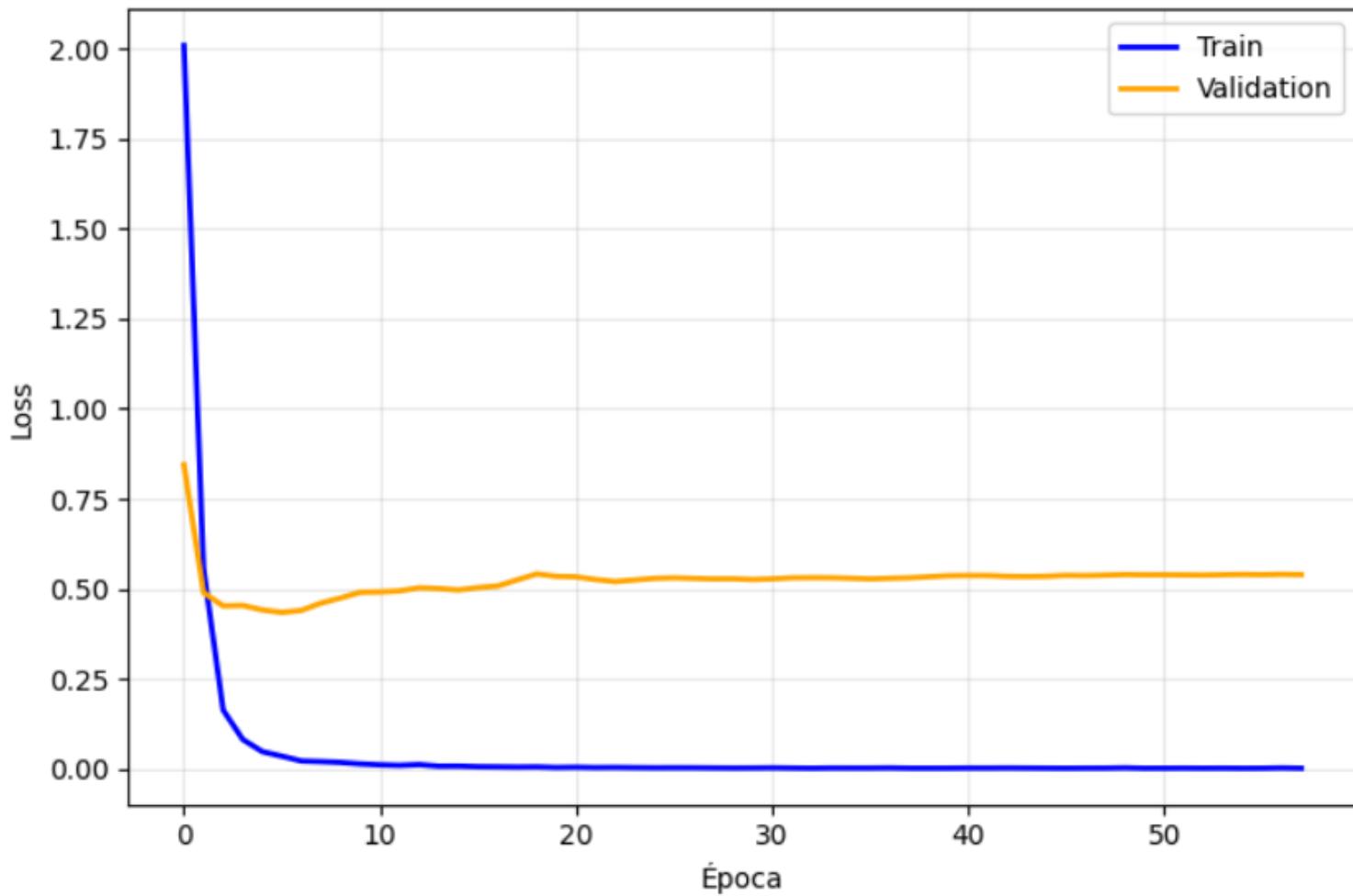
## Curvas de Entrenamiento

- Accuracy Train vs Validation
- Loss Train vs Validation
- Guardadas en curvas\_entrenamiento.png para análisis de overfitting/underfitting

### Accuracy: Train vs Validation



# Pérdida (Loss): Train vs Validation



# 06 | EVALUACIÓN FINAL — DATOS DE TEST

## classification\_report()

- Precision · Recall · F1-score por clase
- zero\_division=0 → evita errores con clases sin predicciones

## confusion\_matrix()

- Heatmap con seaborn (cmap='Blues')
- Guardada en matriz\_confusion.png

## Correcciones Aplicadas al Código

### RAM / Sesión

SAMPLE\_FRACTION=0.2  
gc.collect() tras cada paso  
clear\_session() entre modelos

### NameError

assert var in globals()  
antes de cada celda crítica  
para diagnóstico rápido

### Keras 3

Input(shape=...) en vez  
de input\_dim= (depreciado)  
en TF/Keras ≥ 2.x

### Predicciones

Manejo binario (sigmoid)  
vs multiclase (softmax)  
Shape (n,1) vs (n,k)

# CONCLUSIONES



El modelo MLP es capaz de clasificar diagnósticos médicos multiclasificación con capas Dense, BatchNorm y Dropout para regularización.



La gestión de RAM es crítica en Colab gratuito: SAMPLE\_FRACTION, clear\_session() y gc.collect() son esenciales para evitar crashes.



El StandardScaler se entrena SOLO en Train para garantizar que no haya data leakage hacia Val y Test.



La búsqueda manual de hiperparámetros permite comparar arquitecturas y seleccionar la óptima por Val Accuracy.



El modelo final se guarda en formato .keras (compatible con Keras 3) para su reutilización y despliegue futuro.