

# Test Design (Segunda iteración)

En esta fase se define cómo se medirá la generalización del modelo y se especifica el particionado de datos. Se establecen las métricas primarias y secundarias en unidades operativas y se fijan umbrales de aceptación operativa que conectan el rendimiento técnico con decisiones veterinarias.

El resultado esperado es un plan de experimentos claro que evite fugas de información y permita comparar modelos de forma justa.

## 2.1. DIVISIÓN DE DATOS

Para garantizar una evaluación robusta del modelo LSTM, prevenir overfitting y respetar la naturaleza temporal de los datos, se implementó una estrategia de división especializada que considera las características únicas de series temporales y la estructura secuencial del problema.

La división de 80-20% se fundamenta en principios de machine learning para series temporales y consideraciones prácticas del volumen de secuencias disponibles:

### Training Set (80% ~ N secuencias):

- **Propósito principal:** Entrenamiento del modelo LSTM y ajuste de hiper parámetros mediante validación interna.
- **Composición:** Incluye secuencias de múltiples vacas en diferentes etapas de lactancia (temprana, media, tardía) para capturar toda la variabilidad fisiológica.
- **Ventaja temporal:** Proporciona suficientes ejemplos de patrones de declive, picos de producción y transiciones entre fases para que el LSTM aprenda dependencias temporales complejas.
- **Estructura:** Cada muestra es una secuencia de 30 días con 50+ features por timestep.
  - Shape: (n\_train\_sequences, 30 timesteps, 50+ features)
  - Representa aproximadamente [COMPLETAR: X] ciclos de lactancia completos

### Test Set (20% ~ N secuencias):

- **Función crítica:** Simular condiciones de producción con secuencias completamente nuevas no vistas durante entrenamiento.
- **Preservación de integridad:** Nunca utilizado durante desarrollo del modelo, callbacks de early stopping o ajuste de arquitectura.
- **Tamaño adecuado:** [COMPLETAR: ~X] secuencias proporcionan significancia estadística para métricas de evaluación (MAE, RMSE, R<sup>2</sup>).
- **Representatividad:** Incluye vacas en diferentes rangos de DIM (días en leche) para evaluar generalización del modelo:
  - 0-30 días hasta secado (predicción a corto plazo)

- 31-60 días (ventana óptima de decisión)
- 61-120 días (planificación anticipada)
- 121+ días (pronóstico temprano)

## 2.2. Time Series Split Tradicional

A diferencia de problemas de forecasting clásicos (predecir próximos valores de una misma serie), nuestro problema tiene características particulares:

- Cada vaca representa una serie temporal independiente. La secuencia de la Vaca #123 no tiene relación causal con la Vaca #456.
  - En lugar de una serie larga continua, tenemos múltiples series (una por vaca/ciclo) de longitud moderada.
- Queremos que el modelo funcione en vacas nuevas, no solo predecir el futuro de vacas ya vistas.

## 2.3. VALIDACIÓN DURANTE ENTRENAMIENTO

Durante el entrenamiento del LSTM, implementamos una segunda división dentro del training set:

```
Python
## División automática durante training*
history = model.fit(
    X_train, y_train,
    validation_split=0.2,  ## 20% de X_train usado como
validación*
    epochs=100,
    batch_size=32,
    callbacks=[early_stopping, reduce_lr]
)
```

### Training Subset (64% del total):

- Usado para actualización de pesos del modelo en cada epoch.
- El modelo calcula gradientes y aplica backpropagation solo con estos datos.
- Shape: `(n_train * 0.8, 30, 50+)`.

### Validation Subset (16% del total):

- **Función:** Monitorear generalización durante entrenamiento sin afectar pesos del modelo
- **Uso en callbacks:**
  - **Early Stopping:** Detiene entrenamiento si `val_loss` no mejora durante 15 epochs consecutivos.

- **ReduceLROnPlateau:** Reduce learning rate si `val_loss` se estanca durante 5 epochs.
- **Prevención de overfitting:** Si `train_loss` sigue bajando pero `val_loss` sube, indica overfitting y el early stopping interviene.
- **Preservación:** Nunca se usa para ajuste de hiper parámetros manuales (eso requeriría un tercer conjunto).

## 2.4. ESTRATEGIA DE SECUENCIAS TEMPORALES

### Construcción de Secuencias para LSTM

Proceso de generación:

```
Python
sequence_length = 30  *# Ventana de observación*

for i in range(len(data) - sequence_length):
    X_sequence = data[i : i + sequence_length]  *# 30 días de
historia*
    y_target = target[i + sequence_length]      *#
Predicción: día 31*

    X_sequences.append(X_sequence)
    y_targets.append(y_target)
```

**Características de las secuencias:**

1. **Predicción de 30 días:**
  - Cada secuencia captura exactamente 30 timesteps consecutivos.
  - El modelo observa días  $t$ ,  $t+1$ , ...,  $t+29$  para predecir el target en  $t+30$
2. **Train y test:**
  - Secuencias del test set provienen de vacas o segmentos temporales completamente separados del training set.
  - Garantiza que el modelo no tiene "memory leak" de información futura.
3. **Manejo de ciclos incompletos:**
  - **Ciclos <30 días de datos:** Se excluyen completamente (insuficiente historia para una secuencia).
  - **Ciclos válidos:** Sólo aquellos con al menos 30 días consecutivos de registros de ordeño.

## 2.5. Normalización de Features

Aplicación de MinMaxScaler:

Python

```
## Creación de secuencias deslizantes*
sequence_length = 30  ## Ventana de observación*

for i in range(len(data) - sequence_length):
    X_sequence = data[i : i + sequence_length]  ## 30 días de
    historia*
    y_target = target[i + sequence_length]      ##
    Predicción: día 31*

    X_sequences.append(X_sequence)
    y_targets.append(y_target)
```

- **Scaler se ajusta SOLO con training data:** Los valores min/max usados para normalización provienen exclusivamente del training set.
- **Test data se transforma con parámetros de train:** Simula condiciones reales donde datos nuevos se normalizan con estadísticas históricas.

#### Implicación práctica:

- Si una vaca en test set tiene producción ligeramente fuera del rango observado en train (e.g., 2 kg más alta que el máximo histórico), el scaler mapeará ese valor a >1.0, que luego se clipea a 1.0.
- Esto es intencionado: el modelo debe aprender que valores extremos indican situaciones excepcionales.