Results Evaluation (Tercera iteración)

4.1. Resultados generales

Todos los modelos se entrenaron sobre el mismo dataset balanceado y fuertemente aumentado (≈90,660 registros, 8 features), usando un train/test split 80/20 con `random_state=42`.

Métricas en test (clasificación de “Estado de Producción”)

Modelo	Accuracy	Macro F1	Comentario rápido
Regresión Logística	0.785	0.64	Baseline lineal, claramente con underfitting.
Random Forest	0.992	0.99	Rendimiento casi perfecto en todas las clases. Muy sospechoso dado el tamaño/diseño del dataset.
XGBoost	0.975	0.95	Muy alto desempeño, con pequeña brecha train/test (97.9 vs 97.5).
MLP	0.934	0.87	Buen desempeño global; algo más débil en “Previo a Secado”.
TabNet	0.889	0.82	Menor accuracy, pero muy buen recall para clases minoritarias.

A nivel numérico puro, Random Forest domina, seguido de XGBoost y MLP, con Regresión Logística claramente atrás. Pero esos números, como vemos, están inflados por cómo se construyó y particionó el dataset.

4.2. Interpretación de resultados

Regresión Logística

- Accuracy moderado (~0.78) con macro F1 = 0.64, y gran disparidad entre clases:
  - “En Producción”: precisión 0.98, recall 0.84 (la clase fácil y dominante).
  - “En Monitoreo” y “Previo a Secado”: precisión 0.48/0.40 y recall 0.61/0.63.
- Interpretable, estable, pero no captura bien lo no lineal entre estados productivos. Sirve como baseline, no como modelo final.

Random Forest

- Accuracy 0.9921 con F1 casi perfecto en las tres clases (0.99, 0.99, 0.98).
- Esto significa que, el modelo casi memoriza la relación entre features y estado.

- Dado que las reglas de augmentation imponen rangos casi deterministas por estado, un Random Forest es casi un “buscador de umbrales” de esas reglas. Es decir: aprende perfectamente las reglas que se impusieron.

XGBoost

- Accuracy 0.9747, macro F1 0.95, con desempeño fuerte incluso en “Previo a Secado” (precisión 0.92, recall 0.91).
- Train vs test: 0.9787 vs 0.9747 brecha mínima, señal de buen control del overfitting dentro de este dataset.
- Usa RandomizedSearchCV con `f1_macro` como métrica y early stopping sobre el set de prueba, lo que añade regularización efectiva.
- Menos “perfecto” que RF, pero más razonable si pensamos en generalización.

MLP

- Accuracy 0.934, macro F1 0.87; muy bueno en “En Producción” (0.97/0.97) y decente en “Monitoreo” (0.84/0.92), pero baja algo en “Previo a Secado” (0.80/0.73).
- Las curvas de entrenamiento (loss, accuracy, precision, recall) para train vs validation están bastante pegadas, lo que indica poco overfitting clásico gracias a dropout, early stopping, etc.
- Modelo más complejo y costoso de entrenar/deployar que RF/XGB, sin mejorar sus métricas.

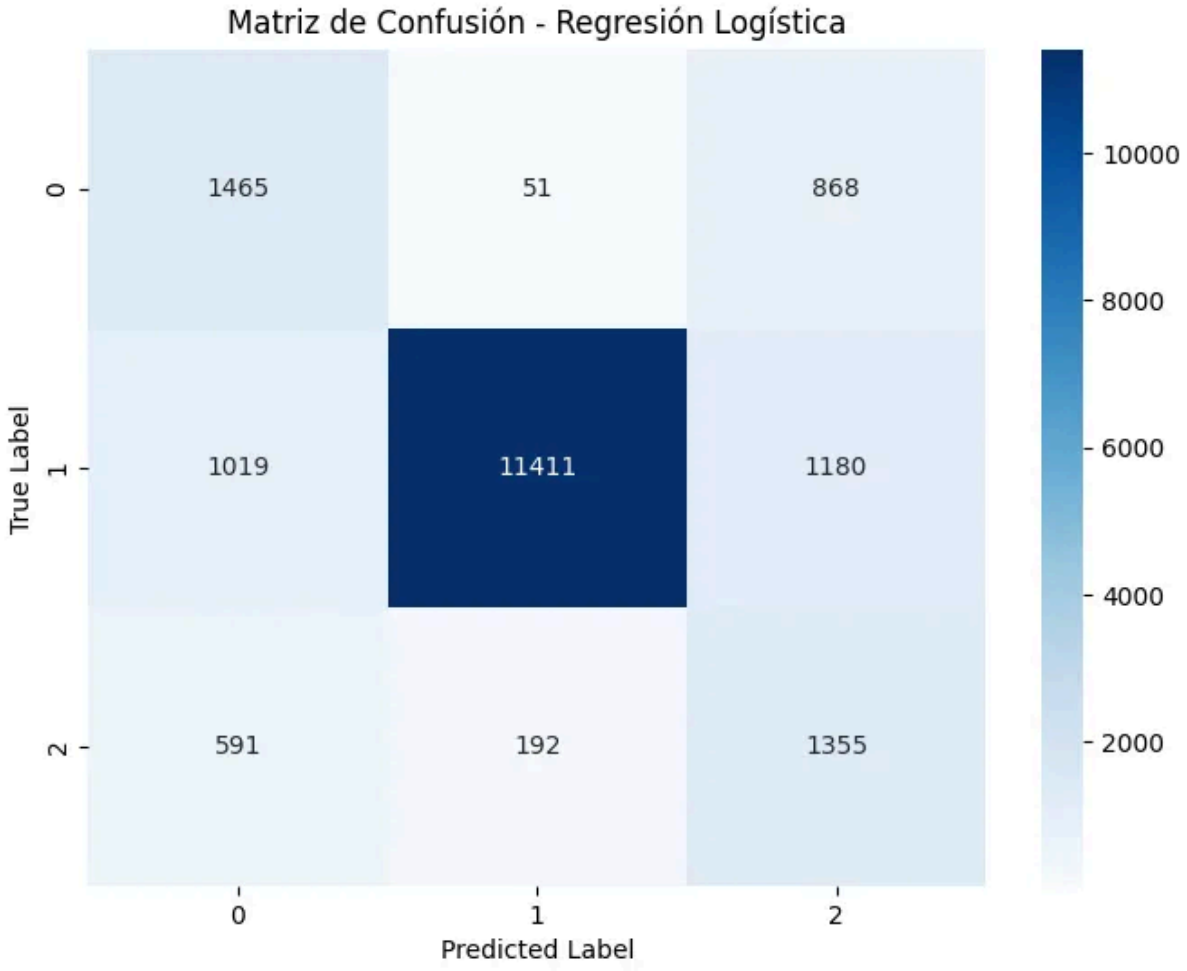
TabNet

- Accuracy ~0.889 con macro F1 0.816.
- Fuerte en recall para clases críticas:
    - “En Monitoreo”: recall 0.92,
    - “Previo a Secado”: recall 0.80,
    - sacrificando precisión (más falsos positivos).
- Es un modelo más pesado y sofisticado para tabular, pero aquí no logra superar a XGB/MLP, quizá por falta de fine tuning o por el tamaño relativamente limitado del dataset.

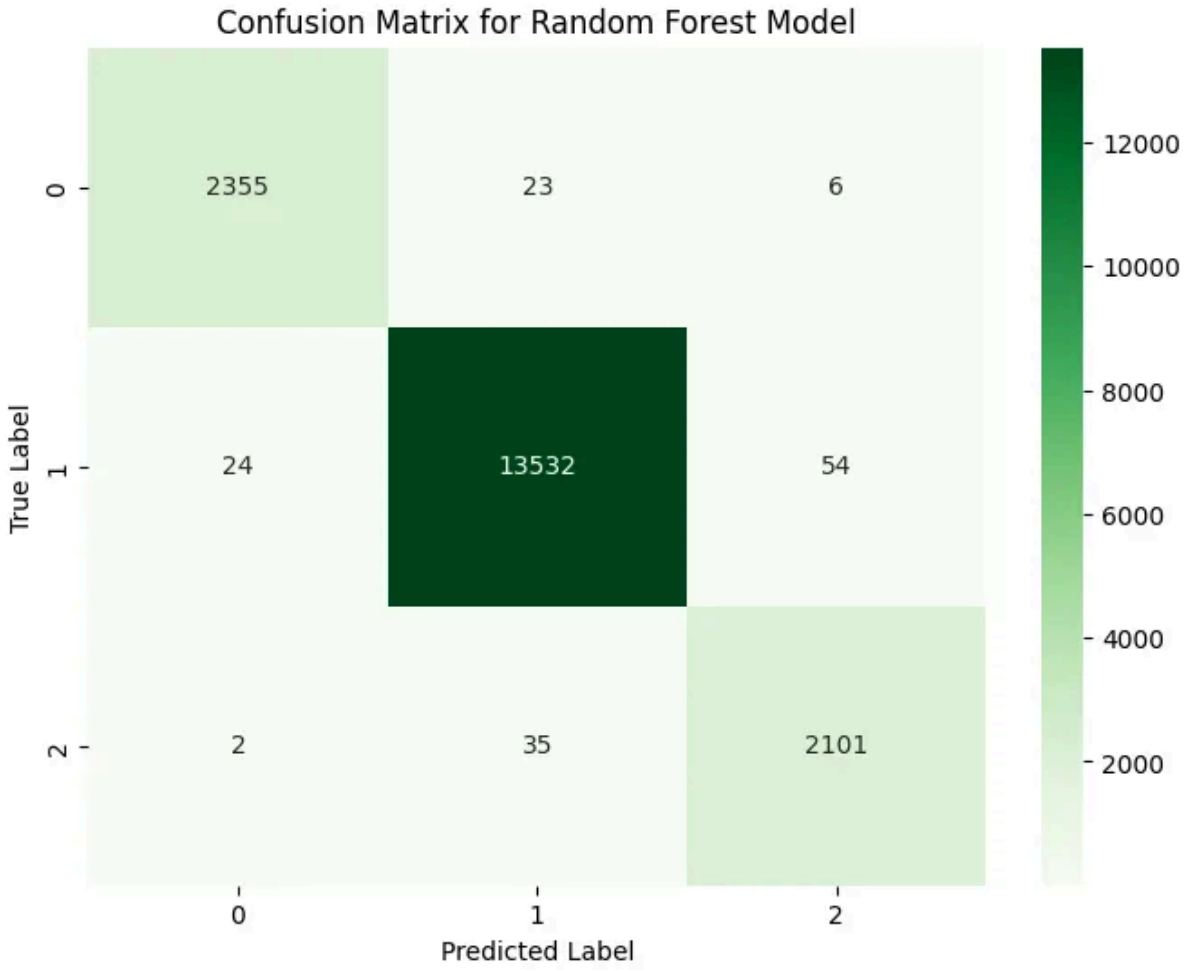
4.3. Análisis visual de resultados

- RF y XGB parecen perfectos.
- MLP y TabNet muestran patrones más realistas (algo de error, sacrificios entre clases), lo que curiosamente puede ser signo de menos alineación ciega con las reglas de augmentation.

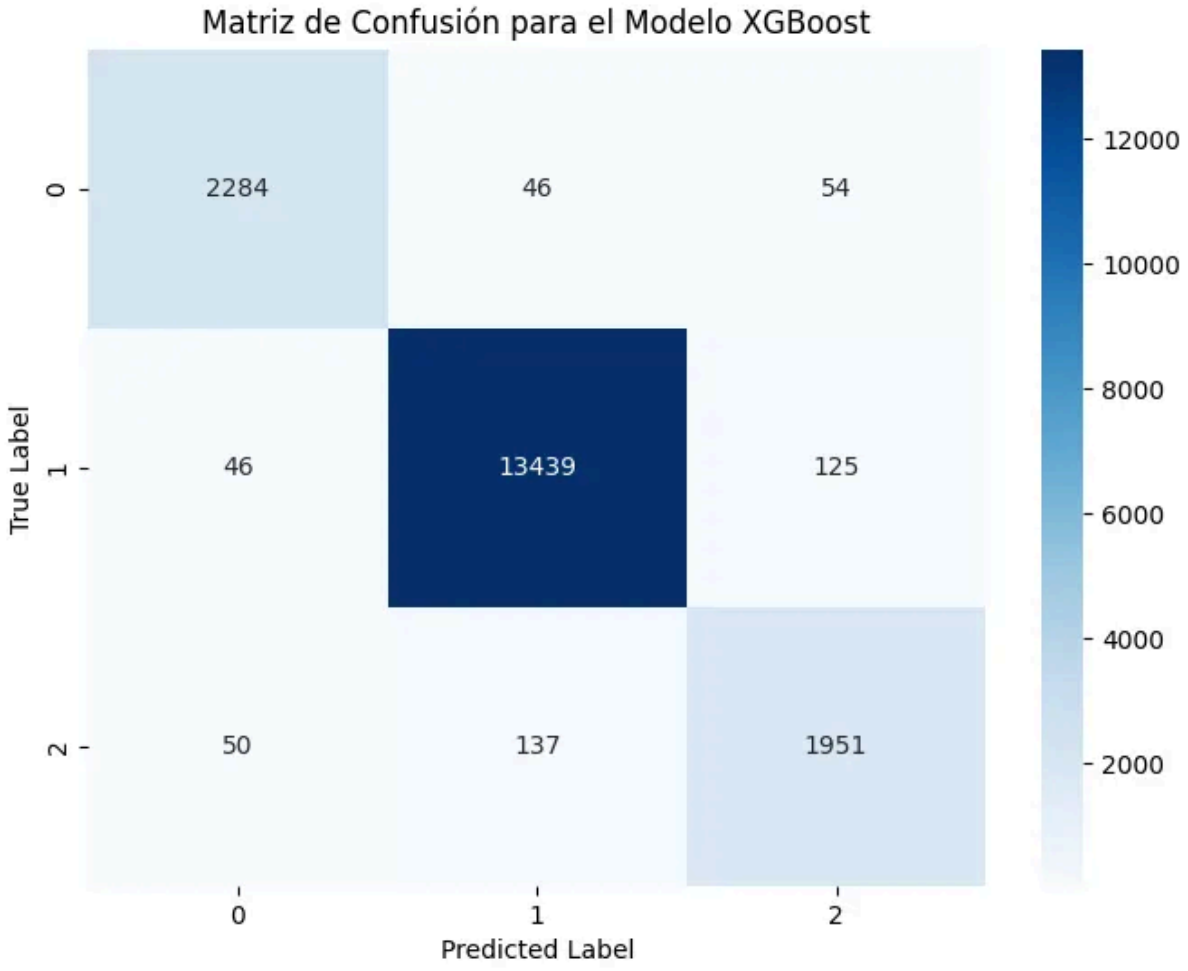
1. Regresión Logística



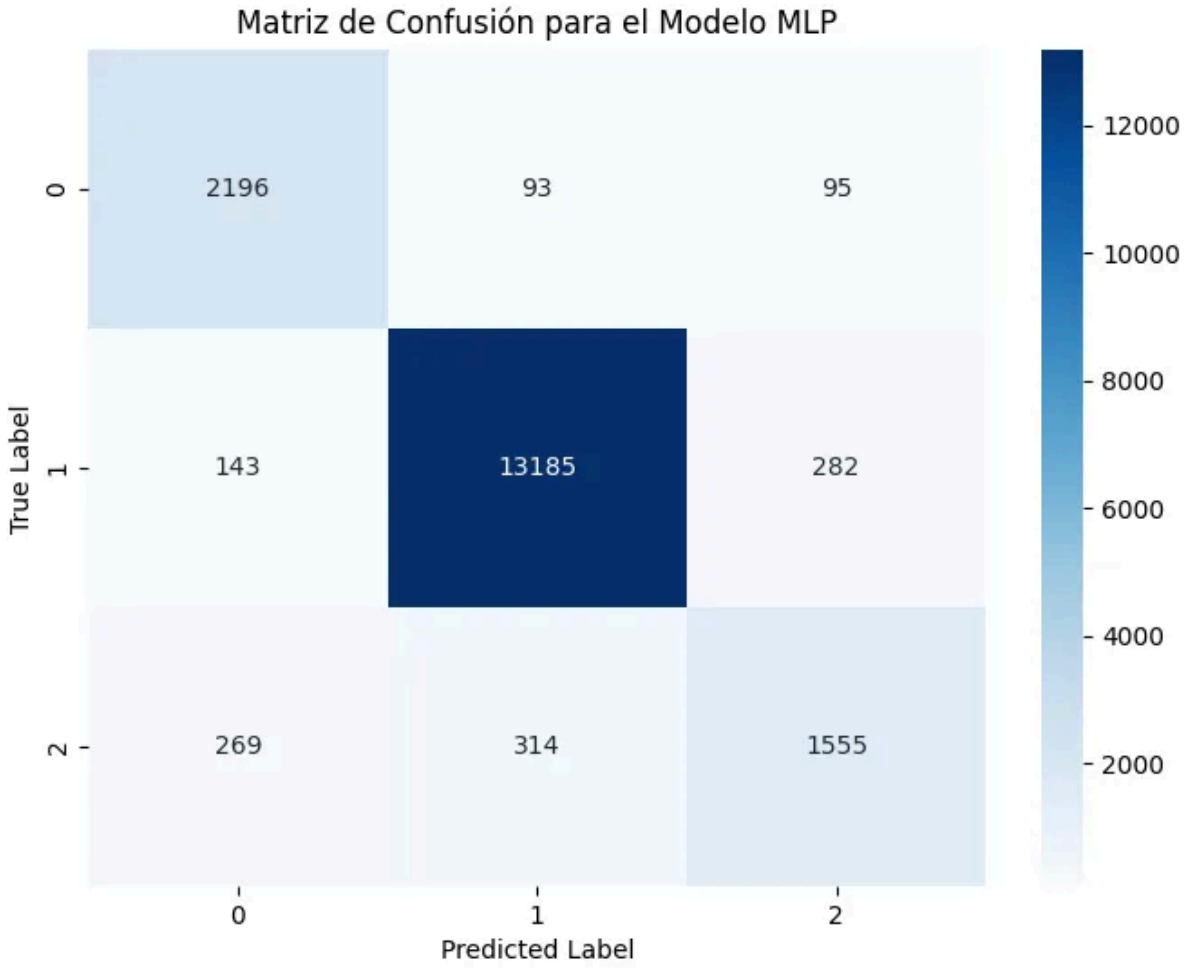
2. Random Forest



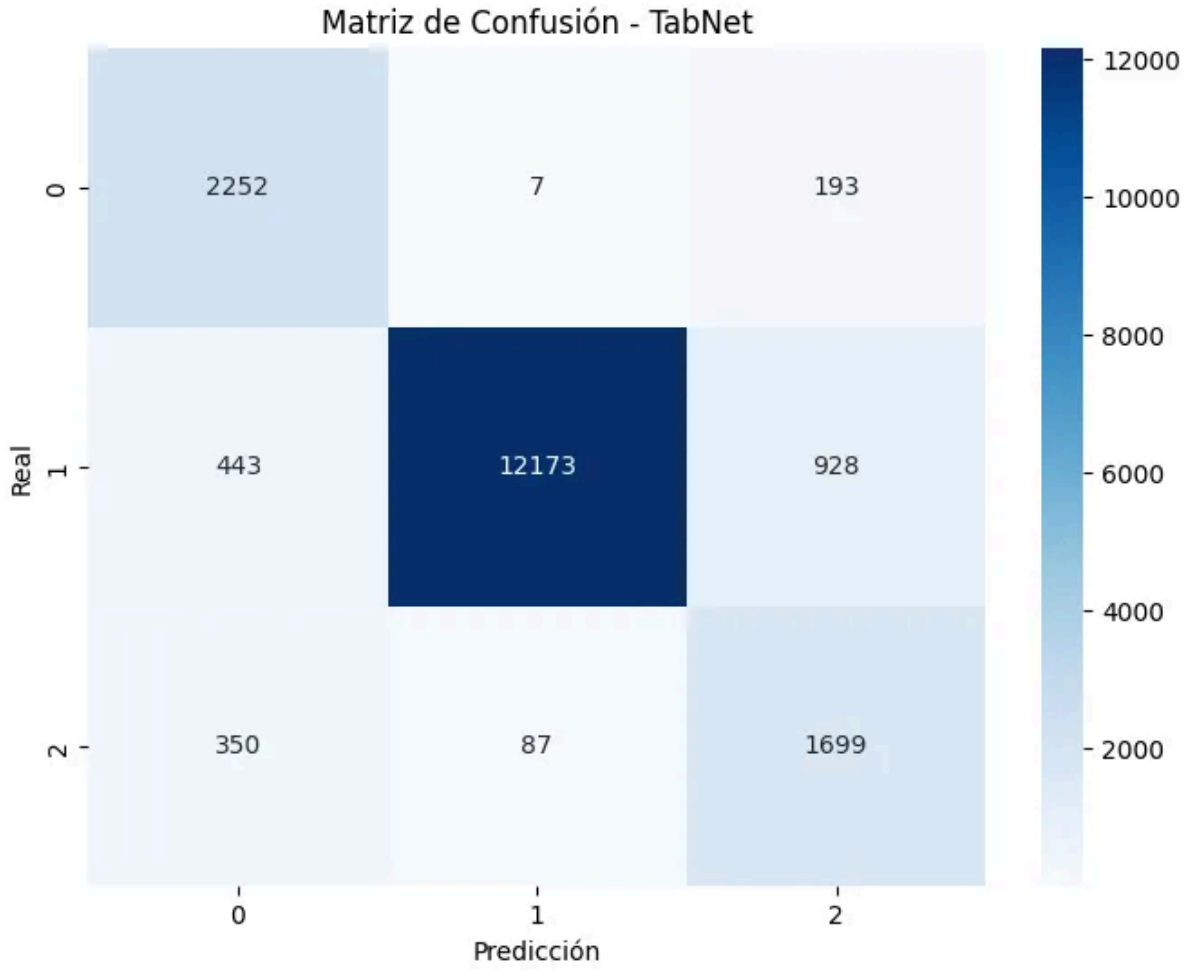
3. XGBoost


Matriz de Confusión para el Modelo XGBoost
True Label
Predicted Label
0
1
2
2284
46
54
46
13439
125
50
137
1951
12000
10000
8000
6000
4000
2000

4. MLP


Matriz de Confusión para el Modelo MLP
2196
93
95
143
13185
282
269
314
1555
True Label
Predicted Label
0
1
2
12000
10000
8000
6000
4000
2000

5. TabNet



4.4. Limitaciones de los modelos

1. Data augmentation casi determinista

El pipeline de augmentation inyecta reglas de negocio muy fuertes:

- Para cada animal, `augment_animal_sequence()` modifica producción con ruido y luego clipea según el estado
  - En Monitoreo: producción ≤ 25 L
  - Preñada: 20–50 L
  - Seca: producción = 0
- `balance_dataset()` repite esto para alcanzar ≥10,000 muestras por clase, pasando de 34 registros originales a ~30,000 por clase, (~90,000 en total).

2. Mismo animal en train y test

El dataset balanceado se construye por animal y luego se hace un `train_test_split` directo sobre filas: no se usa un split por grupo que mantenga cada animal en un solo conjunto.

Eso significa:

- Múltiples filas casi idénticas (misma vaca, mismas características con ruido leve) terminan repartidas entre train y test.
- El modelo ve instancias casi clonadas en entrenamiento y evaluación → los resultados en test no representan un escenario real “vacunas nuevas / hatos nuevos”.

Este es probablemente el factor más fuerte detrás de los números casi perfectos de RF y XGB.

3. Métricas optimistas y riesgo de distribution shift

Todo lo anterior implica que las métricas (especialmente de RF y XGB) son:

- Optimistas respecto a lo que verían en:
    - Nuevas vacas,
    - Otros hatos,
    - Cambios en el manejo, etc.
- Sensibles al hecho de que muchas muestras “nuevas” no seguirán tan estrictamente los rangos impuestos por el augmentation (p.ej. vacas enfermas, errores de medición, sensores).

4.5. Decisiones de aceptación del modelo

- **Modelo a priorizar para despliegue: XGBoost**
    - Tiene muy buen rendimiento (accuracy 0.9747, macro F1 0.95).
    - La brecha train/test es pequeña pero no ridícula (97.9 vs 97.5), lo que sugiere buena regularización dentro del dataset actual.
    - Es más robusto a ruido que RF y suele generalizar mejor en tabular si está bien tuneado.
- Random Forest
    - Mantenerlo como modelo de referencia o baseline fuerte, pero:
        - No lo tomaría como verdad absoluta por lo perfecto de sus métricas (F1 ~0.99) en un dataset con pocos datos.
    - Es excelente como benchmark interno y como modelo de respaldo si XGB falla.
- MLP y TabNet
    - A pesar de tener menos accuracy, pueden ser útiles como modelos alternos cuando existan más datos.
    - Hoy, no justifican su complejidad frente a XGB.
- Regresión Logística
    - Debe conservarse explícitamente como baseline, para:
        - Validar que no haya roturas graves (drift),

    - Explicar de manera simple a stakeholders cómo se comportan las features.

4.6. Conclusiones

Numéricamente, Random Forest y XGBoost parecen “casi perfectos” sobre el dataset augmentado, con XGBoost ligeramente por debajo en accuracy pero mucho mejor regularizado explícitamente.

La principal amenaza no es el overfitting clásico, sino:

- El leakage entre train y test por compartir vacas/augmentations.
- El hecho de que los modelos están aprendiendo reglas impuestas por el augmentation y que existen muy pocos datos para trabajar.

XGBoost se perfila como el mejor compromiso entre:

- Alto desempeño,
- Mecanismos claros para controlar overfitting (regularización + early stopping),
- Flexibilidad para recalibrar y actualizar en producción.

Después de tres iteraciones completas, incluyendo modelos lineales, métodos basados en árboles, enfoques temporales con LSTM y modelos tabulares avanzados (MLP, XGBoost, TabNet), se logró obtener un modelo con desempeño suficientemente robusto para implementación operativa.

Dado que el dataset ya fue depurado, balanceado, validado, y considerando que no se cuenta con más datos reales para nuevas iteraciones, los resultados obtenidos representan el punto máximo de rendimiento alcanzable con la información disponible.