XGBoost tiene una gran cantidad de hiperparámetros. La clave no es probar todos, sino entender cuáles "frenan" la capacidad del modelo de memorizar ruido.

Regularización L1 (Alpha) y L2 (Lambda): Diferencias entre penalizar los pesos de las hojas y cómo ayudan a mantener el modelo simple.

Complejidad del Árbol: El papel de max_depth (profundidad máxima) y min_child_weight (suma mínima de pesos en una hoja) para evitar que el modelo cree ramas basadas en muy pocos ejemplos.

Reducción de Varianza: El uso de subsample y colsample_bytree para introducir aleatoriedad y robustez, similar a cómo funciona Random Forest.

Cómo “frenar” el sobreajuste en XGBoost con pocos hiperparámetros clave

La teoría de XGBoost se basa en minimizar una función de pérdida más un término de **regularización del árbol** que penaliza árboles muy complejos (muchas hojas y pesos grandes) para mejorar la generalización (Wang et al., 2019; Contente & Prates, 2025; Ugirumurera et al., 2024). Los hiperparámetros que comentas son justamente los que controlan esa complejidad.

1. Regularización L1 (alpha) vs L2 (lambda)

En XGBoost se penaliza la **suma de cuadrados o valores absolutos de los pesos de las hojas**:

- **L2 (lambda)**: añade una penalización cuadrática a los pesos de las hojas, empujándolos hacia 0 sin anularlos del todo. Esto suaviza las predicciones y reduce la sensibilidad al ruido, de forma análoga al ridge en regresión (Wang et al., 2019; Contente & Prates, 2025).
- **L1 (alpha)**: penaliza la suma absoluta de los pesos; muchos pesos pequeños acaban exactamente en 0, lo que hace el árbol “más escaso” (menos hojas efectivas) (Wang et al., 2019; Contente & Prates, 2025).

En la práctica:

- Aumentar **lambda** → predicciones más suaves, menos explosivas ante outliers.
- Aumentar **alpha** → poda implícita de hojas poco informativas, lo que simplifica el modelo.

2. Complejidad del árbol: max_depth y min_child_weight

Estudios aplicados y guías empíricas destacan que **max_depth** y **min_child_weight** son de los hiperparámetros con mayor impacto en error y sobreajuste (Sibindi et al., 2022; Contente & Prates, 2025; Verma, 2024; Sakti et al., 2025):

- **max_depth**: limita cuántas particiones consecutivas puede hacer un árbol.
  - Profundidades grandes → muchas interacciones y alta varianza (capacidad de memorizar ruido).
  - Profundidades moderadas (p.ej. 3–8 en regresión/tablas estándar) suelen equilibrar bien sesgo y varianza (Sibindi et al., 2022; Contente & Prates, 2025; Sakti et al., 2025).
- **min_child_weight**: exige una **suma mínima de pesos** (equivalente a “cantidad efectiva de datos”) para crear una nueva hoja.
  - Valores altos impiden que el modelo cree ramas basadas en muy pocos ejemplos, lo que en estudios de tuning se ve como un regularizador clave para controlar sobreajuste (Sibindi et al., 2022; Contente & Prates, 2025; Verma, 2024; Sakti et al., 2025).

En trabajos recientes, gamma (mínima reducción de pérdida requerida para un split) también actúa como freno: splits con mejora marginal no se realizan (Sibindi et al., 2022; Verma, 2024; Sakti et al., 2025).

3. Subsample y colsample_bytree: reducción de varianza "tipo Random Forest"

XGBoost integra ideas de Random Forest al introducir aleatoriedad:

- **subsample**: porcentaje de filas usado para entrenar cada árbol. Valores <1 introducen variación entre árboles, reduciendo la correlación entre ellos y la **varianza del ensemble**, a costa de un ligero aumento de sesgo (Sibindi et al., 2022; Contente & Prates, 2025; Sakti et al., 2025).
- **colsample_bytree / bylevel / bynode**: porcentaje de columnas usado por árbol/nivel/nodo, análogo al mtry en Random Forest. Al usar solo un subconjunto de características en cada árbol, el modelo depende menos de un conjunto reducido de predictores y resulta más robusto a ruido y colinealidad (Sibindi et al., 2022; Contente & Prates, 2025; Ugirumurera et al., 2024; Sakti et al., 2025).

Trabajos empíricos muestran que combinar **subsample (≈0.6–0.8)** y **colsample (≈0.6–0.9)** reduce claramente el sobreajuste frente a entrenar siempre con todos los datos y todas las columnas (Sibindi et al., 2022; Contente & Prates, 2025; Ugirumurera et al., 2024; Sakti et al., 2025). En un estudio sobre tráfico y otro sobre duración de estudios universitarios, los autores destacan subsample y colsample como hiperparámetros centrales para evitar sobreajuste en XGBoost (Contente & Prates, 2025; Sakti et al., 2025).

4. Visión unificada: qué controlar primero

Síntesis desde la literatura de tuning de XGBoost:

Objetivo	Hiperparámetros más influyentes	Mecanismo	Citaciones
Limitar complejidad del árbol	max_depth, min_child_weight, gamma	Poda estructural: impedir splits con pocos datos o poca ganancia	(Sibindi et al., 2022; Contente & Prates, 2025; Verma, 2024; Sakti et al., 2025)
Suavizar predicciones	lambda (L2), alpha (L1)	Penalizar pesos grandes; hojas "pequeñas" o nulas	(Wang et al., 2019; Contente & Prates, 2025; Ugirumurera et al., 2024)
Bajar varianza del ensemble	subsample, colsample_by*	Bagging parcial de filas y columnas, estilo Random Forest	(Sibindi et al., 2022; Contente & Prates, 2025; Ugirumurera et al., 2024; Sakti et al., 2025)

FIGURE 1 Hiperparámetros de XGBoost que más controlan el sobreajuste.

En muestras pequeñas o ruidosas, la evidencia sugiere priorizar:

1. limitar **max_depth** y subir **min_child_weight**,
2. aplicar **subsample/colsample** < 1,
3. reforzar **lambda/alpha**,
   antes de buscar configuraciones más exóticas. Esto mantiene el modelo expresivo, pero con capacidad controlada para "memorizar" ruido.

These papers were sourced and synthesized using Consensus, an AI-powered search engine for research. Try it at https://consensus.app

References

Sibindi, R., Mwangi, R., & Waititu, A. (2022). A boosting ensemble learning based hybrid light gradient boosting machine and extreme gradient boosting model for predicting house prices. *Engineering Reports*, 5. https://doi.org/10.1002/eng2.12599

Wang, C., Deng, C., & Wang, S. (2019). Imbalance-XGBoost: Leveraging Weighted and Focal Losses for Binary Label-Imbalanced Classification with XGBoost. *ArXiv*, abs/1908.01672. https://doi.org/10.1016/j.patrec.2020.05.035

Contente, J., & Prates, P. (2025). Predicting edge cracking in sheet metal forming: evaluating machine learning models and data transformations. *The International Journal of Advanced Manufacturing Technology*, 138, 3089 - 3107. https://doi.org/10.1007/s00170-025-15721-6

Ugirumurera, J., Bensen, E., Severino, J., & Sanyal, J. (2024). Addressing bias in bagging and boosting regression models. *Scientific Reports*, 14. https://doi.org/10.1038/s41598-024-68907-5

Verma, V. (2024). Exploring Key XGBoost Hyperparameters: A Study on Optimal Search Spaces and Practical Recommendations for Regression and Classification. *International Journal of All Research Education and Scientific Methods*. https://doi.org/10.56025/ijaresm.2024.1210243259

Sakti, M., Jailani, J., Retnawati, H., Hidayati, K., Waryanto, N., Ibrahim, Z., Khoirunnisa, A., Wibowo, F., Berlian, M., & Batubara, A. (2025). THE EFFECT OF SAMPLE SIZE ON THE STABILITY OF XGBOOST MODEL PERFORMANCE IN PREDICTING STUDENT STUDY PERIOD. *BAREKENG: Jurnal Ilmu Matematika dan Terapan*. https://doi.org/10.30598/barekengvol19iss4pp2679-2692