Explicación del algoritmo XGBoost

Atlas de Riesgos para la Nutrición de la Niñez en México

Maestro codificador: Dr.(C) Antonio Villalpando-Acuña

Octubre de 2024

En este estudio, el modelo XGBoost se entrena para un problema de regresión con varios hiperparámetros clave que controlan el comportamiento del modelo y su capacidad de generalización, es decir, de hacer inferencia estadísticamente válida. A continuación, se explican en detalle los hiperparámetros utilizados y su impacto en el rendimiento del modelo y explica la inspección que se llevó a cabo para evitar el efecto umbral.

1 Hiperparámetros

1.1 objective = "reg:squarederror"

Este hiperparámetro define el tipo de problema que el modelo va a resolver. En este caso, se utiliza el error cuadrático medio (MSE) como la función objetivo. El MSE mide el promedio de los errores al cuadrado entre las predicciones y los valores reales, penalizando más los errores grandes.

El MSE se minimiza durante el entrenamiento para hacer que las predicciones del modelo sean lo más cercanas posible a los valores reales de la variable dependiente. Al elegir `"reg:squarederror"`, el modelo está optimizado específicamente para problemas de regresión continua, como en el caso del peso para la talla, la talla para la edad y el peso para la edad.

1.2 eval_metric = "rmse"

Este hiperparámetro define la métrica de evaluación que el modelo utiliza durante el proceso de entrenamiento para medir su desempeño en cada iteración. En este caso, se utiliza el error cuadrático medio de la raíz (RMSE), que es simplemente la raíz cuadrada del MSE. El RMSE se prefiere en muchos casos de regresión porque proporciona una interpretación directa en las mismas unidades que la variable dependiente.

El RMSE es más sensible a grandes desviaciones entre los valores predichos y los reales, lo que significa que el modelo intentará minimizar los errores grandes, mejorando la precisión en observaciones extremas. Esto es particularmente relevante en el contexto de predecir variables de salud como el peso para la talla.

1.3 nrounds = 100

Este hiperparámetro define el número de iteraciones o "boosting rounds" que el modelo va a realizar. En cada una de estas rondas, XGBoost construye un nuevo árbol de decisión que corrige los errores cometidos por los árboles anteriores. El valor de `nrounds = 100` indica que el modelo se entrenará en 100 iteraciones, lo que puede impactar significativamente la capacidad del modelo para aprender patrones complejos.

Demasiadas iteraciones: si `nrounds` es demasiado alto, existe el riesgo de sobreajuste (overfitting), donde el modelo se ajusta demasiado bien a los datos de entrenamiento y pierde capacidad para generalizar a nuevos datos.

Muy pocas iteraciones: si `nrounds` es demasiado bajo, el modelo puede quedar subajustado (underfitting), lo que significa que no logra capturar los patrones subyacentes en los datos.

El valor adecuado de `nrounds` generalmente se elige mediante validación cruzada o en función del early stopping, una técnica que detiene el entrenamiento cuando el modelo deja de mejorar después de un cierto número de iteraciones. En este estudio se eligió realizar el proceso de validación cruzada a priori y verificar el ajuste modificando los hiperparámetros manualmente, lo que se explica más adelante en este documento.

2 Funcionamiento de XGBoost

XGBoost es una implementación optimizada del algoritmo de boosting basado en árboles de decisión, donde varios árboles "débiles" se entrenan de forma secuencial. Cada árbol siguiente se ajusta a los errores cometidos por el árbol anterior, con el objetivo de minimizar una función de pérdida. Los puntos clave son:

2.1 Función de pérdida

La función de pérdida mide qué tan bien el modelo ajusta los datos. En problemas de regresión, XGBoost generalmente minimiza el error cuadrático medio (MSE), mientras que en clasificación puede minimizar log loss. La función objetivo $L(\theta)$ del modelo se compone de la pérdida de predicción y una penalización por la complejidad del modelo:

$$L(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{1}$$

Donde:

- $l(y_i, \hat{y}_i)$ es la función de pérdida, que mide la diferencia entre el valor real y_i y el valor predicho $\hat{y}_i$.
- $\Omega(f_k)$ es el término de regularización que penaliza la complejidad de los árboles.

2.2 Boosting con gradiente

Cada nuevo árbol se entrena para minimizar los residuos o errores del árbol anterior. XGBoost utiliza una aproximación de segundo orden al minimizar la función objetivo, lo que implica que se consideran tanto el gradiente (primera derivada) como el Hessiano (segunda derivada) de la función de pérdida. Esto le permite ajustar el modelo con mayor precisión.

En cada iteración t, la función de pérdida se aproxima utilizando una expansión de Taylor de segundo orden (algoritmo que es perfecto, por analogía, para aproximar los cambios en expresiones de tercer grado como la evolución del IMC a lo largo del tiempo en menores de edad):

$$L^{(t)} \approx \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \Omega(f_t) \tag{2}$$

Donde:

- $g_i = \frac{\partial l(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ es el gradiente.
- $h_i = \frac{\partial^2 l(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$ es el Hessiano.
- $f_t(x_i)$ es la predicción del nuevo árbol en la iteración t.

3 Ajuste de hiperparámetros y ronda de inspección para evitar el efecto umbral

Durante el proceso de ajuste, se realizó una ronda de inspección para evitar el efecto umbral. El efecto umbral ocurre cuando un modelo de aprendizaje automático se ajusta excesivamente a ciertos valores críticos o específicos de los datos de entrenamiento, lo que resulta en una pérdida de capacidad para generalizar a nuevos datos. Es decir, el modelo se vuelve "sensible" a umbrales particulares de las variables, lo que puede llevar a un mal rendimiento en observaciones fuera de esos rangos o valores.

Este problema es especialmente relevante en modelos como XGBoost en los que las decisiones se basan en la creación de múltiples árboles de decisión que dividen los datos en distintos grupos. Si el modelo se ajusta demasiado a ciertos puntos umbrales, como divisiones específicas dentro de los árboles, puede sobreajustar esos datos y perder capacidad predictiva en datos de prueba o en situaciones nuevas. Los siguientes hiperparámetros se ajustaron como parte de esta inspección:

- `max_depth = 6`: Controla la profundidad máxima de cada árbol. Un valor de 6 limita la complejidad de los árboles, evitando el sobreajuste.

- `eta = 0.1`: El learning rate se estableció en 0.1, lo que reduce la magnitud de las actualizaciones en cada iteración.
- `lambda = 1` y `alpha = 0.1`: Controlan la regularización L2 (Ridge) y L1 (Lasso).
- `early_stopping_rounds = 10`: Detiene el entrenamiento si no hay mejora después de 10 iteraciones consecutivas.

4 Conclusión sobre la ronda de inspección

Después de realizar la ronda de inspección, no se detectó sobreajuste significativo, por lo que se decidió utilizar los valores por defecto de los hiperparámetros en las iteraciones finales. Esta inspección verifica que el modelo mantiene un buen equilibrio entre flexibilidad y capacidad de generalización.