CAPÍTULO 3

ALGORITMO GRADIENT BOOSTING: FUNDAMENTOS MATEMÁTICOS Y VENTAJAS DE CATBOOST EN MACHINE LEARNING

https://doi.org/10.22533/at.ed.316122508043

Fecha de aceptación: 02/05/2025

Carlos Alberto Peña Miranda
https://orcid.org/0000-0002-4339-4615

Jesus Adalberto Zelaya Contreras
https://orcid.org/0009-0003-4767-6366

Elizabeth Cosi Cruz
https://orcid.org/0000-0002-0255-7705

RESUMEN: Este artículo presenta una visión general sobre el algoritmo Gradient Boosting, enfocándose especialmente en CatBoost, una de las variantes más destacadas. Se explica el principio fundamental de Gradient Boosting, un método iterativo que combina modelos débiles (como árboles de decisión) para minimizar una función de pérdida mediante el gradiente negativo. Además, se analiza el fundamento matemático detrás de este algoritmo y su implementación en CatBoost, que destaca por su capacidad para manejar eficientemente variables categóricas sin necesidad de transformaciones complicadas. Finalmente, se discuten las principales ventajas de CatBoost, como su velocidad de entrenamiento, su manejo de grandes volúmenes de datos y su capacidad para reducir el sobreajuste, lo que lo convierte en una herramienta poderosa para diversas aplicaciones de machine learning.

PALABRAS CLAVE: Gradient Boosting, CatBoost, Variables Categóricas

GRADIENT BOOSTING ALGORITHM: MATHEMATICAL FOUNDATIONS AND ADVANTAGES OF CATBOOST IN MACHINE LEARNING

ABSTRACT: This article provides an overview of the Gradient Boosting algorithm, with a particular focus on CatBoost, one of its most prominent variants. It explains the fundamental principle of Gradient Boosting, an iterative method that combines weak models (such as decision trees) to minimize a loss function through negative gradient descent. Additionally, the mathematical foundation behind this algorithm and its implementation in CatBoost are analyzed, highlighting its ability to efficiently handle categorical variables without requiring complex transformations. Finally, the key advantages of CatBoost are discussed, including its training speed, handling of large datasets, and ability to reduce overfitting, making it a powerful tool for diverse machine learning applications.

KEYWORDS: Gradient Boosting, CatBoost, Categorical Variables

INTRODUCCIÓN

El *Gradient Boosting* es una de las técnicas más utilizadas en aprendizaje automático para problemas de clasificación y regresión. Este método construye modelos de manera aditiva, combinando múltiples árboles de decisión para minimizar una función de pérdida de forma iterativa. Sin embargo, a pesar de su efectividad, presenta ciertos desafíos, como el manejo adecuado de variables categóricas y la eficiencia computacional.

CatBoost (Categorical Boosting) es un algoritmo desarrollado para abordar estas limitaciones, optimizando el tratamiento de variables categóricas y mejorando la estabilidad del modelo. A diferencia de otros métodos de boosting como XGBoost y LightGBM, CatBoost introduce una novedosa estrategia de *permutación ordenada* para la codificación de variables categóricas, evitando la fuga de información (*data leakage*) y proporcionando estimaciones más precisas.

En este artículo, exploraremos los fundamentos matemáticos del *Gradient Boosting*, los principios clave detrás de CatBoost, su implementación y sus ventajas con respecto a otros algoritmos de boosting. A través de este análisis, se demostrará cómo CatBoost mejora la eficiencia en el entrenamiento y optimiza el rendimiento en tareas de predicción.

INTRODUCCIÓN AL GRADIENT BOOSTING

El *Gradient Boosting* es un método de aprendizaje supervisado que construye modelos de manera secuencial, ajustando cada nueva iteración para minimizar una función de pérdida. Este enfoque se basa en la idea de combinar múltiples modelos débiles (generalmente árboles de decisión) para crear un modelo fuerte que mejore la precisión en la predicción. A continuación, se presenta una descripción formal del algoritmo y su fundamento matemático.

DESCRIPCIÓN FORMAL DEL ALGORITMO Y FUNDAMENTO MATEMÁTICO.

El algoritmo de *Gradient Boosting* optimiza una función de pérdida mediante un enfoque iterativo, en el cual cada nuevo modelo aprende a corregir los errores cometidos por la suma de los modelos anteriores. Formalmente, dado un conjunto de datos de entrenamiento $\{(x_i,y_i)\}_{i=1}^{n}$, donde x_i representa las características e y_i la variable objetivo, el modelo se construye de la siguiente manera:

1. Se inicializa el modelo con un predictor constante:

$$F_0(x) = \sum_{i=1}^{n} L(y_i, c)$$

donde L(y, F(x)) es la función de pérdida. Aquí, c representa un valor escalar óptimo que minimiza la función de pérdida en la primera etapa del modelo. La elección de c depende del tipo de problema: en regresión con pérdida cuadrática suele ser la media de los valores objetivo, mientras que en clasificación con pérdida logarítmica suele ser la moda de las clases. Este predictor inicial proporciona un punto de partida antes de que el modelo comience a iterar y mejorar sus predicciones.

2. Para cada iteración m = 1,…, M:

- Se calcula el residuo o pseudo-residuo, que representa la dirección del gradiente negativo de la función de pérdida:

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x_i)}$$

donde $f_{m-1}(x)$ es el modelo construido hasta la iteración m –1. Se utiliza esta notación porque el objetivo del nuevo modelo es corregir los errores del modelo actual, por lo que el cálculo del gradiente se evalúa en la predicción del modelo en su estado anterior.

En este paso, el residuo se toma como el gradiente negativo de la función de pérdida porque en optimización, el gradiente indica la dirección de mayor incremento de la función. Para minimizar la pérdida, el modelo debe moverse en la dirección opuesta, es decir, en el gradiente negativo. Esto permite que cada iteración corrija los errores acumulados y mejore la precisión del modelo.

- Se ajusta un nuevo modelo débil $h_m(x)$, generalmente un árbol de decisión, que predice los residuos $r_i^{(m)}$. Este modelo débil tiene la tarea de aproximar los residuos actuales y suele ser un árbol de decisión poco profundo para evitar el sobreajuste. Su función es detectar patrones en los errores del modelo anterior y corregirlos en la siguiente actualización.

Se encuentra el coeficiente de actualización γ resolviendo:

$$\gamma_m = \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

donde γ_m es un factor de ajuste que controla cuánto debe contribuir el nuevo modelo $h_m(x)$ a la actualización del modelo global. Este coeficiente se encuentra minimizando la función de pérdida, ya que el objetivo es garantizar que la corrección aplicada por $h_m(x)$ reduzca los errores de la mejor manera posible. En otras palabras, γ_m es el peso óptimo que equilibra la influencia del nuevo modelo débil en la predicción final.

- Se actualiza el modelo:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

lo que significa que el modelo global se ajusta agregando el modelo débil $h_m(x)$ multiplicado por su peso óptimo . Este proceso iterativo continúa hasta alcanzar el número máximo de iteraciones M o hasta que la mejora en la función de pérdida sea insignificante.

3. El modelo final se obtiene tras M iteraciones:

$$F_m(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x_i)$$

Este proceso permite que el modelo aprenda de manera secuencial, ajustando cada nueva iteración para minimizar la función de pérdida en la dirección del gradiente negativo, lo que da origen al nombre de Gradient Boosting.

EJEMPLO 1.

Supongamos que tenemos un conjunto de datos con tres observaciones y queremos entrenar un modelo de Gradient Boosting para efectos prácticos, utilizando tres iteraciones. Emplearemos la función de pérdida cuadrática media (MSE):

$$L(y, f(x)) = \frac{1}{2}(y - F(x))^2$$

INICIALIZACIÓN:

El modelo inicial $f_0(x)$ se obtiene minimizando la función de pérdida. Para MSE, la mejor constante c es la media de los valores objetivo:

$$F_0(x) = \frac{1}{n} \sum_{i=1}^{n} y_i$$

Supongamos que nuestros datos son:

$$\{(x_1, y_1) = (1, 3), (x_2, y_2) = (2, 5), (x_3, y_3) = (3, 7)\}$$

Entonces, la inicialización del modelo es:

$$F_0(x) = \frac{3+5+7}{3} = 5$$

ITERACIÓN 1:

1. Cálculo de los residuos:

Usamos la derivada de la función de pérdida respecto a F(x). Para una función de pérdida cuadrática:

$$L(y, f(x)) = \frac{1}{2}(y - F(x))^2$$

Calculamos la derivada parcial con respecto a F(x):

$$\frac{\partial L(y, F(x))}{\partial F(x)} = \frac{\partial}{\partial F(x)}\left[\frac{1}{2}(y - F(x))^2\right]$$

Aplicamos la regla de la cadena:

$$\frac{\partial L(y, F(x))}{\partial F(x)} = \frac{1}{2} \cdot 2\,(y - F(x))(-1) = -(y - F(x))$$

Así, los residuos en la primera iteración se calculan como:

$$r_i^{(m)} = -\left[\frac{\partial L(y, F(x))}{\partial F(x)}\right] = y_i - F_0(x_i)$$

Calculamos los residuos para cada punto:

$$r_1^{(1)} = 3 - 5 = -2,\ r_2^{(1)} = 5 - 5 = 0,\ r_3^{(1)} = 7 - 5 = 2$$

2. Ajuste del modelo débil $h_1(x)$: En lugar de predecir los residuos con su media, utilizamos la mediana por subgrupos para hacerlo más robusto.

Dividimos los datos en subgrupos según x y calculamos la mediana de los residuos en cada uno:

- Para el grupo 1, con valores de residuos {-2, 0}, la mediana es:

  mediana({-2, 0}) = -1

- Para el grupo 2, con el único valor de residuo {2}, la mediana es:

  mediana({2}) = 2

Así, el modelo débil se define como:

$$h_1(x) = \{-1,\ si\ x \in \{1, 2\}\ 2,\ \ si\ x = 3$$

3. Cálculo del coeficiente de actualización : En regresión cuadrática, se puede demostrar que:

$$\gamma_m = \frac{\sum_{i=1}^{n} r_i^{(m)} h_m(x_i)}{\sum_{i=1}^{n} \left[h_m(x_i)\right]^2}$$

Esto es para encontrar el valor óptimo de , debemos minimizar la función de pérdida:

$$\gamma_m = \sum_{i=1}^{n} L(y_i,\ F_{m-1}(x_i) + \gamma h_m(x_i))$$

Dado que estamos usando la pérdida cuadrática media (MSE), la función de pérdida es:

$$L(y_i,\ f(x_i)) = \frac{1}{2}(y_i - F(x_i))^2$$

Sustituyendo $F_m(x_i) = F_{m-1}(x_i) - \gamma_m h_m(x_i)$, la función de pérdida se convierte en:

$$L\left(y_i, Fm\left(x_i\right)\right) = \frac{1}{2}(y_i - F_{m-1}(x_i) + \gamma_m h_m(x_i))^2$$

Diferenciamos con respecto a y_m:

$$\frac{\partial}{\partial \gamma_m} \sum_{i=1}^{n} \frac{1}{2}(y_i - (F_{m-1}(x_i) + \gamma_m h_m(x_i)))^2 = 0$$

Aplicamos la regla de la cadena:

$$\sum_{i=1}^{n} \frac{1}{2}(y_i - (F_{m-1}(x_i) + \gamma_m h_m(x_i)))(-h_m(x_i)) = 0$$

Reescribimos en términos del residuo $r_i^{(m)} = y_i - F_{m-1}(x_i)$: :

$$\sum_{i=1}^{n} \frac{1}{2}(r_i^{(m)} - \gamma_m h_m(x_i)))(-h_m(x_i)) = 0$$

Distribuimos:

$$\sum_{i=1}^{n} \frac{1}{2} r_i^{(m)} h_m(x_i) + \gamma_m \sum_{i=1}^{n} \left[h_m(x_i)\right]^2 = 0$$

Despejamos y_m :

$$\gamma_m = \frac{\sum_{i=1}^{n} r_i^{(m)} h_m(x_i)}{\sum_{i=1}^{n} \left[h_m(x_i)\right]^2}$$

Ahora procedemos a realizar el cálculo:

$$\gamma_1 = \frac{r_1^{(1)} h_1(x_1) + r_2^{(1)} h_1(x_2) + r_3^{(1)} h_1(x_3)}{\left[h_1(x_1)\right]^2 + \left[h_1(x_2)\right]^2 + \left[h_1(x_3)\right]^2} = \frac{(-2)(-1)+(0)(-1)+(2)(2)}{(-1)^2+(-1)^2+(2)^2}$$

Por lo tanto, tenemos $y_1 = 1$.

4. Actualización del modelo:

$$F_1(x) = F_0(x) + \gamma_1 h_1(x)$$

Reemplazando los valores de $F_0(x), \gamma_1$ y $h_1(x)$ obtenemos

$$F_1(x) = \{4, \ si\, x \in \{1, 2\}\ 7, \ \ si\, x = 3$$

ITERACIÓN 2:

1. Cálculo de los residuos:

Usamos la fórmula general:

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x_i)}$$

Para una función de pérdida cuadrática, como vimos en la iteración 1, tenemos:

$$r_i^{(2)} = y_i - F_1(x_i)$$

Evaluamos para cada punto:

$$r_1^{(2)} = 3-4 = -2,\ r_2^{(2)} = 5-4 = 1,\ r_3^{(2)} = 7-7 = 0$$

2. Ajuste del modelo débil $h_2(x)$

Suponemos que el nuevo modelo predice los residuos mediante la mediana de los valores en cada grupo:

- Para x = 2, el residuo es {1}, por lo que la mediana es 1.
- Para x {1,3}, los residuos son {-1,0}, cuya mediana es -0.5.

Entonces, el modelo débil queda definido como:

$$h_2(x) = \{-0.5,\ si\ x \in \{1, 3\}\ 1,\ \ si\ x = 2$$

3. Cálculo de y_2:

$$\gamma_2 = \frac{r_1^{(2)}h_2(x_1)+r_2^{(2)}h_2(x_2)+r_3^{(2)}h_2(x_3)}{\left[h_2(x_1)\right]^2+\left[h_2(x_2)\right]^2+\left[h_2(x_3)\right]^2} = \frac{(-1)(-0.5)+(1)(1)+(0)(-0.5)}{(-0.5)^2+(1)^2+(-0.5)^2}$$

Por lo tanto, tenemos y_2= 1.

4. Actualización del modelo:

$$F_2(x) = F_1(x) + \gamma_2 h_2(x)$$

Reemplazando los valores de $F_1(x)$, γ_2 y $h_2(x)$obtenemos

$$F_2(x) = \begin{cases} 3.5, & \text{si } x = 1 \\ 5, & \text{si } x = 2 \\ 6.5, & \text{si } x = 3 \end{cases}$$

ITERACIÓN 3:

1. Cálculo de los residuos:

Usamos la fórmula general:

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x_i)}$$

Para una función de pérdida cuadrática, como vimos en la iteración 1, tenemos:

$$r_i^{(3)} = y_i - F_2(x_i)$$

Evaluamos para cada punto:

$$r_1^{(3)} = 3 - 3.5 = -0.5,\ r_2^{(3)} = 5 - 5 = 0,\ r_3^{(3)} = 7 - 6.5 = 0.5$$

2. Ajuste del modelo débil $h_3(x)$

Suponemos que el nuevo modelo predice los residuos mediante la mediana de los valores en cada grupo:

- Para x{1,2}, los residuos son {-0.5,0}, cuya mediana es –0.25
- Para x = 3, el residuo es {0.5}, por lo que la mediana es 0.25.

Entonces, el modelo débil queda definido como:

$$h_3(x) = \{-0.25,\ \ si\ x \in \{1,\ 2\}\ 0.25,\ \ si\ x = 3$$

3. Cálculo de y_3 :

$$\gamma_3 = \frac{r_1^{(3)}h_3(x_1)+r_2^{(3)}h_3(x_2)+r_3^{(3)}h_3(x_3)}{[h_3(x_1)]^2+[h_3(x_2)]^2+[h_3(x_3)]^2} = \frac{(-0.5)(-0.25)+(0)(-0.25)+(0.5)(-0.25)}{(-0.25)^2+(-0.25)^2+(0.25)^2}$$

Por lo tanto, tenemos $y_3 = 1.3$

4. Actualización del modelo:

$$F_3(x) = F_2(x) + \gamma_3 h_3(x)$$

Por lo tanto, el modelo final después de M iteraciones, para nuestro caso particular con M = 3:

$$F_3(x) = \begin{cases} 3.175, & \text{si } x = 1 \\ 4.675, & \text{si } x = 2 \\ 6.825, & \text{si } x = 3 \end{cases}$$

Este ejemplo ilustra el proceso iterativo del Gradient Boosting. En la práctica, el modelo débil $h_m(x)$ suele ser un árbol más sofisticado, lo que permite mejorar la predicción en cada iteración. Como resultado, el modelo final es una combinación ponderada de estos modelos débiles, donde la cantidad de iteraciones y la configuración de los hiperparámetros determinan su desempeño final.

Este marco del *Gradient Boosting* es fundamental para nosotros, ya que sobre él se basa el algoritmo *CatBoost.*

FUNDAMENTO MATEMÁTICO DE CATBOOST

El algoritmo *CatBoos*t surge como una solución a los desafíos presentes en el manejo de variables categóricas dentro de los métodos de boosting. Propuesto por Prokhorenkova (2019), este método introduce estrategias innovadoras para reducir el sesgo introducido por la asignación de valores a categorías raras y mejorar la eficiencia en el entrenamiento.

A diferencia de otros enfoques tradicionales como *XGBoost* o *LightGBM*, *CatBoost* emplea una técnica de permutación ordenada para calcular las estadísticas de las características categóricas, evitando la fuga de información y proporcionando estimaciones más robustas. A continuación, se presenta el fundamento matemático que sustenta este algoritmo.

Dado un conjunto de $D = \{(x_i, y_i)\}_{i=1}^{n}$, , donde es un vector de características e y_i es la etiqueta correspondiente, el objetivo del Gradient Boosting es aprender una función F(x) que minimice una función de pérdida L(y, F(x)). El modelo se construye de manera aditiva, donde en cada iteración m se añade un nuevo modelo débil $h_m(x)$ para corregir los errores residuales de la iteración anterior. Formalmente, el modelo en la iteración m se expresa como:

$$F_m(x) = F_{m-1}(x) + \nu h_m(x)$$

La cual tiene una forma similar a la vista en la sección anterior, donde:

- $f_{m-1}(x)$ es el modelo en la iteración anterior.
- $h_m(x)$ es el modelo débil añadido en la iteración m.
- $_V$es la tasa de aprendizaje, un hiperparámetro que controla la contribución de cada modelo débil.

En cada iteración, el modelo débil $h_m(x)$se ajusta para minimizar la función de pérdida $L(x, F_{m-1}(x) + h_m(x))$. Esto se realiza utilizando el gradiente de la función de pérdida con respecto a $f_{m-1}(x)$.En particular, el gradiente $g_m(x)$se calcula como:

$$g_m(x) = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x_i)}$$

El modelo débil $h_m^{(x)}$ se entrena para aproximar el gradiente negativo $-g_m(x)$, lo que permite corregir los errores residuales de la iteración anterior.

FUNCIÓN DE PÉRDIDA LOGARÍTMICA

En problemas de clasificación, una función de pérdida comúnmente utilizada es la *pérdida logarítmica* (log loss), definida como:

$$L(y, F(x)) = -[\, y \ln(p) + (1 - y) \ln(1 - p) \,],$$

donde $p = \alpha(F(x))$ es la probabilidad predicha de que la etiqueta y sea 1, α es la función sigmoide y el logaritmo aplicado es el logaritmo natural (base e). En problemas de regresión, se utiliza comúnmente la *pérdida cuadrática*:

$$L(y, f(x)) = \frac{1}{2}(y - F(x))^2$$

El objetivo del Gradient Boosting es minimizar la función de pérdida L(y, F(x)) sobre el conjunto de datos . Esto se logra mediante la optimización iterativa de los modelos

débiles $h_m(x)$, que se ajustan para reducir el gradiente de la función de pérdida en cada iteración.

EJEMPLO 2.

Supongamos que tenemos un conjunto de datos con tres observaciones representado en la tabla 1, donde cada observación tiene una etiqueta binaria y_i y una predicción del modelo $F(x_i)$.

TABLA 1

Conjunto de datos para el ejemplo 2

y_i	$F(x_i)$
1	1.2
0	-0.8
1	0.5

Calculamos la probabilidad predicha usando la función sigmoide:

$$p_i = \sigma\left(F\left(x_i\right)\right) = \frac{1}{1+e^{-F\left(x_i\right)}}$$

Aplicamos esta transformación a cada observación:

$$p_1 = \frac{1}{1+e^{-1.2}} \approx 0.7685$$

$$p_2 = \frac{1}{1+e^{0.8}} \approx 0.3110$$

$$p_3 = \frac{1}{1+e^{-0.5}} \approx 0.6225$$

La función de pérdida logarítmica se define como:

$$L(y, F(x)) = -[\, y \ln(p) + (1 - y) \ln(1 - p)$$

Sustituyamos los valores para cada observación:

$$L_1 = -\left[1 \cdot ln(0.7685) + (1-1) \cdot ln(1 - 0.7685)\right] \approx -ln(0.7685) \approx 0.2633$$

$$L_2 = -\left[0 \cdot ln(0.3100) + (1-0) \cdot ln(1 - 0.3100)\right] \approx -ln(0.6900) \approx 0.3711$$

$$L_3 = -\left[1 \cdot ln(0.6225) + (1-1) \cdot ln(1 - 0.6225)\right] \approx -ln(0.6225) \approx 0.4740$$

Por lo tanto, la pérdida total:

$$L_{total} = \frac{1}{n}\sum_{i=1}^{n} L_i = \frac{0.2633 + 0.3711 + 0.4740}{3} \approx 03695.$$

El valor obtenido de log loss es 0.3695 nos indica qué tan bien el modelo está prediciendo las probabilidades correctas para cada clase. Un valor más cercano a 0 sugiere que el modelo está haciendo predicciones más seguras y precisas, mientras que un valor mayor indica que las probabilidades asignadas por el modelo no se alinean bien con las etiquetas reales.

En este caso, el log loss de 0.3695 sugiere que el modelo está logrando una calidad de predicción razonable, pero aún podría mejorarse.

MANEJO DE VARIABLES CATEGÓRICAS

Las *variables categóricas* son un tipo de variable que representa datos cualitativos, divididos en categorías discretas. A diferencia de las variables numéricas, estas no tienen un orden intrínseco ni una escala de medida cuantitativa.

En el contexto de modelos estadísticos y machine learning, las variables categóricas suelen clasificarse en:

- **Nominales**: Cuando no existe un orden entre las categorías (ej.: color, género).
- **Ordinales**: Cuando las categorías tienen un orden natural (ej.: nivel educativo: primaria, secundaria, universidad).

Para incluirlas en modelos matemáticos, generalmente se aplican técnicas como:

- **One-Hot Encoding**: Creación de variables binarias (0/1) para cada categoría.
- **Label Encoding**: Asignación de un valor numérico a cada categoría (útil para variables ordinales).

El manejo adecuado de estas variables es crucial, ya que una codificación incorrecta puede afectar significativamente el desempeño del modelo.

MANEJO DE VARIABLES CATEGÓRICAS EN CATBOOST.

Una de las principales ventajas de CatBoost es su capacidad para manejar eficientemente variables categóricas mediante la técnica de *ordered target encoding*. Esta técnica es especialmente útil para evitar el problema de *data leakage*, que ocurre cuando la información del conjunto de prueba se filtra en el conjunto de entrenamiento, lo que puede llevar a una sobreestimación del rendimiento del modelo, debido a que se ha entrenado con datos ya conocidos.

En el *ordered target encoding*, las variables categóricas se codifican utilizando estadísticas basadas en la variable objetivo, como la media de la variable objetivo para

cada categoría. Sin embargo, a diferencia de otras técnicas de codificación, CatBoost utiliza una permutación aleatoria de los datos durante el entrenamiento para evitar el data leakage. Formalmente, para una categoría c en la variable categórica x, el valor codificado Enc(c) se calcula como:

$$Enc(c) = \frac{\sum_{i=1}^{n} I(x_i=c) \cdot y_i + \alpha \cdot p}{\sum_{i=1}^{n} I(x_i=c) + \alpha}$$

donde:

- $I(x_i=c)$ es la función indicadora que vale 1 si = c y 0 en caso contrario.
- y_i es la variable objetivo.
- α es un parámetro de regularización que controla el efecto de la media global p (la proporción de la clase positiva en el conjunto de datos).

Al utilizar una permutación aleatoria de los datos, CatBoost asegura que la codificación de cada instancia se base únicamente en las instancias anteriores en la permutación, evitando así el data leakage.

EJEMPLO 3.

A continuación, mostramos un ejemplo de uso de *ordered target encoding* para la predicción de compras por distrito en Lima. Consideremos la tabla 2 un dataset de 6 clientes con la variable categórica *Distrito* y la variable objetivo y (1=Compra, 0=No compra):

Tabla 2

Conjunto de datos para el ejemplo 3

Cliente	Distrito	y
1	Miraflores	1
2	Barranco	0
3	San Isidro	1
4	Miraflores	1
5	Barranco	0
6	San Isidro	0

Parámetros:

- $\alpha = 1$
- p = 0.5
- Permutación aleatoria de clientes: [3, 5, 1, 2, 6, 4]

Además el valor de c para este caso toma 3 únicos valores: Miraflores, Barranco y San Isidro.

Ahora vamos recorriendo en el orden de la permutación aleatoria:

CÁLCULO PARA CLIENTE 3 (SAN ISIDRO, Y=1):

Primera aparición de San Isidro y datos anteriores ninguno.

$$Enc(San\ Isidro) = \frac{\alpha \cdot p}{\alpha} = 0.5$$

CÁLCULO PARA CLIENTE 5 (BARRANCO, Y=0):

Primera aparición de Barranco y datos anteriores cliente 3.

$$Enc(Barranco) = \frac{I(x_1 = Barranco) \bullet y_1 + \alpha \cdot p}{I(x_1 = Barranco) + \alpha}$$

$$= \frac{0 \bullet 1 + \alpha \cdot p}{0 + \alpha} = 0.5$$

CÁLCULO PARA CLIENTE 1 (MIRAFLORES, Y=1):

Primera aparición de Miraflores y datos anteriores clientes [3,5].

$$Enc(Miraflores) = \frac{\sum_{i=1}^{2} I(x_i = Miraflores) \bullet y_i + \alpha \bullet p}{\sum_{i=1}^{2} I(x_i = Miraflores) + \alpha}$$

$$= \frac{0 \bullet 1 + 0 \bullet 0 + \alpha \cdot p}{0 + 0 + \alpha} = 0.5$$

CÁLCULO PARA CLIENTE 2 (BARRANCO, Y=0):

Barranco ya apareció, se ajusta el encoding y datos anteriores [3,5,1].

$$Enc(Barranco) = \frac{\sum_{i=1}^{3} I(x_i = Barranco) \bullet y_i + \alpha \bullet p}{\sum_{i=1}^{3} I(x_1 = Barranco) + \alpha}$$

$$= \frac{0 \bullet 1 + 1 \bullet 0 + 0 \bullet 1 + \alpha \cdot p}{0 + 1 + 0 + \alpha} = 0.5$$

CÁLCULO PARA CLIENTE 6 (SAN ISIDRO, Y=0):

San Isidro ya apareció, se ajusta el encoding y datos anteriores [3,5,1,2].

$$Enc(San\ Isidro) = \frac{\sum_{i=1}^{4} I(x_1 = San\ Isidro) \bullet y_i + \alpha \cdot p}{\sum_{i=1}^{4} I(x_1 = San\ Isidro) + \alpha}$$

$$= \frac{1 \bullet 1 + 0 \bullet 0 + 0 \bullet 1 + 0 \bullet 0 + 1 \bullet 0.5}{0 + 1 + 0 + 1} = \frac{1.5}{2} = 0.75$$

CÁLCULO PARA CLIENTE 4 (MIRAFLORES, Y=1):

Miraflores ya apareció, se ajusta el encoding y datos anteriores [3,5,1,2,6].

$$Enc(Miraflores) = \frac{\sum_{i=1}^{5} I(x_i = Miraflores) \bullet y_i + \alpha \bullet p}{\sum_{i=1}^{5} I(x_i = Miraflores) + \alpha}$$

$$= \frac{0 \bullet 1 + 0 \bullet 0 + 1 \bullet 1 + 0 \bullet 0 + 0 \bullet 0 + 1 \bullet 0.5}{0 + 1 + 0 + 1} = \frac{1.5}{2} = 0.75$$

Tabla 3

Valor final asignado a cada categoría

Distrito	Encoding Final
Miraflores	0.75
Barranco	0.50
San Isidro	0.75

INTERPRETACIÓN COMERCIAL:

- Miraflores y San Isidro muestran alta probabilidad de compra 75%
- Barranco se mantiene en la media global 50%
- El encoding refleja patrones reales: distritos comerciales vs. residenciales

Al final, el uso sería que, al momento de entrenar el modelo, se ingresen los valores numéricos en lugar de la categoría como lo mostrado en la tabla 3. Por ejemplo, en lugar de la categoría Miraflores, iría el valor numérico 0.75, y así con las demás categorías.

VENTAJAS DE CATBOOST SOBRE OTROS ALGORITMOS

El Gradient Boosting ofrece varias ventajas en comparación con otros métodos de aprendizaje supervisado:

FLEXIBILIDAD:

Puede manejar tanto problemas de clasificación como de regresión, y es compatible con diversas funciones de pérdida (Prokhorenkova, 2019).

Ejemplo 4.

Una plataforma de e-commerce necesita:

- Clasificación: Predecir si un cliente comprará (Sí/No) usando *LogLoss*.

- Regresión: Estimar el monto de compra (USD) con *RMSE*.
- Datos mixtos: variables numéricas (edad, historial) y categóricas (ciudad, dispositivo).

CatBoost resuelve ambos casos con la misma librería:

- Cambiando el parámetro *loss_function* según el problema.
- Procesando automáticamente categorías sin preprocesamiento manual.

Esto simplifica la implementación y reduce errores en producción.

CAPACIDAD PARA CAPTURAR RELACIONES NO LINEALES:

Al utilizar árboles de decisión como modelos débiles, el Gradient Boosting puede capturar relaciones no lineales y complejas entre las características y la variable objetivo (Dorogush, 2018).

EJEMPLO 5.

En un estudio de marketing. Un modelo lineal podría predecir que *a mayor inversión en publicidad, mayores ventas* (relación directa). CatBoost, en cambio, identifica que:

- Las ventas crecen con la publicidad hasta cierto punto, pero decrecen si hay saturación (relación cuadrática).
- El efecto cambia según el *grupo demográfico*: jóvenes responden a redes sociales, adultos a TV (interacciones no lineales).

Esto se debe a que sus árboles de decisión dividen los datos en segmentos y combinan sus resultados jerárquicamente.

ROBUSTEZ FRENTE AL SOBREAJUSTE:

La tasa de aprendizaje y la limitación de la profundidad de los árboles ayudan a reducir el sobreajuste, especialmente en conjuntos de datos pequeños (Prokhorenkova, 2019).

EJEMPLO 6.

En un hospital con pocos registros de pacientes (200 casos), se quiere predecir el riesgo de diabetes. Un modelo complejo:

- Podría memorizar ruido (como valores atípicos en análisis de sangre) en lugar de patrones reales.
- Tendría alto rendimiento en los datos de entrenamiento pero fallaría con nuevos pacientes.

CatBoost controla esto mediante:

- Árboles de profundidad máxima 4 (evita patrones demasiado específicos).
- Ajuste gradual con tasa de aprendizaje baja (=0.01), mejorando iterativamente sin sobreajustar.

Así logra generalizar bien incluso con datos limitados.

CONCLUSIÓN

Este artículo ha explorado los fundamentos del Gradient Boosting, centrándose en CatBoost como una de sus implementaciones más avanzadas. Mediante ejemplos prácticos, se demostró cómo este algoritmo optimiza modelos débiles, mediante un aprendizaje secuencial que mejora progresivamente la precisión predictiva. Además, se destacó su manejo eficiente de variables categóricas mediante técnicas como el ordered target encoding, que simplifica su implementación y supera limitaciones de otros métodos de boosting. CatBoost también se distingue por su capacidad para mitigar el sobreajuste y su eficiencia computacional, lo que lo convierte en una herramienta ideal para el análisis de grandes volúmenes de datos. Los casos presentados ilustran cómo sus fundamentos teóricos se traducen en aplicaciones prácticas, consolidando su relevancia en problemas diversos de machine learning.

REFERENCIA BIBLIOGRÁFICA

Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32.

Dorogush, A. V., Ershov, V., y Gulin, A. (2018). Catboost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363. https://arxiv.org/abs/1810.11363. doi: 10.48550/arXiv.1810.11363

Menard, S. (2002). Applied logistic regression analysis. Thousand Oaks: Sage Publications.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2019). Catboost: Unbiased boosting with categorical features. arXiv preprint, v5. https://doi.org/10.48550/arXiv.1706.09516