Clasificación de bacterias resistentes a antibióticos mediante modelo XGBoost

Richard Muñoz Henao
Universidad Nacional de Colombia, Sede Medellín

Noviembre 2025

Resumen

En este informe se documenta la estrategia, matemáticas, optimizaciones computacionales y resultados cuantitativos del modelo de clasificación basado en **XGBoost** para distinguir genomas bacterianos resistentes y susceptibles al antibiótico (ampicilina) usando frecuencias de *k-mers* extraídas de *Open Reading Frames* (ORFs). El documento sintetiza las decisiones de diseño (unidad de análisis, ingeniería de características, elección de modelo), presenta la función objetivo del algoritmo y muestra la evolución de la precisión del sistema según distintas mejoras.

Índice

1. Marco Teórico y Bioinformática

1.1. Open Reading Frames (ORFs)

Los *Open Reading Frames* (ORFs) son segmentos continuos de una secuencia de ADN que tienen el potencial de ser traducidos en proteínas; típicamente comienzan con un codón de inicio (p. ej. ATG) y terminan en un codón de parada (TAA, TAG o TGA) en el mismo marco de lectura. En bacterias, los ORFs capturan la mayoría de la información funcional relevante (genes codificantes de proteínas, factores de resistencia, enzimas), por lo que se usan como **unidad de análisis** en lugar de la secuencia genómica completa cuando el objetivo es caracterizar fenotipos

ligados a funciones moleculares.
Justificaciones principales para usar ORFs:

- **Señal funcional concentrada:** muchos determinantes de resistencia (por ejemplo, genes codificantes de ß-lactamasas) se encuentran en ORFs; analizar ORFs reduce ruido de regiones intergénicas no codificantes.
- **Reducción dimensional y ruido:** procesar únicamente ORFs evita invertir cómputo en largas regiones repetitivas o no informativas (p. ej. regiones intergénicas, elementos móviles no codificantes).
- **Biológicamente interpretables:** las características (k-mers) pueden mapearse luego a ORFs específicas, facilitando interpretación biológica y búsqueda de motivos asociados a resistencia.

1.2. K-mers: frecuencia como descriptor genómico

Un *k-mer* es una subcadena contigua de longitud k de nucleótidos extraída de una secuencia. La representación de una secuencia mediante el conteo o la frecuencia de todos los k-mers posibles captura patrones locales de composición y señales de "firma" genómica (bias composicional, motivos fonales, señales codificantes). Para secuencias bacterianas:

- Los k-mers reflejan tanto la composición general (GC content) como motivos codificantes y señales de elementos móviles.
- Al usar k-mers sobre ORFs, se enfatizan patrones relacionados con secuencias codificantes (p. ej. motivos de aminoácidos codificados) y se mejora la sensibilidad a variantes funcionales.

En el trabajo aquí presentado se empleó inicialmente $k = 6$ (lo que genera $4^6 = 4096$ características por muestra cuando se consideran las 4 bases A,C,G,T), un tamaño que equilibra resolución (motivos suficientemente largos) y dimensionalidad.

1.3. Ingeniería de características: frecuencias relativas de k-mers

En lugar de usar conteos crudos de k-mers se usa la *frecuencia relativa* (normalización por la longitud efectiva de la región considerada). Si $c_s(u)$ es el conteo del k-mer u en la muestra s y L_s la longitud total (en nucleótidos) de las ORFs analizadas para s, la frecuencia relativa se define como:

$$f_s(u) \;=\; \frac{c_s(u)}{\sum_v c_s(v)} \;\approx\; \frac{c_s(u)}{L_s - k + 1}. \tag{1}$$

Ventajas de esta normalización:

- Permite comparar muestras con longitudes de ORFs distintas evitando sesgos por tamaño.
- Reduce varianza entre muestras debida a diferencias en cobertura o ensamblaje (cantidad de contigs concatenados para el genoma).
- Facilita la convergencia de algoritmos de aprendizaje (features en escala comparable) y evita que features con mayor escala dominen la función de pérdida.

2. Metodología

La metodología seguida para el desarrollo del clasificador XGBoost se estructuró en cinco etapas principales: (1) extracción de datos desde bases de datos biológicas, (2) procesamiento y representación de las secuencias mediante ORFs y k-mers, (3) búsqueda de hiperparámetros mediante optimización bayesiana, (4) entrenamiento del modelo final y (5) evaluación cuantitativa mediante métricas de clasificación binaria.

2.1. Extracción de datos

Se recopilaron 6460 genomas bacterianos de la especie *Escherichia coli* a partir de la base de datos BV-BRC junto con sus etiquetas de resistencia o susceptibilidad frente a un antibiótico específico (ampicilina). Cada genoma fue descargado en formato `FASTA` y preprocesado para extraer representaciones vectoriales adecuadas para el posterior entrenamiento.

2.2. Procesamiento y generación de características

El flujo de procesamiento incluyó varias etapas consecutivas:

1. **Concatenación de secuencias:** las secuencias genómicas provenientes de múltiples contigs fueron concatenadas para obtener una representación continua por genoma.
2. **Identificación de ORFs:** se aplicó un algoritmo de detección de *Open Reading Frames* (ORFs) para identificar las regiones potencialmente codificantes, utilizando como criterios la presencia de codones de inicio y terminación válidos.
3. **Extracción de k-mers:** de cada ORF se extrajeron todos los substrings contiguos de longitud k, generando un vector de frecuencia de k-mers para cada muestra.
4. **Normalización:** los conteos de k-mers se transformaron en *frecuencias relativas* según la ecuación 1, con el fin de eliminar sesgos asociados a la longitud total de las ORFs y hacer comparables las muestras.

El resultado fue una matriz de características $X \in \mathbb{R}^{n \times m}$, donde $n = 6460$ corresponde al número total de muestras analizadas y $m = 4^k$ al número total de posibles k-mers para el valor de k seleccionado. En este trabajo se empleó $k = 6$, lo que produce un espacio de características de tamaño $m = 4^6 = 4096$. Cada muestra queda representada entonces por un vector de 4096 frecuencias relativas de k-mers extraídas de las ORFs correspondientes a su genoma.

2.3. Selección del modelo: ¿por qué XGBoost?

XGBoost (Extreme Gradient Boosting) es un método de boosting basado en árboles de decisión que destaca en datos tabulares. Motivos para elegir XGBoost en esta tarea:

- **Rendimiento en datos tabulares:** a menudo supera a modelos lineales y redes neuronales estándar en conjuntos con features estructuradas como frecuencias de k-mers.
- **Robustez y generalización:** incorpora regularización explícita y técnicas para reducir overfitting.
- **Eficiencia y escalabilidad:** implementaciones optimizadas (paralelización, manejo de sparse matrices) permiten entrenar modelos rápidamente, lo que facilita la búsqueda de hiperparámetros.
- **Interpretabilidad:** importancias de features y herramientas SHAP permiten interpretación de las contribuciones de k-mers.

2.4. Estrategia de búsqueda de hiperparámetros: Optimización Bayesiana (Hyperopt)

Para la sintonía de hiperparámetros (por ejemplo: η, max_depth, min_child_weight, subsample, colsample_bytree, k de los k-mers, regularizadores λ, γ) se empleó **Optimización Bayesiana** mediante la librería `hyperopt`. Justificación:

- **Eficiencia en evaluaciones costosas:** la evaluación de cada configuración implica entrenamiento y validación (a menudo con validación cruzada), por lo que minimizar el número de evaluaciones necesarias es crítico.
- **Modelado de incertidumbre:** Hyperopt (con TPE) construye un modelo de la función objetivo (distribución de buenas vs malas configuraciones) y prioriza regiones prometedoras del espacio hiperparamétrico.
- **Exploración/explotación balanceada:** a diferencia de grid o random search, la optimización bayesiana dirige las pruebas hacia configuraciones con alta probabilidad de mejorar la métrica objetivo.
- **Resultados prácticos:** en este caso se exploraron 200 configuraciones con validación cruzada a 4 folds (4 horas de cómputo) y se obtuvo una mejora clara de la exactitud del modelo.

2.5. Entrenamiento del modelo

Con los hiperparámetros óptimos seleccionados, se entrenó el modelo final de **XGBoost** utilizando todo el conjunto de entrenamiento. El modelo se entrenó con la función de pérdida logística (`logloss`) para clasificación binaria y se monitoreó la convergencia del error mediante validación cruzada. El entrenamiento se realizó sobre hardware de alto rendimiento (CPU Intel Core i9 con 32 hilos, GPU RTX ADA 2000, 64 GB de RAM), permitiendo un procesamiento paralelo eficiente en la fase de preprocesamiento y entrenamiento.

2.6. Evaluación y métricas

El modelo final fue evaluado sobre un conjunto de prueba independiente, calculando métricas estándar de clasificación binaria: **Precisión (Precision)**, **Sensibilidad (Recall)**, **F1-score**, **Accuracy**, **Macro y Weighted Averages**, y el **ROC-AUC**. Estas métricas permitieron cuantificar tanto el desempeño global del clasificador como el balance entre falsos positivos y falsos negativos en la predicción de resistencia bacteriana.

A continuación se presentan las fórmulas utilizadas para las métricas de clasificación binaria

y los valores obtenidos en el experimento final.

$$\text{Precision (Precisión)} = \frac{TP}{TP + FP}$$

$$\text{Recall (Sensibilidad)} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Support} = \text{Número de muestras reales de cada clase}$$

$$\text{Accuracy (Exactitud)} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Macro Average} = \frac{1}{2}(\text{métrica para clase True} + \text{métrica para clase False})$$

$$\text{Weighted Average} = \frac{(\text{métrica clase True} \cdot \text{support True}) + (\text{métrica clase False} \cdot \text{support False})}{\text{support True} + \text{support False}}$$

$$\text{ROC-AUC} = \text{Área bajo la curva ROC, que mide la capacidad de separar clases}$$

Reporte completo de métricas resultantes

	precision	**recall**	**f1-score**	**support**
False	0.82	0.77	0.80	571
True	0.83	0.87	0.85	721
accuracy			0.83	1292
ROC-AUC			0.89	1292
macro avg	0.83	0.82	0.82	1292
weighted avg	0.83	0.83	0.82	1292

Evolución de la Exactitud

Hito del Modelo	**accuracy**
Sin búsqueda de hiperparámetros ($k = 7$)	0.67
Optimización de k a 6	0.73
Con optimización bayesiana de hiperparámetros	0.77
Con búsqueda de ORFs y cálculo de k-mers en ORFs	0.81
K-mers convertidos a Frecuencias Relativas	0.83

Los resultados anteriores muestran una mejora incremental al incorporar mejores prácticas: optimización del tamaño del k-mer, búsqueda bayesiana de hiperparámetros y preprocesamiento centrado en ORFs y frecuencias relativas.

3. Optimización Computacional

3.1. Paralelización en preprocesamiento (cálculo de k-mers)

El cómputo de k-mers sobre miles de genomas y miles de ORFs por genoma es una operación embarrassingly parallel (independiente por muestra y por contig). Se implementaron las siguientes optimizaciones:

- **Procesamiento por chunks:** lectura de archivos FASTA en bloques por genoma y extracción concurrente de ORFs.
- **Paralelización por hilos/procesos:** se asignaron múltiples workers (p. ej. 32 hilos en una plataforma *Core i9 con 32 hilos*) que procesan genomas en paralelo, haciendo uso de pools y operaciones vectorizadas en NumPy cuando fue posible.
- **I/O eficiente y streaming:** se evitó la creación de enormes strings concatenadas en memoria mediante streaming y almacenando recuentos parciales en estructuras dispersas (sparse) para reducir memoria.

Gracias a estas optimizaciones el tiempo de preprocesamiento se redujo drásticamente: tareas que antes tardaban **días** en un flujo secuencial se ejecutan en **minutos** cuando se aprovechan 32 hilos y una gestión eficiente de I/O y memoria. Esto habilita iteraciones rápidas sobre k y estrategias de validación.

4. Discusión y Conclusiones

- La elección de ORFs como unidad de análisis aumentó la señal funcional disponible para el clasificador y redujo ruido de regiones no codificantes.
- El paso de contar k-mers a normalizarlos como frecuencias relativas mejoró la comparabilidad entre muestras y la estabilidad del entrenamiento, produciendo la última mejora observada (de 0,81 a 0,83 de accuracy).
- XGBoost resultó ser una herramienta adecuada por su eficiencia en datos tabulares, robustez frente al sobreajuste y soporte de optimizaciones computacionales.
- La optimización bayesiana (Hyperopt) permitió dirigir la búsqueda de hiperparámetros de manera eficiente, justificando su uso frente a búsquedas exhaustivas o aleatorias en problemas con evaluaciones costosas.

Referencias

[1] Nature (2024). Resistance crisis: By 2050, antimicrobial resistance could be responsible for 1.91 million deaths per year. (Gráfico y resumen del problema global de AMR).

[2] Öz-Tornacı, B. (2024). Simplified structure of XGBoost. ResearchGate.

[3] P. Melsted and J. K. Pritchard. Efficient counting of k-mers in DNA sequences using a Bloom filter. *Bioinformatics*, 27(6):764–770, 2011. doi: 10.1093/bioinformatics/btr018.

[4] D. Hyatt, G. L. Chen, P. F. LoCascio, M. L. Land, F. W. Larimer, and L. J. Hauser. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, 11(1):119, 2010. doi: 10.1186/1471-2105-11-119.