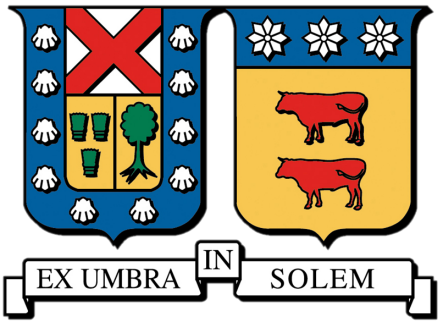
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



Aplicación de técnicas de Machine Learning para realizar detección de promotores en múltiples especies

Tesis de Grado presentada por

MARCELO IVÁN GONZÁLEZ HENRÍQUEZ

como requisito parcial para optar al título de

MAGÍSTER EN CIENCIAS DE LA INGENIERÍA ELECTRÓNICA

PROFESOR GUÍA:	NICOLÁS JARA
CO-SUPERVISOR:	ROBERTO DURÁN
EVALUADORA EXTERNA:	CONSTANZA CÁRDENAS

Enero 2025

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título; ☒ Tesis de Postgrado;

Título del trabajo: Aplicación de técnicas de Machine Learning para realizar detección de promotores en múltiples especies

Nombre del candidato(a): Marcelo Iván González Henríquez

Carrera / Grado: Magíster en Ciencias de la Ingeniería Electrónica

Campus: Casa Central Valparaiso **;** **Departamento**: Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nicolás Alonso Jara Carvallo, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 01/09/2025 ; **Firma:** _____

Estudiante o Candidato(a):

Fecha: 01/09/2025 ; **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Quiero expresar mi profundo agradecimiento a los profesores Nicolás Jara, Mauricio Araya y a Roberto Durán por su respaldo en este trabajo y por fomentar instancias de intercambio de ideas más allá de la Universidad. Su guía y estímulo han sido fundamentales para mi crecimiento tanto personal como profesional.

Asimismo, deseo agradecer a mis amigos y familiares, quienes han sido un pilar fundamental en los momentos más difíciles. Su apoyo incondicional, su compañía y las distintas perspectivas que me han brindado no solo me han dado fuerzas para seguir adelante, sino que también me han permitido crecer y comprender las situaciones desde nuevas miradas.

Resumen

En aspectos de la genética, un promotor es una secuencia de ADN que regula y promueve el inicio de la transcripción genética, siendo grandes responsables del control de todos los procesos fisiológicos de una célula. La identificación precisa de los promotores es crucial para comprender la transcripción de ADN, lo que puede llevar a beneficios como el incremento en la producción de fármacos en microorganismos y la mejora de las propiedades de los cultivos agrícolas. A pesar de su importancia, los promotores carecen de patrones de secuencia de ADN claramente conservados entre diferentes tipos de promotores, así como entre especies, lo que complica su caracterización. En la última década, la disponibilidad pública de datos de promotores de diversas especies ha fomentado el desarrollo de modelos de Machine Learning para abordar esta tarea.

La detección de promotores utilizando métodos de Machine Learning implica múltiples desafíos. Los datos genéticos disponibles a menudo son insuficientes, lo que limita la disponibilidad de modelos efectivos en términos del rango de especies que pueden abarcar. Además, un problema significativo es la falta de consenso sobre la definición de *clase negativa* en el contexto de clasificación binaria, es decir, qué secuencias de ADN se deben considerar como no promotores. Este problema no solo complica la creación de conjuntos de datos representativos, sino que también afecta la capacidad de los modelos para generalizar en diferentes contextos genómicos.

En este trabajo, se propone un enfoque basado en técnicas de Machine Learning para la detección de promotores, abordando el problema como una tarea de clasificación binaria en múltiples especies. Para ello, se construyen dos conjuntos de datos: uno basado en secuencias codificantes y otro en secuencias generadas sintéticamente, utilizadas como clases negativas. Además, se considera el contenido de GC como una variable de interés clave, ya que podría introducir sesgos que afecten el rendimiento de los modelos.

El enfoque propuesto incluye la evaluación de modelos utilizando métricas estándar para clasificadores binarios y se comparan con métodos convencionales. También se analizan diversos escenarios representando especies con distribuciones variables de contenido de GC

para evaluar el impacto del origen de las secuencias no promotoras en los resultados.

Finalmente, se examina la capacidad de los modelos desarrollados para adaptarse a datos genómicos reales, y se proponen recomendaciones para mitigar los posibles sesgos derivados del contenido de GC, mejorando la generalización de los modelos en contextos biológicos complejos. Este trabajo establece una base para futuras investigaciones en la predicción de promotores y el diseño de herramientas automatizadas en genómica.

Abstract

Promoters, essential DNA sequences that regulate gene transcription, play a critical role in controlling cellular physiological processes. Accurate promoter identification is vital for understanding transcription mechanisms, with applications such as improving microbial drug production and enhancing crop traits. Despite their importance, promoters lack conserved sequence patterns across types and species, complicating their characterization. Recent advances in publicly available promoter datasets have driven the development of Machine Learning models to address these challenges.

This study proposes a Machine Learning-based framework for binary classification of promoters across multiple species. Two datasets are constructed: coding sequences and synthetically generated sequences, used as the negative class. GC content, a key variable due to its potential to introduce biases, is analyzed to assess its influence on model performance.

The framework evaluates models using standard binary classification metrics, comparing their effectiveness across diverse GC content distributions. Scenarios involving species with varying genomic characteristics are analyzed to evaluate the impact of the negative sequence origin. The models' adaptability to real genomic data is also examined, with recommendations proposed to mitigate GC-related biases and improve generalization.

This work provides a foundation for future research on promoter prediction and the development of automated genomic tools, enabling more accurate and scalable applications in genomic analysis.

keywords: deep learning, machine learning, bacterial promoters, BERT, natural language processing, convolutional neural networks (CNN), binary classification, GC content, bioinformatics, promoter prediction.

Índice de figuras

Índice general

Capítulo 1

Introducción

1.1. Definición del problema y motivación

La genómica, que se centra en analizar el genoma completo de un organismo, ha avanzado significativamente gracias al gran crecimiento de información disponible [1], además de la introducción de técnicas de secuenciación de próxima generación (NGS) [2]. Estos métodos han acelerado la recolección de datos genéticos, aprovechando la tecnología de cómputo contemporánea.

La regulación de la transcripción genética es un proceso fundamental en los organismos, ya que controla cómo se traduce la información contenida en el ADN en ARN, el paso previo a la síntesis de proteínas. Este proceso no solo asegura que los genes se activen en el momento adecuado, sino que también regula la cantidad de ARN producido, lo que es esencial para mantener el equilibrio en las funciones celulares. En este contexto, los promotores juegan un papel fundamental al actuar como elementos regulatorios que definen los puntos de inicio de la transcripción, sirviendo como los principales *interruptores* que controlan la expresión génica. Comprender y predecir la ubicación y función de los promotores es esencial para desentrañar los mecanismos subyacentes de la expresión génica y sus implicaciones en la salud y la biotecnología.

La detección de promotores es esencial debido a su papel en la regulación de la expresión

génica, lo que implica el control de cuándo y en qué cantidad se transcribe un gen. Esta regulación es clave para garantizar la producción de las proteínas necesarias en los momentos y lugares precisos dentro del organismo. Las herramientas que hagan este tipo de tareas permiten identificar con precisión los puntos de inicio de la transcripción, lo que ayuda en el diseño de terapias genéticas [3] y en la ingeniería genética [4]. Además, mejoran nuestra capacidad para modificar organismos para aplicaciones en biotecnología y agricultura [5, 6], y contribuyen al desarrollo de diagnósticos más precisos y tratamientos personalizados en medicina [3, 7].

No obstante, identificar promotores es un reto considerable. A pesar de que hay patrones de secuencia conocidos para identificar promotores en especies como *E.coli* o en humanos, en otras especies no existe un patrón claro y uniforme, el cual varía entre especies. Estos factores sugieren la implementación de estrategias sofisticadas para abordar la diversidad y complejidad de estos elementos reguladores. En este sentido, las técnicas de machine learning han demostrado ser particularmente útiles en la detección de patrones complejos en diversos dominios, como la clasificación de imágenes médicas para el diagnóstico de enfermedades y la detección de fraudes en transacciones financieras. Su capacidad para identificar patrones en los datos las posiciona como una herramienta prometedora para la detección de promotores.

Recientemente, se han desarrollado modelos basados en técnicas de Machine Learning para abordar el desafío de la detección de promotores. Estos modelos pueden diseñarse tanto para identificar promotores específicos de una única especie [8, 9, 10, 11, 12, 13, 14, 15] como para operar en un contexto multi-especie utilizando un modelo único [16, 17, 18]. En ambos casos, el enfoque más común consiste en emplear un clasificador binario, diseñado para diferenciar entre secuencias promotoras y no promotoras.

En paralelo, un área significativa de investigación en Machine Learning se centra en el procesamiento de lenguaje natural (NLP), donde la arquitectura Transformer [19] ha impulsado el desarrollo de Large Language Models (LLMs), modelos de lenguaje de gran tamaño entrenados en conjuntos de datos masivos. Estos modelos aprovechan el mecanismo de atención para capturar relaciones entre elementos distantes en las secuencias y permiten un

entrenamiento altamente paralelizado. En el contexto de la bioinformática, estas capacidades de los LLMs, como su habilidad para aprender patrones complejos y generalizar en tareas diversas, se han comenzado a explorar para abordar problemas como la detección de promotores. Además, se han desarrollado modelos de este tipo adaptados específicamente para datos genómicos, permitiendo capturar características únicas de las secuencias biológicas y mejorar el rendimiento en tareas relacionadas con la regulación genética [20, 21, 22, 23].

A pesar del desarrollo de soluciones basadas en Machine Learning para la detección de promotores, persisten diversos desafíos que limitan su efectividad. Uno de los principales problemas es la falta de promotores anotados [1] disponibles para su uso, lo que dificulta la construcción de conjuntos de datos para el entrenamiento de modelos. Otro desafío importante radica en la falta de consenso sobre qué secuencias utilizar como no-promotores para la clasificación. Aunque es común emplear fragmentos de secuencias codificantes o regiones intergénicas como conjuntos negativos, a menudo no se detalla claramente el procedimiento de extracción de estas secuencias. Por otro lado, en modelos entrenados con promotores *E. coli* y utilizando secuencias codificantes de dicha especie como datos negativos, se han identificado sesgos relacionados con el contenido GC, es decir, el porcentaje de guanina (G) y citosina (C) presentes en las secuencias de ADN [12]. Estos sesgos afectan la clasificación al dificultar la detección precisa de promotores. Para mitigar este problema, se ha sugerido utilizar secuencias generadas aleatoriamente pero con la misma distribución de contenido GC, permitiendo reducir el sesgo de los modelos y mejorar su capacidad de generalización para modelos de una especie [12].

Esta tesis tiene como objetivo desarrollar un modelo único para la detección de promotores en múltiples especies, abordando el desafío de generalizar las predicciones a través de contextos genómicos diversos. Un obstáculo significativo para este enfoque es el sesgo relacionado con el contenido GC, introducido frecuentemente al utilizar secuencias codificantes (CDS) como dataset negativo. Para resolver este problema, esta tesis propone explorar el empleo de un dataset con promotores de múltiples especies, utilizando secuencias generadas

[1] Secuencias reportadas y caracterizadas con información funcional, genómica o estructural basada en datos experimentales o computacionales.

sintéticamente que emulen la distribución de contenido GC de los promotores como secuencias no promotoras, para comprobar si acaso se mitiga el sesgo y así obtener un modelo con resultados fiables, contribuyendo al avance de las herramientas computacionales en la regulación genética.

1.2. Hipótesis

Tomando en cuenta la definición del problema y el estado del arte, las hipótesis planteadas para este trabajo son las siguientes:

1. *El uso de secuencias codificantes (CDS) como dataset negativo en un modelo único, diseñado para realizar predicciones de promotores en múltiples especies, introduce un sesgo relacionado con el contenido GC en las predicciones del modelo.*
   Diversos modelos de detección de promotores emplean secuencias codificantes (CDS) como dataset negativo, pero se ha evidenciado que dicho enfoque introduce un sesgo relacionado con el contenido GC, principalmente en modelos diseñados para una especie [12]. Dado que no existen investigaciones que evalúen este problema en el contexto de un modelo único para múltiples especies, este trabajo busca determinar si dicho sesgo persiste en este nuevo contexto.

2. *El empleo de secuencias generadas sintéticamente, configuradas para emular la distribución de contenido GC de los promotores, como dataset negativo en un modelo único diseñado para predecir promotores en múltiples especies, mitiga el sesgo relacionado con el contenido GC en las predicciones del modelo.*
   Se ha demostrado que este sesgo puede mitigarse empleando secuencias no promotoras con una distribución de contenido GC similar a la de los promotores como dataset negativo, mejorando así la precisión en la clasificación [12]; sin embargo, este enfoque ha sido estudiado únicamente en modelos diseñados para una especie. En este trabajo, se construirá un dataset negativo compuesto por secuencias con dichas características, con el fin de evaluar si este enfoque es efectivo para mitigar el sesgo en un modelo

único aplicado a múltiples especies.

1.3. Objetivos

El objetivo general del trabajo es implementar un modelo de Machine Learning capaz de realizar la detección de promotores en múltiples especies mediante un modelo único. Para lograr el objetivo general, este se desagrega en los siguientes objetivos específicos:

1. Definir las especies a estudiar, considerando su taxonomía y la distribución del contenido de GC en los promotores reportados en la literatura.

2. Elaborar dos conjuntos de datos que incluyan dos orígenes distintos de secuencias etiquetadas como no promotoras: secuencias codificantes y secuencias generadas sintéticamente.

3. Desarrollar y evaluar modelos de Machine Learning diseñados para identificar promotores en múltiples especies, abordando el problema como una clasificación binaria.

4. Evaluar cualitativamente la influencia del contenido GC en las decisiones de los modelos, en función del origen de las secuencias no promotoras.

1.4. Estructura del documento

Capítulo 1 Introducción

En este capítulo se presenta una visión general de esta investigación. Se discuten brevemente los antecedentes, el problema y la motivación de la investigación. Además, se plantean las preguntas de investigación, el objetivo general y los objetivos específicos del problema identificado.

Capítulo 2 Estado del Arte

En este capítulo se proporciona una visión general de las aproximaciones actuales en la

predicción de promotores, abarcando tanto modelos diseñados para una única especie como aquellos desarrollados para múltiples especies a través de un modelo único. Se analiza el impacto del sesgo de contenido GC en los modelos en configuraciones para una especie, y se discute cómo este afecta el desempeño y la generalización. Además, se revisa el papel de los *Large Language Models* en el análisis de datos genómicos, destacando su potencial y limitaciones. Finalmente, se incluye una recopilación de conjuntos de datos disponibles para promotores.

Capítulo 3 Marco Teórico

En este capítulo se proporciona una base conceptual para la investigación, abordando temas clave como la definición de un promotor y su relevancia en el contexto genómico. También se presentan los fundamentos sobre modelos de clasificación de texto en el ámbito del procesamiento del lenguaje natural (NLP) y cómo estos modelos se adaptan al análisis de secuencias genómicas. Además, se incluye una descripción de los principales tipos de modelos utilizados, sus características y las métricas de evaluación empleadas para medir su desempeño, proporcionando un marco general para interpretar los resultados del estudio.

Capítulo 4 Metodología

En este capítulo se define el enfoque utilizado para la investigación, que abarca desde el preprocesamiento de datos hasta el diseño del proceso de entrenamiento de los modelos, considerando los conjuntos de datos a estudiar. La metodología detalla los pasos necesarios, incluyendo la selección de especies relevantes, la creación y preprocesamiento de los datasets, y la selección de modelos, estableciendo así las bases para la ejecución y evaluación del proceso experimental.

Capítulo 5 Resultados

En este capítulo se presentan los resultados del preprocesamiento y la caracterización de los dos conjuntos de datos utilizados en este estudio, incluyendo su análisis en términos de contenido de GC. Asimismo, se exponen los hallazgos obtenidos durante la evaluación de

los modelos entrenados con los diferentes orígenes de secuencias no promotoras propuestos, utilizando métricas estándar para clasificadores binarios. Además, se analiza la influencia del contenido de GC en el desempeño de la clasificación según el origen de las secuencias. Por último, se examina el rendimiento de los modelos generados a partir de cada origen en un contexto real.

Capítulo 6 Conclusiones y Trabajos futuros

Este capítulo aborda los desafíos asociados con el desarrollo de modelos predictivos, destacando la importancia de abordar sesgos en los datos y equilibrar métricas clave como sensibilidad y especificidad. Asimismo, presenta conclusiones basadas en los hallazgos y plantea oportunidades para mejoras futuras.

Capítulo 2

Estado del Arte

2.1. Modelos para una especie

En el contexto de desarrollo de modelos detectores de promotores para una especie, se tienen varios trabajos relacionados a la detección de promotores en la especie *E.coli*. Lu et. al. (2017) presentaron iPromoter-2L [9], un modelo que utiliza un enfoque de clasificación en dos etapas con Random Forest para determinar si una secuencia es un promotor y luego identificar la clase del factor sigma. Rahman et. al. (2018) introdujeron iPro70-FMWin [11], un modelo de aprendizaje automático enfocado en la clasificación de promotores Sigma70, en donde realizaron evaluaciones con varios métodos, en donde se obtuvo que métodos como Regresión Logística y Support Vector Machines tienen mejor desempeño para realizar la detección. He et. al. (2018) propusieron 70ProPred [10], otro modelo basado en Support Vector Machines, dedicado a la detección de promotores Sigma70. Hernández et. al (2022) presentaron PromoterLCNN [15], un modelo que permite realizar la identificación y clasificación de promotores a través de dos etapas, basadas en Redes Neuronales Convolucionales (CNN). En comparación con otros clasificadores basados en CNN, PromoterLCNN mostró un rendimiento similar o mejor, reduciendo el tiempo de entrenamiento, predicción y optimización de hiperparámetros en un 30-90 %. En todos estos estudios se utilizó RegulonDB 9.0 [24] como fuente de datos para obtener secuencias promotoras de la especie *E.coli* y se

extrajeron secuencias no promotoras de regiones codificantes y zonas intergénicas para el entrenamiento de los modelos.

2.2. Modelos para mútiples especies

Por otro lado, se han desarrollado modelos para realizar la detección de promotores en varias especies a través de un modelo único, o bien, uno por cada especie. Umarov et. al. (2017) presentaron CNNProm [8], un modelo basado en CNN para la predicción de promotores en cinco especies distintas. El modelo fue entrenado con secuencias promotoras y no promotoras, en donde estas últimas fueron seleccionadas aleatoriamente de la cadena opuesta en regiones codificantes de genomas, en otras palabras, el complemento de las secuencias codificantes. Por otro lado, Oubounyt et. al. (2019) presentan DeePromoter [25], un modelo que integra una CNN con una red neuronal recurrente LSTM, diseñado para analizar las características de las secuencias promotoras y reconocer éstas en humanos y ratones. Los resultados evidencian que DeePromoter es superior a CNNProm en base a las especies evaluadas.

Chevez-Guardado & Peña-Castillo (2021) proponen Promotech [17], un método de reconocimiento de promotores entrenado en un conjunto de datos de secuencias de promotores de nueve especies y evaluados en otras cuatro especies, mediante un modelo único. Las secuencias con etiqueta negativa se extraen de zonas del genoma que no incluyan promotores reportados previamente. Se evaluaron distintas representaciones matemáticas de secuencias *(encodings)* y modelos de Machine Learning como Random Forest y Deep Learning como LSTM y *Gated Recurrent Unit (GRU)*, con el objetivo de encontrar la configuración que tenga mejores métricas en términos de AUC y área bajo la curva Precision-Recall *(Area under PRC-curve, AUPRC)*. Se probaron codificaciones como One-Hot, frecuencia de tetranucleótidos y un vector de características generados mediante una CNN. Además, comparan los resultados de las métricas de las configuraciones con métodos del estado del arte, en donde las técnicas basadas en Random Forest superaron a estos últimos. No obstante, para 3 especies de prueba se obtuvieron valores por debajo de 0.7 tanto de AUC como de AUPRC,

lo que evidencia un desempeño regular.

Zhang et. al. (2022) proponen iPro-WAEL [18], un modelo basado en *ensemble learning*, utilizando métodos basados en Random Forest y CNN que en conjunto realizan la predicción. Se evaluaron distintas formas de representación matemática de la secuencia nucleotídica, como Word2Vec [26], RCKmer [27] y CKSNAP [28]. El modelo es entrenado con secuencias promotoras de humanos y evaluado en 7 especies, desarrollando un modelo por cada especie. Utiliza como secuencias negativas porciones del genoma humano que tengan una alta similitud con secuencias promotoras. En la evaluación, es comparado con otros métodos del estado del arte, en donde iPro-WAEL los supera en términos de *Accuracy*, área bajo la curva ROC *(Area under ROC-curve, AUC)* y coeficiente de correlación de Matthews *(Matthews correlation coefficient, MCC)*. Los resultados evidencian que iPro-WAEL tiene buen desempeño pero mejorable en la mayoría de las especies, reflejado en valores superiores a 0.80 en las métricas mencionadas.

Zhu et. al. (2023) publicaron TIMER [16], un enfoque basado en redes neuronales siamesas para identificar promotores tanto mediante un modelo único como uno por especie. TIMER utiliza secuencias de ADN como entrada y emplea tres redes neuronales siamesas con capas *Attention* para entrenar y optimizar los modelos, utilizando promotores de 13 especies distintas y fragmentos del genoma de cada especie (exceptuando promotores). Como la cantidad de fragmentos es mayor a la cantidad de promotores disponibles, se realizan operaciones de filtrado de redundancia con el fin de crear un dataset sin secuencias redundantes. Los resultados reportaron que supera en métricas como *Accuracy* y *F1-Score* a los modelos del estado del arte, incluyendo iPro-WAEL y Promotech.

2.3. Sesgo de contenido GC en modelos de una especie

En base a la literatura, se evidencia que no hay un consenso sobre el origen de secuencias consideradas como negativas, en el contexto de un clasificador binario. En esta línea, Cassiano et. al. (2020) realizaron una comparación sistemática de varias herramientas de predicción de promotores para la especie *E.coli* [12], en donde las secuencias negativas fue-

ron generadas aleatoriamente, pero con una distribución de bases nitrogenadas de adenina (A) y timina (T) similar a la de los promotores. Los resultados evidenciaron que modelos como CNNProm, iPro70-FMWin y 70ProPred son capaces de realizar la detección con un rendimiento regular, reflejado en valores de accuracy entre 0.72 y 0.76. Además, se detectó que dichos modelos arrojaron valores entre 0.51 y 0.69 en especificidad, sugiriendo que éstos son propensos a detectar falsos positivos. Los autores relacionan este resultado debido a que hay un sesgo de poca cantidad de A y T en las regiones consideradas como secuencias negativas en los modelos, evidenciado por la alta variabilidad en un rango de valores deficientes y regulares de especificidad. En base a este trabajo, se sugiere la relevancia del contenido GC, definido como la proporción de bases citosina (C) y guanina (G) presentes en toda la cadena, en la construcción de un dataset de secuencias negativas.

2.4. Large Language Models en datos genómicos

En el último tiempo, en el área de procesamiento de lenguaje natural *(natural language processing, NLP)* se ha hecho popular el uso de la arquitectura de Transformers [19], que, en base al mecanismo de *Attention*, es capaz de capturar la relación entre elementos secuenciales que están lejos unos de otros, además de ser capaz de entrenar utilizando paralelización. Luego, en base a la arquitectura de Transformers, Devlin et al. (2018) proponen BERT [29], que permite obtener representaciones contextuales en cadenas de texto. El uso de BERT involucra un entrenamiento para generar entendimientos generales a partir de una gran cantidad de datos no etiquetados, para luego aplicar la técnica de *fine-tuning*, que consiste en usar BERT preentrenado para resolver problemas particulares con modificaciones mínimas en la arquitectura.

En el contexto de detección de promotores, Ji et al. (2022) proponen DNABERT [20], un modelo preentrenado con promotores humanos que permite obtener representaciones matemáticas de las secuencias para utilizarlas en diversas tareas en el campo de la biotecnología, incluyendo la detección de promotores. DNABERT emplea una representación basada en *k-mers* para la *tokenización*, lo que le permite capturar información contextual de las

secuencias de nucleótidos tanto aguas arriba como aguas abajo de manera más efectiva. Este modelo logró un alto desempeño en comparación con DeePromoter en la detección de promotores en humanos, obteniendo valores superiores a 0.92 en métricas como *Accuracy*, F1-Score y MCC.

Posteriormente, los mismos autores publicaron en 2023 DNABERT-2, que tiene el mismo propósito que su predecesor, pero introduce varias mejoras significativas. DNABERT-2 reemplaza la *tokenización* basada en *k-mers* con la técnica de Byte Pair Encoding (BPE) [30], e integra técnicas avanzadas como Attention with Linear Biases (ALiBi) [31] y reemplaza el mecanismo de *Attention* original por Flash Attention para mejorar la eficiencia computacional y manejar entradas más largas. Además, incorpora la técnica de Low-Rank Adaptation (LoRA) [32] para una adaptación más eficiente de los parámetros durante las etapas de *fine-tuning*. Aunque DNABERT-2 presenta ventajas en términos de tamaño de parámetros, tiempo y gasto de memoria, se evidencia que su predecesor tiene un mejor desempeño en términos de MCC para la tarea de clasificación de promotores en humanos.

Por otro lado, el *Nucleotide Transformer* [22] utiliza una estrategia de tokenización basada en *6-mers*, específicamente diseñada para predecir con precisión fenotipos moleculares y analizar secuencias de nucleótidos, incluso en entornos con datos limitados.

2.5. Datasets de promotores disponibles

La detección de promotores es un paso esencial en el estudio de la regulación génica, ya que estos elementos definen los sitios donde inicia la transcripción de los genes. Para facilitar esta tarea, existen bases de datos que compilan información detallada y experimentalmente validada sobre promotores, permitiendo el desarrollo de modelos computacionales más precisos y aplicaciones biotecnológicas. A continuación, se describen dos de las bases de datos más relevantes:

- RegulonDB [24] es una base de datos especializada en la regulación génica de Escherichia coli K-12, destacando por su información detallada sobre promotores. Este

recurso proporciona datos curados manualmente que incluyen la anotación precisa de regiones promotoras, especificando secuencias -10 y -35, sitios de inicio de transcripción (TSS) y espaciadores, además de promotores reconocidos por diferentes factores sigma. En la actualidad, la versión 13.0 de RegulonDB dispone de un total de 4052 promotores anotados.

- Prokaryotic Promoter Database (PPD) [33] es una base de datos recopilada por Su et. al. (2021) que integra promotores de diversas especies con evidencia experimental. Contiene 129.148 promotores de 63 especies procariotas, extraídas a partir de publicaciones en la academia. Además, proveen una interfaz de acceso público para buscar, visualizar y descargar datos.

Capítulo 3

Marco Teórico

3.1. ¿Qué es un promotor?

Dentro de la genómica, existe el proceso de regulación de la transcripción genética, el cual convierte la información del ADN en ARN, un paso esencial antes de la síntesis de proteínas. La regulación genética juega un papel importante, determinando cuándo y cómo se activan los genes para producir ARN. Explorar estos mecanismos requiere comprender términos clave como:

- **Gen:** Una secuencia de ADN compuesta por adenina (A), guanina (G), citosina (C) y timina (T), la cual codifica para una proteína y regula características hereditarias y funciones biológicas.

- **Genoma:** El genoma es el conjunto completo de material genético de un organismo. Incluye tanto los genes, que contienen las instrucciones para las proteínas, como las secuencias de ADN que no codifican proteínas, siendo esencial para la estructura y función del organismo.

- **Promotor:** Un promotor es una región del ADN cercana al inicio de un gen que actúa como un *interruptor* para ese gen. Su tarea principal es controlar cuándo y cómo se inicia el proceso de convertir la información del gen en ARN, un paso previo a la

formación de proteínas en la célula de cualquier organismo. Es homólogo a un *botón de inicio* que la maquinaria celular utiliza para comenzar a leer y usar las instrucciones contenidas en un gen.

- **Secuencias Codificantes** (CDS, *CoDing Sequences*): Partes de un gen cuya secuencia determina la secuencia de aminoácidos para producir proteínas.

El genoma se asemeja a una extensa biblioteca que almacena toda la información genética de un organismo. Cada gen representa un volumen único en esta biblioteca, portador de instrucciones específicas para diversas funciones o características del organismo. Los promotores funcionan a manera de índices o prefacios de estos volúmenes, determinando el momento y la manera en que los genes deben ser leídos o expresados. Las secuencias codificantes en los genes se equiparan a capítulos fundamentales en cada volumen, proporcionando instrucciones detalladas que se convertirán en acciones o productos concretos, como las proteínas.

3.2. Modelos de clasificación de texto en NLP

En el campo del procesamiento de lenguaje natural (NLP), los modelos de clasificación de texto son herramientas esenciales que permiten a las máquinas comprender y categorizar grandes volúmenes de texto de manera eficiente y efectiva. Estos modelos encuentran aplicaciones en diversas áreas, como el análisis de sentimientos [34], la detección de spam [35] y la clasificación de documentos [36]. Para entrar en detalle, es necesario introducir conceptos como **tokenización** de texto, la conversión de tokens en vectores a través de **embeddings** y la utilización de estos en un modelo de clasificación para predecir etiquetas o categorías. Un flujo general se muestra en la Figura 3.1.

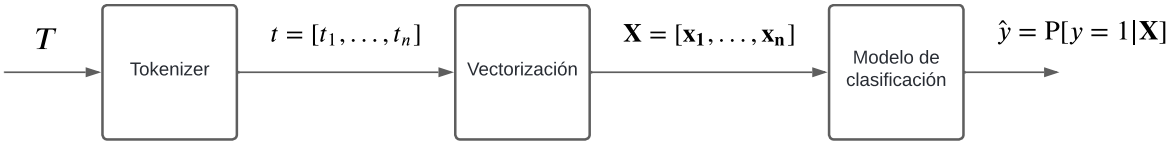


Figura 3.1: Flujo general de clasificación de texto en NLP.

A continuación se explica cada proceso:

- **Tokenizer:** La tokenización es el proceso de dividir un texto en unidades más pequeñas, conocidas como tokens. Estos pueden ser palabras, frases o incluso caracteres. Dicho proceso recibe el texto completo T, dividiéndolo en n tokens $t = [t_1, ..., t_n]$.

  Tipos de Tokenización

  - **Tokenización por Palabras:** Este método es ideal para idiomas que muestran una separación clara de palabras, como el inglés. Funciona dividiendo el texto en unidades basadas en espacios y signos de puntuación. Por ejemplo, el texto `"Hello world"` se puede representar como `['Hello', 'World']`.
  - **Tokenización por Caracteres:** Útil en idiomas donde no hay separación clara entre palabras o para análisis que requieren un nivel de detalle muy fino, como en el procesamiento de textos antiguos o en idiomas como el chino. Este método divide el texto en sus caracteres individuales. Por ejemplo, la palabra `"hello"` se puede representar como `['h', 'e', 'l', 'l', 'o']`.
  - **Tokenización por Subpalabras:** Descompone las palabras en fragmentos que son más pequeños que las palabras completas pero mayores que los caracteres individuales. Este enfoque es particularmente útil para manejar vocabularios extensos y para procesar palabras desconocidas, reduciendo la cantidad de información necesaria para entrenar modelos de lenguaje eficientes.

- **Vectorización:** es el proceso donde se mapea cada token $t_i \in t$ en un vector $\mathbf{x}_i \in \mathbb{R}^d$, donde d es la dimensión del vector. Estos vectores permiten la representación numérica de los tokens, facilitando el procesamiento de los datos para el modelo de clasificación.

  Tipos de Vectorización

  - **Encodings:** Los encodings transforman tokens o palabras en vectores numéricos de una forma directa y generalmente no densa. Ejemplos comunes incluyen:
    - **One-Hot Encoding:** Cada palabra en el vocabulario es representada por un vector donde solo un elemento es '1' y todos los demás son '0'. Por

ejemplo, para un vocabulario `['cat', 'dog', 'bird']`, la palabra `'dog'` sería representada como $[0, 1, 0]$.

- **Label Encoding:** Cada palabra se asigna a un número entero único. Este método es útil para algoritmos que necesitan etiquetas numéricas pero no captura información semántica entre las palabras.

- **Embeddings:** Los embeddings son vectores densos y de baja dimensión que capturan no solo la presencia de palabras, sino también sus relaciones semánticas y contextuales. Son especialmente útiles en modelos complejos de procesamiento de lenguaje natural.

  - **Word2Vec [26]:** Este método aprende representaciones vectoriales para palabras basadas en su contexto en grandes corpus de texto. Por ejemplo, palabras como `['rey, 'reina', 'hombre', 'mujer']` podrían tener relaciones semánticas capturadas por los vectores, de modo que el resultado de `rey - hombre + mujer` se aproxima al vector de `reina`.

  - **FastText [37]:** Una extensión de Word2Vec que incluye subpalabras en su modelo, lo que permite manejar palabras desconocidas o con errores tipográficos. Por ejemplo, para la palabra `'running'`, se consideran subpalabras como `'run', 'unn', 'nni', 'ing'`, mejorando las representaciones de palabras compuestas.

  - **BERT (Bidirectional Encoder Representations from Transformers) [29]:** Este modelo genera embeddings contextuales, donde el vector de una palabra varía según su uso en diferentes oraciones. Por ejemplo, la palabra `'bank'` tendría un embedding diferente en las frases `'I sat on the river bank'` y `'I need to go to the bank to deposit money'` dado que el contexto de las frases son distintos.

- **Modelo de clasificación:** Utilizando la vectorización del texto de entrada $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, el modelo de clasificación aprende a asignar una probabilidad de que el texto T corresponda a una categoría específica. Sea y la variable aleatoria que

representa la clase del texto, donde $y = 1$ indica que el texto pertenece a la clase de interés e $y = 0$ indica que no pertenece a la clase de interés. El modelo estima la probabilidad $P[y = 1|X]$, que es la probabilidad condicional de que el texto pertenezca a la clase de interés el texto de entrada vectorizado X.

3.2.1. Representación Numérica de Secuencias de ADN

En el análisis de secuencias de ADN, los encodings y las técnicas de vectorización permiten transformar cadenas de nucleótidos en representaciones numéricas procesables por modelos computacionales. Mientras los encodings capturan la identidad directa de los nucleótidos, las vectorizaciones como los embeddings incorporan relaciones contextuales, facilitando la detección de patrones complejos y mejorando el rendimiento en tareas como clasificación de promotores. A continuación, se ejemplifican las técnicas más comunes utilizadas en este ámbito.

1. **One-Hot Encoding:** Representa cada nucleótido como un vector binario donde solo un elemento es 1, indicando su presencia. Los nucleótidos se codifican como $A \to (1, 0, 0, 0)$, $C \to (0, 1, 0, 0)$, $G \to (0, 0, 1, 0)$, $T \to (0, 0, 0, 1)$. Por ejemplo, la representación one-hot $\mathbf{X}_{\text{one-hot}}$ de la secuencia `ACGTAC` se codifica como:

$$\mathbf{X}_{\text{one-hot}} = \begin{bmatrix} 1,0,0,0 \\ 0,1,0,0 \\ 0,0,1,0 \\ 0,0,0,1 \\ 1,0,0,0 \\ 0,1,0,0 \end{bmatrix}.$$

   Existe también la variante de una dimensión, donde la secuencia completa se representa concatenando los vectores, resultando en:

$$\mathbf{X}_{\text{one-hot-1D}} = (1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,1,0,0).$$

2. **Frecuencia de Tetranucleótidos [38]:** Calcula las frecuencias de todas las combinaciones de 4 nucleótidos consecutivos, capturando patrones locales. Para `ACGTAC`, los tetranucleótidos son: $\{\texttt{ACGT}, \texttt{CGTA}, \texttt{GTAC}\}$ con frecuencias normalizadas $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.

3. **Frecuencia de k-mers [39]:** Calcula las frecuencias de todas las combinaciones de k-nucleótidos consecutivos en una secuencia, capturando patrones locales que dependen del valor de k. Para una secuencia de longitud n, el número de k-mers posibles es $n - k + 1$. Por ejemplo, para $k = 3$ en la secuencia `ACGTAC`, los 3-mers consecutivos corresponden a $\{\texttt{ACG}, \texttt{CGT}, \texttt{GTA}, \texttt{TAC}\}$ y cada uno aparece una vez en la secuencia. Por lo tanto, las frecuencias normalizadas (relativas) $\mathbf{f}_{\text{3-mers}}$ corresponden a:

$$\mathbf{f}_{\text{3-mers}} = (0{,}25, 0{,}25, 0{,}25, 0{,}25).$$

3.2.2. Evaluación del Error de Predicción

Para evaluar el desempeño de un modelo de clasificación binaria, se utiliza comúnmente la función de pérdida Binary Cross Entropy. Esta métrica mide la discrepancia entre las predicciones del modelo, $\hat{y} = P[y = 1 \mid \mathbf{X}]$, y las etiquetas verdaderas $y \in \{0, 1\}$ y está definida por:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{m} \sum_{i=1}^{m} \Big(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big),$$

donde m es el número de muestras en el conjunto de datos, y_i es la etiqueta verdadera para la i-ésima muestra y $\hat{y}_i = P[y = 1 \mid \mathbf{X}_i]$ es la probabilidad predicha para la clase positiva.

3.2.3. Métricas de evaluación

En la evaluación de modelos de clasificación, se utilizan métricas estándar [40] para medir el rendimiento de los modelos. Estas métricas permiten evaluar de manera completa la capacidad del modelo para clasificar correctamente los ejemplos positivos y negativos. A continuación, se explican las utilizadas en este trabajo:

Especificidad (Sp): También conocida como tasa de falsos positivos (False Positive Rate, FPR). Mide la proporción de verdaderos negativos correctamente identificados, es decir, la capacidad del modelo para identificar correctamente las instancias negativas.

$$\mathrm{Sp} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Sensibilidad (Sn): También conocida como recall o tasa de verdaderos positivos (True Positive Rate, TPR), mide la proporción de verdaderos positivos correctamente identificados, es decir, la capacidad del modelo para detectar correctamente las instancias positivas.

$$\mathrm{Sn} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Precisión (Pre): Indica la proporción de verdaderos positivos entre todas las instancias clasificadas como positivas, es decir, la exactitud de las predicciones positivas del modelo.

$$\mathrm{Pre} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Accuracy (Acc): Mide la proporción de predicciones correctas (tanto positivas como negativas) entre todas las instancias evaluadas.

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Coeficiente de correlación de Matthews (MCC): Es una métrica que considera el equilibrio entre las diferentes clases, proporcionando una medida más completa del rendimiento del modelo. Un MCC de 1 indica una predicción perfecta, 0 indica un rendimiento

aleatorio, y -1 indica una predicción completamente opuesta a los resultados reales.

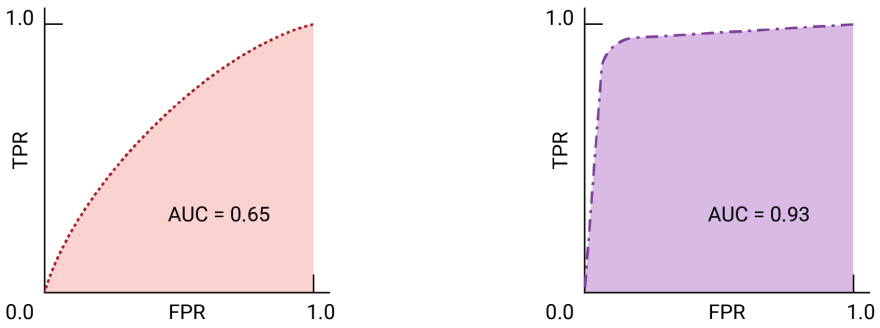
$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

F1 Score (F_1): Es la media armónica de la precisión y la sensibilidad, proporcionando una medida balanceada del rendimiento del modelo, especialmente útil en casos de clases desequilibradas.

$$\text{F1} = 2 \times \frac{\text{Pre} \times \text{Sn}}{\text{Pre} + \text{Sn}}$$

Área bajo la curva ROC (ROC AUC): Mide la capacidad del modelo para distinguir entre clases positivas y negativas en función de diferentes umbrales, proporcionando una evaluación general del rendimiento del modelo. Una puntuación de 1 indica un modelo perfecto, mientras que una puntuación de 0.5 indica un rendimiento similar al azar.

En la Figura 3.2, se ejemplifican curvas ROC y el área bajo la curva (AUC). El eje horizontal representa la tasa de falsos positivos (FPR), mientras que el eje vertical muestra la tasa de verdaderos positivos (TPR). Cuanto mayor es el área bajo la curva, mejor es el rendimiento del modelo en la clasificación.



(a) Curva ROC con AUC = 0.65 (b) Curva ROC con AUC = 0.93

Figura 3.2: Ejemplo de curvas ROC para dos modelos. El modelo (a) presenta una clasificación regular, mientras que el modelo (b) muestra un rendimiento superior.

3.3. Modelos empleados para detección de promotores

3.3.1. Random Forest

El algoritmo de Random Forest es una técnica de aprendizaje supervisado ampliamente utilizada tanto para problemas de clasificación como de regresión. Fue introducido por Breiman en 2001 [41] y se basa en el concepto de ensemble learning, el cual combina múltiples modelos base (en este caso, árboles de decisión) para mejorar la precisión y reducir el riesgo de sobreajuste (overfitting).

En esencia, un modelo de Random Forest está compuesto por un conjunto de árboles de decisión independientes con un enfoque conocido como bootstrap aggregating, en donde cada árbol es construido utilizando un subconjunto aleatorio de los datos de entrenamiento y un subconjunto aleatorio de las características disponibles, disminuyendo la correlación entre ellos y permitiendo capturar diferentes patrones presentes en los datos. Esta estrategia introduce una mayor diversidad en el modelo, mejorando su desempeño y reduciendo la probabilidad de sobreajuste.

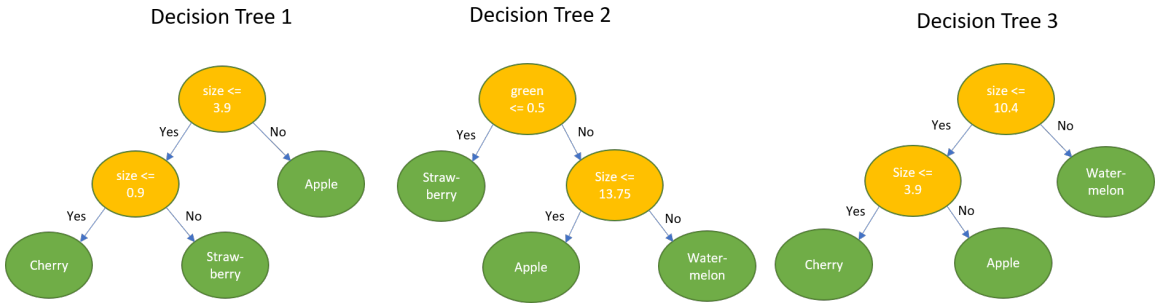


Figura 3.3: Ejemplo de Random Forest

Una vez que se obtienen las predicciones individuales de todos los árboles, el modelo las combina mediante un proceso de agregación para generar el resultado final. En problemas de clasificación, se toma la clase más votada entre los árboles como la predicción final, mientras que en tareas de regresión se calcula el promedio de las predicciones. Esta forma de integración hace que el Random Forest sea más robusto frente al sobreajuste en comparación con un único árbol de decisión, ya que aprovecha la diversidad de los árboles independientes para

generalizar mejor a nuevos datos, incluso en presencia de ruido o características irrelevantes.

3.3.2. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (CNN) son un tipo de arquitectura neuronal diseñada para procesar datos estructurados en forma de cuadrículas, como imágenes bidimensionales o señales tridimensionales. Su arquitectura está basada en operaciones convolucionales, que consisten en la aplicación de filtros deslizantes (*kernels*) sobre la entrada para extraer características locales. A través de una jerarquía de capas, las CNNs extraen características de bajo nivel en las primeras capas y características más abstractas en las capas posteriores. Este tipo de arquitectura ha mostrado resultados prometedores para tareas como detección de objetos y reconocimiento facial, análisis de sentimientos en texto y en la detección de promotores.

Su funcionamiento se basa en tres etapas principales: extracción de características, clasificación y predicción. En la etapa de extracción, las capas convolucionales aplican filtros (kernels) a la entrada mediante operaciones de convolución, produciendo mapas de características que destacan patrones locales, como bordes o texturas. Estas capas pueden incluir hiperparámetros como el tamaño del kernel (por ejemplo, 3×3) y el stride, que determina el desplazamiento del filtro. Para reducir la dimensionalidad y evitar el sobreajuste, se utilizan operaciones de max-pooling, que seleccionan el valor máximo dentro de regiones definidas (por ejemplo, 2×2) con un stride correspondiente. Tras esta extracción jerárquica de características, los mapas resultantes se aplanan y se pasan a capas densas (fully connected), que combinan las características aprendidas para modelar relaciones más complejas. Finalmente, una capa de salida, generalmente con una función de activación como softmax, asigna probabilidades a cada clase, permitiendo así la clasificación. Un ejemplo de esta arquitectura se puede ver en la Figura 3.4 (Fuente: [42]).

Una variante interesante de esta arquitectura es la CNN en una dimensión, donde tanto la entrada como las operaciones de convolución se realizan en una sola dimensión, lo que la hace ideal para procesar datos secuenciales como series temporales o secuencias genómicas [8, 15].

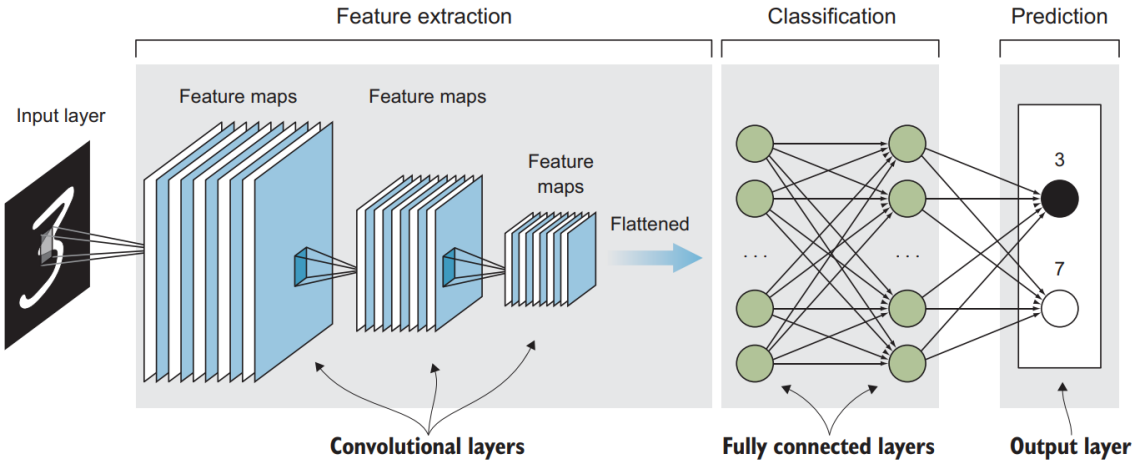


Figura 3.4: Arquitectura de una CNN para clasificación de números, destacando la extracción de características, capas densas y predicción final.

En lugar de capturar patrones espaciales bidimensionales como en las imágenes, las CNN en una dimensión se enfocan en detectar dependencias locales a lo largo de una única dimensión. Las operaciones de convolución son similares a las de las CNN tradicionales, aplicando filtros deslizantes sobre los datos, pero adaptadas a trabajar con una sola dimensión, lo que les permite extraer características relevantes en ventanas consecutivas dentro de las secuencias. Un ejemplo de esta variante se puede ver en la Figura 3.5 (Fuente: [43]).

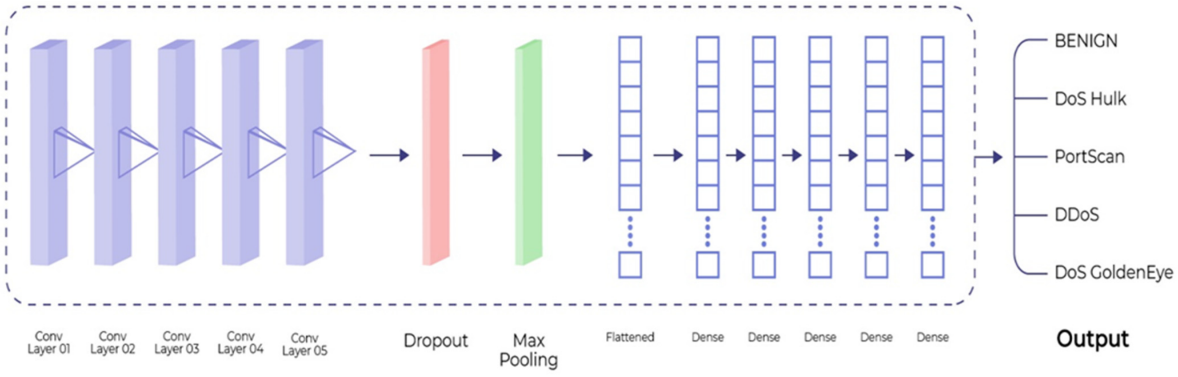


Figura 3.5: Ejemplo de una CNN en una dimensión diseñada para la detección de amenazas en redes, capaz de clasificar el tráfico como benigno o asociado a diversos tipos de ataques.

3.3.3. Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) es un modelo de lenguaje basado en Transformers, presentado por Devlin et al. en 2018, diseñado para capturar representaciones contextuales profundas mediante un entrenamiento bidireccional. A diferencia de los modelos secuenciales tradicionales o arquitecturas unidireccionales, BERT emplea exclusivamente el bloque de encoders de la arquitectura original de Transformers, lo que le permite considerar de manera simultánea el contexto tanto de izquierda a derecha como de derecha a izquierda en una secuencia.

3.3.3.1. Transformers

Los Transformers son una arquitectura de red neuronal propuesta por Vaswani et al. en 2017, diseñada para abordar tareas secuenciales como las del procesamiento del lenguaje natural. A diferencia de modelos tradicionales como las Redes Neuronales Recurrentes (RNNs) [44] y las Long Short-Term Memory Networks (LSTM) [45], los Transformers no procesan las secuencias de forma recurrente. En su lugar, utilizan una arquitectura basada en el mecanismo de *attention*, que permite el procesamiento paralelo de toda la secuencia, reduciendo el tiempo de cálculo.

La arquitectura Transformer (Figura 3.6) se compone principalmente de dos bloques: **encoder** y **decoder**. El bloque encoder se encarga de transformar la entrada en una representación intermedia que captura relaciones contextuales, mientras que el bloque decoder utiliza dicha representación para generar la salida. Ambos bloques están formados por capas que incluyen otros bloques como de atención multicabeza (Multi-Head Attention) (Figura 3.7), redes feed-forward completamente conectadas, mecanismos de normalización por capas y embeddings (input embeddings). A continuación, se describen los bloques más relevantes de esta arquitectura.

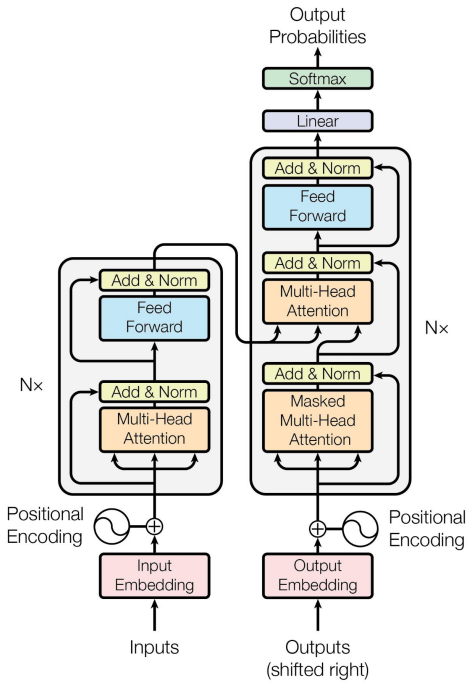
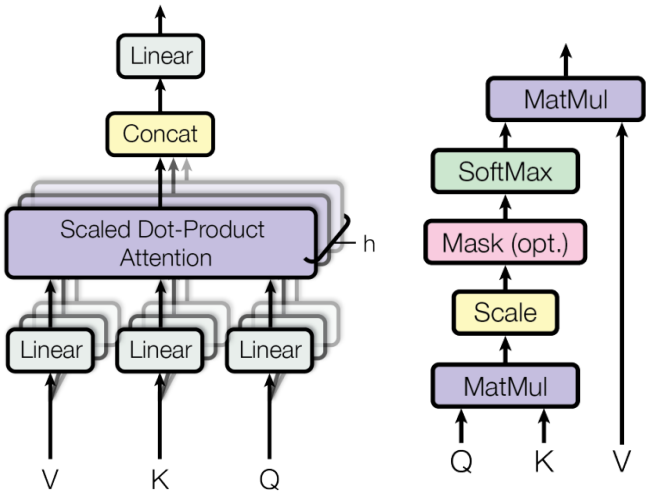


Figura 3.6: Arquitectura del modelo Transformer mostrando los bloques de encoder (izquierda) y decoder (derecha).



(a) Multi-head Attention (b) Scaled Dot Product

Figura 3.7: Ejemplo de los bloques de atención en el modelo Transformer.

Input Embedding

El input embedding se define como la suma de dos vectores asociados a cada posición i en la secuencia de entrada: Word Embedding y Positional Embedding, definida como:

$$E_i = W_i + P_i$$

Donde E_i es el embedding de entrada en la posición i, W_i es el Word Embedding correspondiente a la palabra o token en la posición i y P_i es el Positional Embedding que codifica la posición i en la secuencia.

El Word Embedding (W_i) es un vector de dimensión fija d_{model} que representa la palabra en un espacio continuo. Estos vectores se obtienen de una tabla de embeddings aprendida durante el entrenamiento (EmbeddingLookup). Formalmente:

$$W_i = \text{EmbeddingLookup}(V, x_i)$$

Donde $V \in \mathbb{R}^{|Vocab| \times d_{model}}$ es la matriz de embeddings, siendo $|Vocab|$ el tamaño del vocabulario y x_i es el índice del token correspondiente en el vocabulario.

El Positional Encoding (P_i) se calcula para cada posición i utilizando funciones trigonométricas seno y coseno. En particular, las componentes individuales del embedding posicional están definidas por:

$$P_{i,2k} = \sin\left(\frac{i}{10000^{\frac{2k}{d}}}\right), \quad P_{i,2k+1} = \cos\left(\frac{i}{10000^{\frac{2k}{d}}}\right)$$

Donde i es la posición de la secuencia, k es el índice de la dimensión (pares e impares se calculan con funciones seno y coseno, respectivamente) y $d = d_{model}$ es la dimensión total del embedding. Este diseño utiliza funciones trigonométricas periódicas, donde las frecuencias bajas representan relaciones globales entre posiciones distantes y las frecuencias altas capturan relaciones locales entre posiciones cercanas. Esto permite que diferentes dimensiones del vector de embedding codifiquen patrones posicionales con distintos niveles de granularidad, facilitando que el modelo identifique relaciones relativas entre palabras

dentro de la secuencia.

Scaled Dot-Product Attention

El mecanismo de Scaled Dot-Product Attention utiliza las matrices $Q \in \mathbb{R}^{n \times d_k}$ (queries), $K \in \mathbb{R}^{n \times d_k}$ (keys) y $V \in \mathbb{R}^{n \times d_v}$ (values). El cálculo se define como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

El producto escalar $QK^\top \in \mathbb{R}^{n \times n}$ mide la similitud entre cada query y todos los keys. Estas similitudes se convierten en pesos probabilísticos mediante la función softmax, que asigna a cada key un peso entre 0 y 1 basado en su relevancia relativa para la query correspondiente. Los pesos se normalizan fila por fila, de manera que la suma de los pesos asociados a una query sea igual a 1. Estos pesos probabilísticos se utilizan para ponderar los valores en $V \in \mathbb{R}^{n \times d_v}$, produciendo una salida de dimensión $n \times d_v$. Esto permite al modelo construir representaciones enriquecidas que enfatizan las partes más relevantes del contexto de cada palabra o token en la secuencia.

Multi-Head Attention

El mecanismo de Multi-Head Attention extiende el concepto de Scaled Dot-Product Attention al aplicar múltiples instancias de atención en paralelo llamadas cabezas (*heads*). Esto permite al modelo captar diferentes tipos de relaciones contextuales en distintas subrepresentaciones del espacio de embeddings.

Dado un conjunto de matrices $Q \in \mathbb{R}^{n \times d_{model}}$, $K \in \mathbb{R}^{n \times d_{model}}$ y $V \in \mathbb{R}^{n \times d_{model}}$, el proceso de Multi-Head Attention se realiza dividiendo estas matrices en h subespacios, donde cada cabeza opera sobre un subespacio reducido con dimensión $d_k = d_{model}/h$. El cálculo para cada cabeza i se define como:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

donde $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ y $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ son matrices de proyección

aprendidas que transforman Q, K y V en subespacios específicos para la cabeza i.

Luego de calcular las salidas de todas las cabezas, estas se concatenan y se proyectan de vuelta a la dimensión original mediante una matriz aprendida $W^O \in \mathbb{R}^{(h \cdot d_v) \times d_{model}}$:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$$

El uso de múltiples cabezas permite que el modelo divida su capacidad de atención entre diferentes aspectos del contexto. Algunas cabezas pueden enfocarse en dependencias locales entre palabras cercanas, mientras que otras pueden capturar relaciones globales entre palabras distantes. Además, el cálculo para cada cabeza se puede realizar de forma paralela, mejorando la eficiencia computacional.

3.3.3.2. Arquitectura BERT

En términos de arquitectura, BERT está compuesto únicamente por una pila de bloques encoder conectados de manera secuencial, donde la salida de un bloque se utiliza como entrada del siguiente. Cada bloque en la pila refina progresivamente las representaciones del texto al combinar información contextual de todas las palabras en la secuencia. A diferencia de los Transformers originales, que utilizan Positional Encodings calculados mediante funciones trigonométricas para introducir información sobre la posición de los tokens, BERT emplea Positional Embeddings que son aprendidos durante el entrenamiento. Estos embeddings se optimizan junto con los demás parámetros del modelo, lo que permite a BERT adaptarse mejor a los patrones posicionales relevantes en las tareas de comprensión del lenguaje.

Por otro lado, BERT incorpora tokens especiales que no están presentes en los Transformers genéricos, como `[CLS]`, que se utiliza para representar la información global de la secuencia completa, y `[SEP]`, que separa oraciones o marca el final de una secuencia. Estos tokens son esenciales para estructurar las entradas del modelo y facilitar el aprendizaje de relaciones tanto dentro de una secuencia como entre oraciones. Además, se introducen otros tokens como `[MASK]` y `[PAD]`, utilizados en diferentes fases del proceso de entrenamiento.

El entrenamiento de BERT se divide en dos fases principales: preentrenamiento y fine-tuning. En la fase de preentrenamiento, BERT aprende representaciones generales del lenguaje al procesar grandes volúmenes de texto no etiquetado, capturando relaciones bidireccionales entre palabras y oraciones. Esto significa que BERT analiza simultáneamente el contexto tanto anterior como posterior de cada palabra en una secuencia, permitiéndole comprender el lenguaje de forma más completa. En la fase de fine-tuning, el modelo se ajusta a tareas específicas como clasificación, como respuesta a preguntas o etiquetado de entidades. Para ello, se utilizan datos etiquetados de la tarea objetivo, ajustando los parámetros aprendidos durante el preentrenamiento para optimizar el rendimiento en aplicaciones concretas. A continuación, se describen ambas fases.

3.3.3.3. Entrenamiento

El entrenamiento de BERT comienza con una **fase de preentrenamiento**, basada en dos tareas principales: Masked Language Modeling (MLM) y Next Sentence Prediction (NSP). El objetivo de MLM es permitir que el modelo aprenda representaciones profundas del lenguaje al predecir palabras enmascaradas dentro de una secuencia, utilizando el contexto bidireccional. Por otro lado, NSP busca que el modelo capture relaciones semánticas y discursivas entre pares de oraciones, ya sean consecutivas o no relacionadas. Estas dos tareas permiten que BERT desarrolle una comprensión general y versátil del lenguaje natural, que luego puede ser ajustada para aplicaciones específicas. Después del preentrenamiento, BERT pasa por una **fase de fine-tuning**, donde el modelo preentrenado se adapta a tareas específicas como clasificación de texto, respuesta a preguntas o etiquetado de entidades. Este proceso permite que BERT combine su conocimiento general del lenguaje con una especialización precisa en el problema objetivo. A continuación se describen las tareas de cada fase.

Masked Language Modeling (MLM)

En la etapa de Masked Language Modeling, aproximadamente el 15 % de las palabras en cada secuencia de entrada se seleccionan aleatoriamente para ser enmascaradas y reemplazadas

por un token especial `[MASK]`. De las palabras seleccionadas, el 80 % se reemplazan por dicho token, el 10 % se dejan sin cambios y el 10 % se sustituyen por una palabra aleatoria del vocabulario. Este proceso permite al modelo aprender a predecir las palabras enmascaradas utilizando el contexto bidireccional generado por el encoder. El objetivo principal de MLM es forzar al modelo a comprender relaciones contextuales complejas en ambas direcciones, haciendo que las representaciones aprendidas sean más profundas y generalizables para tareas específicas en el fine-tuning.

Un ejemplo se ilustra en la Figura 3.8, en donde la secuencia de entrada incluye una frase donde ciertas palabras son enmascaradas utilizando el token especial `[MASK]`, como `long` y `thanks`. Estas palabras enmascaradas se pasan junto con el resto de la secuencia a través del encoder bidireccional de BERT, que genera representaciones contextuales profundas para cada token, utilizando información de todo el contexto de la secuencia. A partir de estas representaciones, el modelo aplica un softmax sobre el vocabulario para predecir las palabras enmascaradas, esperando que `[MASK]` sea reemplazado correctamente por `long` y `thanks` en base al contexto. Finalmente, el modelo calcula la pérdida comparando las predicciones con las palabras reales mediante la función de pérdida Cross Entropy Loss.

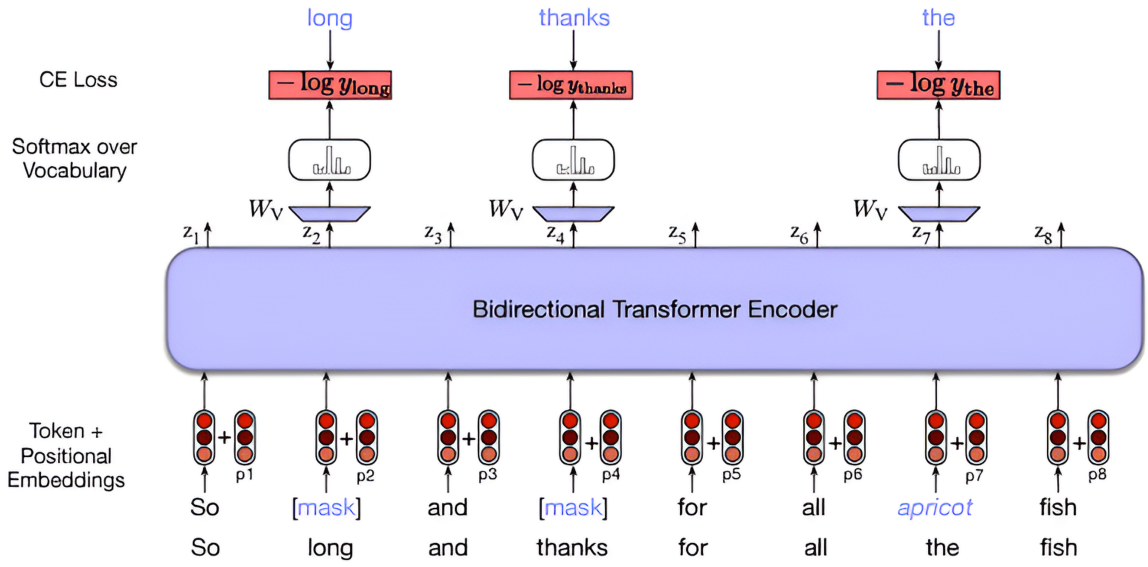


Figura 3.8: Ejemplo de Masked Language Modeling.

Next Sentence Prediction (NSP)

En esta tarea, el modelo recibe pares de oraciones, donde una proporción del 50 % de los pares contiene oraciones consecutivas reales (la segunda oración sigue lógicamente a la primera en el texto original) y el otro 50 % está formado por oraciones aleatorias que no tienen relación entre sí. El objetivo es que el modelo aprenda a identificar si una oración sigue a la otra en el contexto original, capturando relaciones semánticas y discursivas entre oraciones. Para lograrlo, BERT toma las dos oraciones como entrada, separadas por el token `[SEP]`, y utiliza el token `[CLS]` para representar la relación global entre ambas. La predicción se realiza a través de una capa de clasificación que indica si las oraciones son consecutivas o no, y la pérdida se calcula utilizando *Cross-Entropy Loss* para ajustar los parámetros del modelo. En el ejemplo de la Figura 3.9 se presentan al modelo dos oraciones: `Cancel`

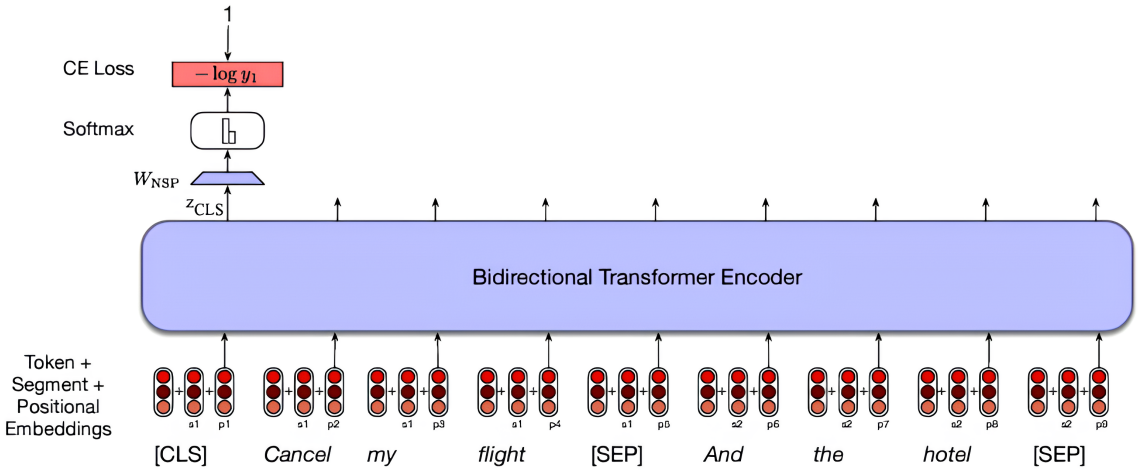


Figura 3.9: Ejemplo de Next Sentence Prediction.

`my flight` y `And the hotel`, separadas por el token especial `[SEP]`. La entrada incluye también el token `[CLS]`, cuya representación final es utilizada para determinar la relación entre ambas oraciones. A través de las capas, el modelo genera representaciones contextuales para cada token, incluyendo `[CLS]`. Posteriormente, esta representación se pasa a una capa de clasificación, que predice si la segunda oración sigue lógicamente a la primera. La salida se evalúa con la función de pérdida Cross-Entropy Loss, ajustando los parámetros del modelo para mejorar su capacidad de capturar relaciones semánticas y discursivas entre oraciones

consecutivas y no relacionadas.

Fine tuning

En esta etapa, el modelo preentrenado se ajusta para tareas específicas, utilizando datos etiquetados relacionados con el objetivo deseado. En esta fase, no solo se ajustan los pesos de BERT, sino también los de una capa adicional específica para la tarea, como una capa lineal en tareas de clasificación. Esta capa se entrena junto con el resto del modelo para garantizar que las representaciones aprendidas durante el preentrenamiento se adapten a las necesidades de la tarea. El entrenamiento supervisado permite que BERT utilice su conocimiento general del lenguaje para capturar patrones específicos del problema objetivo. En el ejemplo de

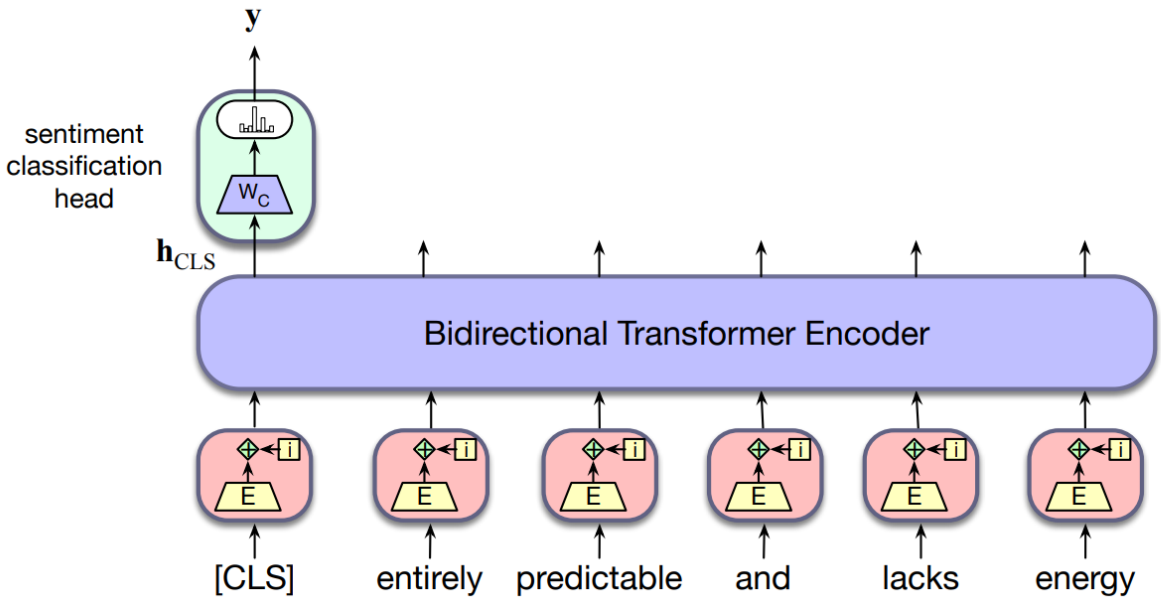


Figura 3.10: Ejemplo de fine tuning en BERT.

la Figura 3.10, BERT se ajusta para una tarea de clasificación de sentimiento, donde se evalúa si una oración tiene una connotación positiva o negativa. La secuencia de entrada incluye el token especial `[CLS]` al inicio, seguido por los tokens de la oración `entirely predictable and lacks energy`. El token `[CLS]` actúa como un vector descriptor de la secuencia completa en un espacio de alta dimensionalidad, ya que captura la información global contextualizada a través del encoder bidireccional. Este vector h_{CLS} ingresa como

entrada a una capa lineal con pesos W_c que se entrena desde cero junto con los pesos de BERT, con el fin de predecir la etiqueta correspondiente al sentimiento. Finalmente, se utiliza una función softmax sobre la salida de la capa lineal para calcular las probabilidades de cada clase, permitiendo asignar la etiqueta más probable a la oración.

3.3.3.4. Adaptaciones de BERT para datos genómicos

DNABERT

DNABERT [20] es una adaptación de BERT a datos genómicos y está diseñado para abordar problemas específicos en bioinformática relacionados con el análisis de secuencias de ADN. Entre sus principales aplicaciones se encuentran la predicción de sitios de unión de proteínas, la identificación de regiones funcionales del genoma, como promotores y regiones reguladoras, y la clasificación de secuencias genómicas. Al capturar patrones contextuales en el ADN, DNABERT ayuda a descubrir relaciones entre nucleótidos que son importantes para entender mecanismos biológicos y funciones celulares.

En aspectos de arquitectura, DNABERT es similar en la mayoría de los aspectos a BERT, exceptuando adaptaciones clave para trabajar con datos genómicos. Utiliza una tokenización basada en k-mers [39], dividiendo las secuencias de ADN en subsecuencias de longitud fija, en lugar de palabras o subpalabras, para capturar patrones locales específicos del código genético. Además, el proceso de preentrenamiento se modifica significativamente en comparación con la implementación original de BERT, eliminando la tarea de Next Sentence Prediction (NSP), ajustando la longitud de las secuencias y obligando al modelo a predecir k tokens contiguos, lo que lo adapta mejor al escenario del ADN y su estructura inherente.

El modelo DNABERT está disponible en versiones preentrenadas para diferentes tamaños de k-mers (3, 4, 5 y 6) y contiene aproximadamente 86 millones de parámetros.

DNABERT-2

DNABERT-2 [21] es una versión mejorada de DNABERT que busca superar las limitaciones identificadas en su predecesor, especialmente en el análisis de secuencias genómicas largas y

contextos más amplios. Estas limitaciones dificultaban la capacidad del modelo para capturar dependencias complejas y realizar un análisis eficiente en tareas que requerían un mayor contexto genómico.

En aspectos de arquitectura, DNABERT-2 conserva la base de BERT, pero introduce adaptaciones clave para trabajar con datos genómicos de manera más eficiente. Una de las principales diferencias es el uso de SentencePiece [46] en combinación con el algoritmo de Byte Pair Encoding (BPE) [30] para la tokenización, lo que permite construir un vocabulario basado en tokens de longitud variable en lugar de los k-mers de longitud fija utilizados en DNABERT. Según los autores, esto mejora la capacidad del modelo para capturar tanto patrones locales como relaciones globales en las secuencias genómicas, al tiempo que optimiza la eficiencia computacional. Por otro lado, DNABERT-2 reemplaza los embeddings posicionales aprendidos de BERT con Attention with Linear Biases (ALiBi) [31], una técnica que introduce sesgos lineales directamente en el mecanismo de atención, eliminando la necesidad de embeddings posicionales explícitos y permitiendo procesar secuencias más largas.

Nucleotide Transformer

El Nucleotide Transformer [22] es una colección de modelos de lenguaje a gran escala desarrollada por InstaDeep en colaboración con NVIDIA y la Universidad Técnica de Múnich. Con tamaños que oscilan entre 500 millones y 2.5 mil millones de parámetros, estos modelos están preentrenados en secuencias de ADN provenientes de más de 3,200 genomas humanos diversos y 850 genomas de múltiples especies. Su propósito principal es generar representaciones contextuales de secuencias nucleotídicas para predecir fenotipos moleculares de manera precisa, incluso en escenarios con datos limitados, y mejorar la priorización de variantes genéticas funcionales, facilitando la comprensión de la relación entre la información genética y los rasgos observables.

En términos de arquitectura, el Nucleotide Transformer introduce adaptaciones clave en comparación con el BERT original para abordar las particularidades de las secuencias genómicas. Utiliza una tokenización basada en k-mers (con $k = 6$) para dividir las secuencias de ADN en subsecuencias de longitud fija, lo que permite capturar patrones locales

específicos del código genético, en contraste con las palabras o subpalabras utilizadas en BERT. Además, reemplaza los embeddings posicionales aprendidos del modelo original con Rotary Position Embeddings (RoPE) [47], un enfoque que integra información posicional directamente en el mecanismo de *attention*, mejorando la capacidad del modelo para manejar dependencias relativas y procesar secuencias más largas.

Capítulo 4

Metodología

El diseño de un clasificador para la detección de promotores presenta un desafío, principalmente debido a las dificultades para crear un conjunto de datos equilibrado. Mientras que obtener secuencias de promotores (positivas) de bases de datos genómicas anotadas es directo, la formulación de un conjunto adecuado de secuencias no promotoras (negativas) para el entrenamiento del modelo sigue siendo incierta, dado que no existe un enfoque universalmente aceptado como el mejor. Esta investigación estudia dos estrategias para generar muestras negativas: la extracción de secuencias no promotoras a partir de secuencias codificantes (CDS) y la generación artificial de secuencias (*Synthetic Random Sequences*, SRS). Para lograr los objetivos planteados, se propone el marco de trabajo mostrado en la Figura 4.1, compuesto de dos fases: de procesamiento de datos y de entrenamiento.

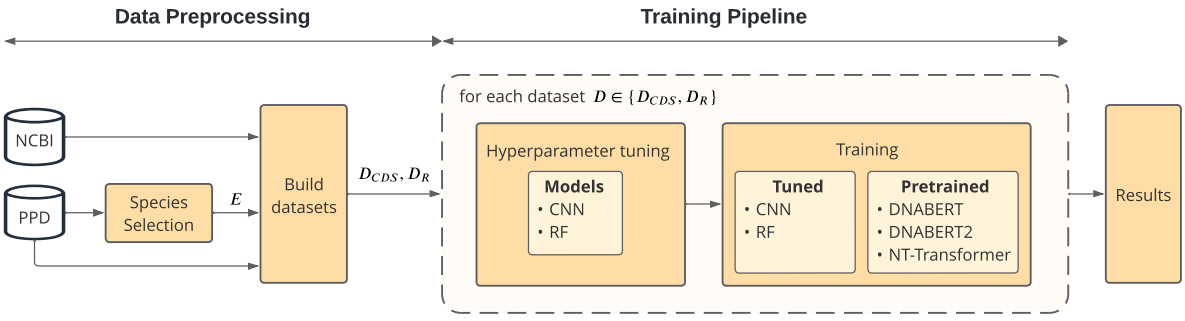


Figura 4.1: Flujo general de metodología aplicada.

En la fase de procesamiento de datos, se realiza la selección de especies utilizando los

promotores disponibles en PPD, basándose en criterios como la distribución de contenido de GC y la proximidad entre grupos taxonómicos. Posteriormente, a partir de las especies seleccionadas, se confeccionan dos conjuntos de datos, los cuales comparten los mismos datos de promotores pero difieren en el origen de las secuencias no promotoras: provenientes de secuencias codificantes (CDS) y generadas sintéticamente (SRS).

En la fase de entrenamiento, se realiza una optimización de hiperparámetros para los modelos basados en Redes Neuronales Convolucionales y Random Forest, seguida de una etapa de entrenamiento propiamente dicha, incluyendo modelos basados en BERT. Esta fase facilita el cómputo de métricas para cada modelo y conjunto de datos, lo que permite una evaluación comparativa del rendimiento de las dos metodologías de construcción de datos.

4.1. Etapa de preprocesamiento de datos

4.1.1. Selección de especies

En esta fase del estudio, se seleccionan especies que comparten características fenotípicas y genómicas similares, asegurando que posean suficientes datos para garantizar una amplia diversidad en el contenido de GC. Se realiza un análisis detallado de las especies y del número asociado de promotores, categorizados por filo. Este enfoque permite identificar un filo que no solo alberga el mayor número de especies, sino que también mantiene un volumen de datos sustancial.

4.1.2. Creación de datasets

El flujo de trabajo de preprocesamiento del conjunto de datos se muestra en la Figura 4.2. Los promotores se obtienen desde PPD, mientras que las cepas de secuencias codificantes para cada especie seleccionada E se obtienen del NCBI.

Los promotores se obtuvieron del PPD, incluyendo todas las secuencias de cada cepa de *Pseudomonadota* con más de 200 promotores. Para el conjunto de datos negativos de CDS, se extrajeron de manera aleatoria secuencias de 81 nucleótidos de las CDS de cada

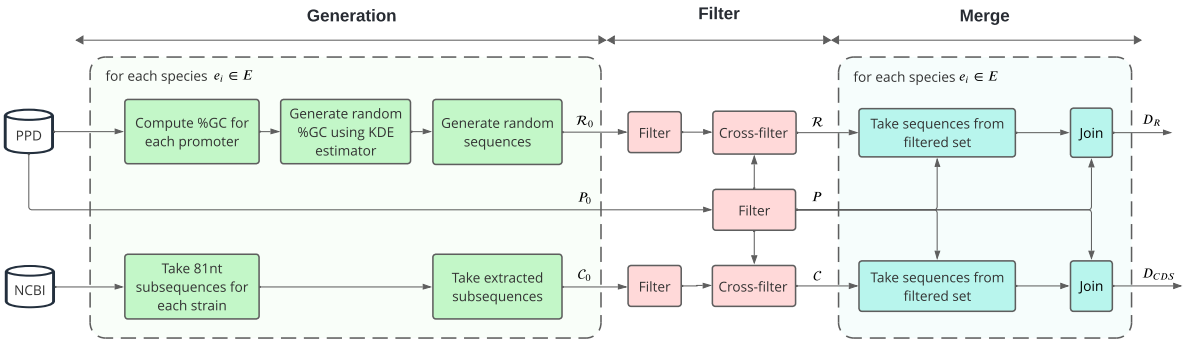


Figura 4.2: Flujo de trabajo de preprocesamiento de los conjuntos de datos a estudiar.

ensamblaje genómico disponible en NCBI asociado a las cepas incluidas en el conjunto de datos positivos [33]. Para el conjunto de datos negativos de SRS, se utilizó la técnica de estimación de densidad mediante la técnica de Kernel Density Estimator [48] para aproximar una función de distribución de probabilidad del contenido GC basada en los promotores disponibles en P_0 por especie incluida e_i. Posteriormente, utilizando la función estimada, se generan valores de contenido GC que a través de un algoritmo generan secuencias sintéticas, generando el conjunto de datos SRS. El algoritmo para generar secuencias con comportamiento aleatorio y con un contenido GC dado se muestra en el Algoritmo 1 del Apéndice A.

En ambos conjuntos de datos negativos, se extrajo el doble de secuencias por especie en comparación con el conjunto de datos positivo. Esto garantiza un número igual o mayor de secuencias en el conjunto de datos negativo después del filtrado, permitiendo un equilibrio entre secuencias de promotores y no promotores en las fases posteriores sin comprometer los datos de promotores. Tras esta fase, se obtuvieron los conjuntos de secuencias no promotoras sin procesar $\mathcal{C}_0$ y $\mathcal{R}_0$ para CDS y SRS, respectivamente. Inicialmente, se aplica una operación de filtrado dentro de cada conjunto de datos sin procesar ($\mathcal{P}_0$, $\mathcal{C}_0$ y $\mathcal{R}_0$) para evitar secuencias repetidas.

Posteriormente, se realiza una operación de filtrado cruzado entre los conjuntos de datos de promotores y no promotores. Este paso elimina secuencias no promotoras similares a los promotores P, lo que resulta en conjuntos filtrados de no promotores C y R. Todo el proceso utiliza el programa CD-HIT [49] con un umbral de 0.8 para garantizar una reducción efectiva

de la redundancia [16, 23].

Tras la etapa de filtrado, los datos de promotores y no promotores se integraron según sus respectivos orígenes. Para cada cepa de especie, las secuencias no promotoras se extrajeron de C y R. La cantidad de secuencias no promotoras extraídas coincide con el número de promotores filtrados disponibles tras la fase de filtrado, asegurando un conjunto de datos balanceado entre los conjuntos positivos y negativos. Posteriormente, las secuencias no promotoras se combinaron con las secuencias extraídas para formar los conjuntos de datos D_{CDS} y D_R.

4.2. Etapa de entrenamiento

4.2.1. Selección de modelos

Dado los modelos utilizados en trabajos anteriores, se decide incorporar al trabajo los modelos de Random Forest (RF), Redes Neuronales Convolucionales (CNN) y arquitecturas basadas en BERT para la predicción de promotores. En particular, se incluyen modelos basados en BERT como DNABERT [20], DNABERT-2 [21] y Nucleotide Transformers [22], ya que han mostrado un gran potencial para desempeñarse bien en otros contextos, aunque no han sido evaluados específicamente en el marco de esta investigación. Para los experimentos realizados en este estudio, se seleccionaron versiones específicas de DNABERT, DNABERT-2 y Nucleotide Transformers. En este caso, se utilizó DNABERT-3, que cuenta con aproximadamente 100 millones de parámetros y emplea tokenización basada en 3-mers; Nucleotide Transformer v2, una versión multi-especie con 500 millones de parámetros; y DNABERT-2, utilizando el único modelo disponible con 117 millones de parámetros.

La arquitectura de CNN utilizada en este estudio incluye tres capas convolucionales unidimensionales diseñadas para datos de secuencia lineal, cada una seguida de normalización por lotes *batch normalization* para estabilizar el entrenamiento y de agrupamiento máximo (max pooling) para reducir las dimensiones espaciales y mejorar la detección de características. Seguido de lo anterior, se incorpora *dropout* para prevenir el sobreajuste, y una capa

flatten convierte las salidas multidimensionales en un arreglo unidimensional. Finalmente, la red incluye tres capas densas con activación ReLU [50], seguidas por una capa densa de salida que utiliza una función de activación sigmoide, interpretable como una aproximación de la probabilidad de que una secuencia corresponda a un promotor.

4.2.2. Encoding de entrada

Para ambos modelos, CNN y RF, las secuencias genómicas en el conjunto de datos se codifican utilizando one-hot encoding. En este esquema, cada nucleótido se mapea distintivamente a un vector de cuatro dimensiones: $A \to (1,0,0,0)$, $C \to (0,1,0,0)$, $G \to (0,0,1,0)$, y $T \to (0,0,0,1)$. Para una secuencia de longitud N, esta codificación se traduce en un vector de dimensiones $(N,4)$ para la CNN. Este formato preserva la relación espacial entre nucleótidos, lo cual es esencial para que las capas convolucionales extraigan patrones de manera efectiva. Por el contrario, para el modelo RF, la codificación se adapta a un vector unidimensional de longitud $4N$ al aplanar la representación bidimensional. Por otro lado, los modelos basados en BERT utilizan un enfoque diferente. Estos modelos integran codificación posicional y embeddings dentro del mecanismo de attention de la arquitectura Transformer, permitiéndoles discernir el contexto y las dependencias a través de segmentos extendidos de la secuencia, reconociendo la importancia y posición de cada nucleótido dentro de los datos genómicos.

4.2.3. Ajuste de hiperparámetros

En el flujo de trabajo propuesto, se utilizaron arquitecturas RF, CNN y modelos basados en BERT adaptados para la lectura de secuencias de ADN, con ajuste de hiperparámetros realizado solo en los primeros modelos. Los modelos RF y CNN son relativamente más simples, y su rendimiento puede mejorar significativamente con un ajuste extenso de hiperparámetros. En contraste, los modelos basados en BERT, que son modelos de lenguaje grande preentrenados en conjuntos de datos extensos, tienden a mostrar un rendimiento robusto sin necesidad de ajustes extensivos de hiperparámetros. El conjunto de datos se

divide en dos subconjuntos: 80 % para entrenamiento y 20 % para validación. Durante la optimización de hiperparámetros, el subconjunto de entrenamiento (80 % de los datos) se subdivide utilizando validación cruzada en $k = 5$ etapas (k-folds), en donde en cada etapa, los datos se dividen en un 80 % para entrenamiento y un 20 % para validación.

4.2.3.1. Convolutional Neural Network

Los hiperparámetros se optimizaron utilizando el algoritmo Optuna [51] en 60 iteraciones, incluyendo la tasa de dropout, que se varió de 0.2 a 0.8 en incrementos de 0.1; el número de neuronas en cada capa convolucional, configurado en 128, 256 o 512; y los tamaños del kernel, seleccionados entre 3, 5 o 7. Además, se aplicaron regularizadores L2 a los pesos y sesgos de cada capa convolucional, con valores posibles de 0.001 o 0.0001. Para las capas densas, el número de neuronas se estableció en 128, 256 o 512, y la regularización L2 se ajustó de manera similar con valores de 0.001 o 0.0001. Se escogerá la combinación de hiperparámetros que maximice la media del F1-Score entre los folds.

4.2.3.2. Random Forest

Se utiliza el algoritmo Grid Search para optimizar los hiperparámetros del modelo, considerando el número de características por iteración (`sqrt` o `log2`) y el número de estimadores (1000, 2000, 3000, 4000 y 5000). El objetivo es maximizar la media del F1-Score entre los folds, siguiendo el mismo criterio aplicado en la optimización de la CNN.

4.2.4. Entrenamiento

En la etapa de entrenamiento, los modelos RF y CNN fueron entrenados utilizando los mejores hiperparámetros identificados durante el proceso de optimización. El conjunto de validación, reservado de la división inicial 80/20 del conjunto de datos, se utilizó para evaluar todos los modelos, incluidas las arquitecturas basadas en RF, CNN y BERT.

Para los modelos CNN, se utilizó el optimizador Adam [52] con una tasa de aprendizaje de 1×10^{-4}, mientras que los modelos basados en BERT emplearon el optimizador AdamW

[53] con un decaimiento de peso (*weight decay*) de 0.01. Se probó un rango de tasas de aprendizaje $[1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}, 4 \times 10^{-5}, 5 \times 10^{-5}]$, seleccionándose la tasa óptima basada en el mejor F1-Score obtenido en los datos de validación.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos a partir de la metodología propuesta, considerando tanto la construcción del conjunto de datos como los resultados que permiten analizar la influencia del contenido de GC en las predicciones de los modelos entrenados con los conjuntos de datos CDS y SRS. Este análisis se aborda desde una perspectiva tanto cuantitativa como cualitativa, complementado con la evaluación del desempeño de estos modelos en un entorno real.

Los experimentos fueron ejecutados en un equipo con sistema operativo Ubuntu 22.04 y con las siguientes especificaciones de hardware: procesador AMD Ryzen 7 5800X, 32 GB de memoria RAM y tarjeta gráfica NVIDIA RTX 4080 SUPER.

5.1. Selección de especies

En la Tabla 5.1 se presenta la cantidad de promotores disponibles en PPD, agrupados por especie y filo. Dada la alta diversidad dentro de las bacterias, se decidió enfocar este estudio en un único filo con el objetivo de mejorar el desempeño en la predicción de promotores. A diferencia de estudios previos, que no han considerado la información taxonómica para seleccionar las relaciones evolutivas entre especies, este enfoque incorpora dicho criterio para enriquecer el análisis.

El filo *Pseudomonadota* fue seleccionado debido a que presentó la mayor cantidad de

datos disponibles, con 80,448 secuencias promotoras, así como la mayor representación de especies (10), cada una perteneciente a un género diferente. Este nivel de diversidad en el conjunto de datos proporciona un marco más robusto para el análisis en comparación con otros filos.

Tabla 5.1: Número de promotores por filo y especie en PPD.

Filo	Especie	N° promotores
Pseudomonadota	*Acinetobacter baumannii* ATCC 17978	1540
	Agrobacterium tumefaciens str C58	706
	Bradyrhizobium japonicum USDA 110	15933
	Burkholderia cenocepacia J2315	10831
	Escherichia coli str K-12 substr. MG1655	8616
	Klebsiella aerogenes KCTC 2190	763
	Pseudomonas putida strain KT2440	7938
	Shigella flexneri 5a str. M90T	14051
	Sinorhizobium meliloti 1021	17003
	Xanthomonas campestris pv. *campestris* B100	3067
	Total	**80448**
Cyanobacteria	*Nostoc* sp. PCC7120	13705
	Synechococcus elongatus PCC 7942	1473
	Synechocystis sp. PCC 6803	944
	Total	**16122**
Campylobacterota	*Campylobacter jejuni* RM1221	2166
	Campylobacter jejuni subsp. *jejuni* 81-176	2142
	Campylobacter jejuni subsp. *jejuni* 81116	1942
	Campylobacter jejuni subsp. *jejuni* NCTC 11168	1905
	Helicobacter pylori strain 26695	2228
	Total	**10370**
Bacillota	*Bacillus subtilis* subsp. *subtilis* str. 168	691
	Paenibacillus riograndensis SBR5	2351
	Staphylococcus aureus subsp. *aureus* MW2	2821
	Staphylococcus epidermidis ATCC 12228	2207
	Streptococcus pyogenes strain S119	892
	Total	**8962**
Euryarchaeota	*Haloferax volcanii* DS2	4749
	Thermococcus kodakarensis KOD1	2720
	Total	**7469**
Actinomycetota	*Corynebacterium diphtheriae* NCTC 13129	1656
	Corynebacterium glutamicum ATCC 13032	3581
	Total	**5237**
Mycoplasmatota	*Onion yellows phytoplasma OY-M*	231
	Total	**231**

5.2. Preprocesamiento de datos

5.2.1. Estructura de los datos

Para construir los conjuntos de datos necesarios para el entrenamiento, se integraron dos fuentes principales de datos: el *Prokaryotic Promoter Dataset (PPD)*, que proporciona información detallada sobre promotores en diferentes especies, y el dataset resultante después del proceso de filtrado, utilizado para clasificar las secuencias como promotores o no promotores. Los campos que conforman ambas fuentes se muestran en la Tabla 5.2 y 5.3, respectivamente. Ambas fuentes se encuentran en formato `.csv`.

Estos conjuntos de datos se integran mediante una unión que combina la información genómica con las etiquetas de clasificación, generando un conjunto de datos consistente y adecuado para el entrenamiento de modelos. En específico, la relación entre los datasets se establece utilizando conjuntamente los campos `id` y `SpeciesName`. Para los casos en que `label` es 1 (promotor), el valor de `id` y `SpeciesName` en el segundo dataset coincide con los correspondientes en el PPD, garantizando que las secuencias etiquetadas como promotores están respaldadas por información genómica real a partir de PPD. En cambio, cuando `label` es 0 (no promotor), el campo `id` se genera como un identificador aleatorio, pero se conserva el valor de `SpeciesName` para mantener la coherencia con el contexto genómico de la especie asociada.

Tabla 5.2: Descripción de los campos de PPD.

Campo	**Descripción**
`id`	Identificador único del registro.
`Type`	Tipo de promotor o secuencia.
`PromoterName`	Nombre asignado al promotor.
`SpeciesName`	Nombre de la especie a la que pertenece el promotor.
`GeneName`	Nombre del gen asociado al promotor.
`GeneStart`	Posición inicial del gen en la secuencia genómica.
`GeneEnd`	Posición final del gen en la secuencia genómica.
`GeneProduct`	Producto codificado por el gen asociado al promotor.
`TSSPosition`	Posición del sitio de inicio de la transcripción (TSS).
`Strand`	Hebra de ADN (directa o complementaria) en la que se encuentra el promotor.
`Locus_tag`	Etiqueta única del locus asociada al gen.
`Sequence`	Secuencia de ADN correspondiente al promotor.

Tabla 5.3: Descripción de los campos del dataset filtrado (CDS/SRS).

Campo	Descripción
`id`	Identificador único del registro.
`SpeciesName`	Nombre de la especie a la que pertenece la secuencia.
`Sequence`	Secuencia de ADN correspondiente al registro.
`label`	Etiqueta de clasificación: 1 para promotores, 0 para no promotores.

5.2.2. Filtrado de redundancia

En la Tabla 5.4 se muestra el número de secuencias antes y después del filtrado de secuencias redundantes para promotores y no promotores, considerando el origen de estas últimas.

Tabla 5.4: Cantidad de promotores y no promotores obtenidos durante la extracción y antes de la fase de filtrado de redundancia para cada origen de secuencias no promotoras.

	Promotores		No promotores		
Especie	Original	Filtrado	Original (SRS/CDS)	Filtrado (SRS)	Filtrado (CDS)
A. baumannii	1540	1111	3080	3018	2993
A. tumefaciens	706	698	1412	1400	1389
B. cenocepacia	10831	9655	21662	21379	19259
B. japonicum	15933	14760	31866	31691	27445
E. coli	8616	4920	17232	17110	11338
K. aerogenes	763	457	1526	1517	1425
P. putida	7938	6847	15876	15702	14040
S. flexneri	14051	9755	28102	27974	20376
S. meliloti	17003	14481	34006	33944	27211
X. campestris	3067	2963	6134	6054	5766
Total	80448	65647	160896	159789	131242

El proceso de filtrado eliminó más secuencias entre los promotores y las secuencias extraídas de secuencias codificantes. Para las secuencias no promotoras generadas sintéticamente, el proceso de filtrado no eliminó una cantidad significativa. Como resultado, hay 65,647 promotores disponibles para la construcción del conjunto de datos. En cuanto a las secuencias no promotoras, se muestra que en todas las especies, el número de estas

secuencias supera al de los promotores, lo que permite la creación de conjuntos de datos equilibrados de secuencias promotoras y no promotoras para cada especie.

5.2.3. Distribución de contenido GC en datasets

Una vez obtenidos los conjuntos de datos, se analizó la distribución del contenido de GC para promotores y no promotores según su origen, como se muestra en la Figura 5.1. En el caso del conjunto de datos SRS, se observa que las secuencias promotoras y no promotoras comparten una distribución de contenido de GC similar en todas las especies. Por el contrario, en el conjunto de datos CDS, la similitud en la distribución del contenido de GC entre los tipos de secuencias varía según la especie. Por ejemplo, en las especies *K. aerogenes* y *A. tumefaciens*, las distribuciones difieren significativamente, mientras que en *P. putida* y *X. campestris* muestran una mayor similitud.

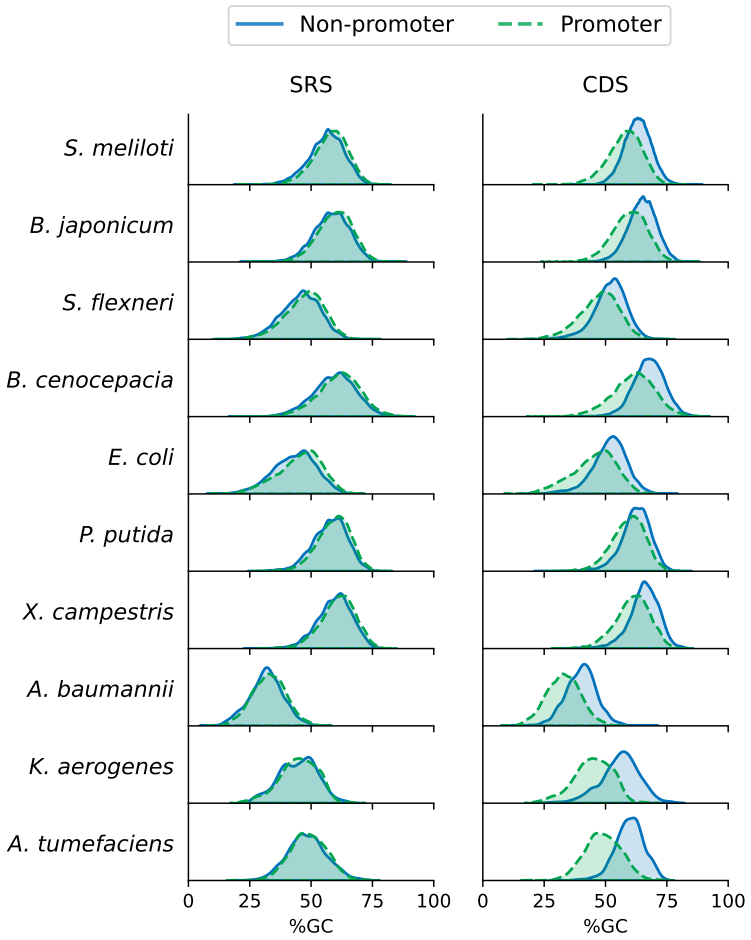


Figura 5.1: Distribución de contenido GC de promotores y no promotores, por cada conjunto de datos y especie.

5.3. Evaluación Global

La Tabla 5.5 presenta una comparación del rendimiento de los modelos para cada conjunto de datos estudiado, evaluado en términos de especificidad, sensibilidad, recall, accuracy, MCC, F1 Score y el AUC de la curva ROC en general. Se observa que DNABERT y Nucleotide Transformer exhiben un rendimiento superior en ambos conjuntos de datos, CDS y SRS, demostrando una mayor capacidad de aprendizaje en ambos escenarios de extracción de secuencias negativas. Dentro de los modelos basados en BERT, se destaca que DNABERT2 tiene el peor desempeño entre sus pares en ambos conjuntos de datos. La diferencia es más notable en el conjunto de datos CDS, donde DNABERT2 es superado por DNABERT por

un margen sustancial de 8 puntos en términos de sensibilidad, lo que resalta una disparidad significativa en la efectividad entre estos dos modelos.

Para los modelos entrenados con el conjunto de datos CDS, DNABERT obtiene los mejores resultados en la mayoría de las métricas en comparación con otros modelos, excepto en recall y especificidad, donde DNABERT2 tiene un mejor desempeño por dos puntos. Para ambos modelos, las diferencias en los valores de las métricas son más pronunciadas en cuanto a especificidad, sensibilidad y recall, oscilando entre 2 y 3 puntos, mientras que las diferencias en otras métricas son marginales. Es importante señalar que todos los modelos exhiben valores de especificidad relativamente altos en comparación con los valores de sensibilidad, lo que indica una brecha que favorece la clasificación precisa de los no-promotores sobre los promotores. Esta tendencia es consistente en todos los modelos, lo que sugiere un desafío general en lograr un rendimiento de clasificación equilibrado para las secuencias promotoras. Las brechas observadas en especificidad y sensibilidad destacan la necesidad de un mayor refinamiento de los modelos para mejorar la detección de secuencias promotoras sin comprometer la clasificación de secuencias no-promotoras. Además, este desequilibrio podría indicar que el problema está más relacionado con los datos que con los modelos, señalando mejoras potenciales en el conjunto de datos como un paso crucial para abordar este problema.

Para los modelos entrenados con el conjunto de datos SRS, los valores en todas las métricas aumentan significativamente entre 15 y 20 puntos en comparación con los resultados del conjunto de datos CDS, con DNABERT y Nucleotide Transformer superando a todos los demás modelos. La diferencia entre los dos modelos en cuanto a especificidad, sensibilidad y recall se mantiene consistente en 1 a 2 puntos, lo que resalta su rendimiento superior en diferentes conjuntos de datos. En cuanto a sensibilidad y especificidad, los modelos del conjunto de datos SRS exhiben un rendimiento más equilibrado, con la brecha entre estas métricas menos pronunciada que en los modelos del conjunto de datos CDS. Esto sugiere que los modelos del conjunto de datos SRS son más aptos para identificar con precisión secuencias promotoras y no promotoras. Sin embargo, es importante tener en cuenta que los no promotores en el conjunto de datos SRS son secuencias generadas aleatoriamente,

por lo que la aplicabilidad y fiabilidad de estos modelos en un contexto del mundo real, como el análisis de un genoma completo, pueden ser limitadas y deben interpretarse con precaución.

Tabla 5.5: Métricas generales para ambos conjuntos de datos

Dataset	Modelo	Sp	Sn	Pre	Acc	MCC	F_1	ROC AUC
CDS	RF	0.7761	0.6440	0.7448	0.7096	0.4237	0.6907	0.7824
	CNN	0.8574	0.7307	0.8387	0.7936	0.5925	0.7810	0.8678
	DNABERT	0.8665	**0.7660**	0.8534	**0.8159**	**0.6355**	**0.8074**	**0.8894**
	DNABERT2	0.8763	0.6851	0.849	0.78	0.5715	0.7583	0.8517
	NT-Transformer	**0.8885**	0.7402	**0.8707**	0.8138	0.6353	0.8002	0.89
SRS	RF	0.8099	0.9520	0.8356	0.8815	0.7704	0.89	0.9554
	CNN	0.9374	0.9325	0.9379	0.9349	0.8698	0.9352	0.9853
	DNABERT	0.9264	0.9666	0.9302	**0.9466**	**0.894**	**0.948**	**0.9885**
	DNABERT2	0.9125	**0.9691**	0.9182	0.941	0.8833	0.943	0.9867
	NT-Transformer	**0.9439**	0.9462	**0.9448**	0.945	0.8901	0.9455	0.9872

5.4. Análisis de especificidad y sensibilidad entre especies

Independientemente del modelo, en el dataset CDS se observa una brecha significativa entre la especificidad y la sensibilidad, siendo la sensibilidad la más afectada negativamente, lo que sugiere que los modelos tienen una mayor capacidad para detectar secuencias no promotoras que secuencias promotoras. Además, es necesario identificar qué especies impactan más significativamente en estas métricas. En este sentido, la Tabla 5.6 presenta las métricas para cada especie utilizando el conjunto de datos CDS y DNABERT, basado en el mejor modelo en cuatro de siete métricas a partir de la Tabla 5.5.

Tabla 5.6: Métricas para el dataset CDS por especie usando DNABERT.

Especie	Sp	Sn	Pre	Acc	MCC	F_1	ROC AUC
A. baumannii	0.7512	0.9685	0.8083	0.8642	0.7419	0.8811	0.9777
A. tumefaciens	0.8203	0.88	0.8516	0.8525	0.7028	0.8656	0.9126
B. cenocepacia	0.9144	0.8213	0.9068	0.8675	0.7385	0.8619	0.9355
B. japonicum	0.8606	0.7736	0.8465	0.8172	0.6367	0.8084	0.8954
E. coli	0.8584	0.6497	0.817	0.7555	0.5203	0.7238	0.8322
K. aerogenes	0.8043	0.9608	0.8448	0.8866	0.7792	0.8991	0.981
P. putida	0.871	0.6513	0.8361	0.7606	0.5351	0.7323	0.8239
S. flexneri	0.8473	0.8749	0.8548	0.8613	0.7226	0.8647	0.9281
S. meliloti	0.8553	0.6833	0.8297	0.768	0.5459	0.7494	0.8386
X. campestris	0.9240	0.9134	0.9232	0.9187	0.8375	0.9183	0.9707

Se observa que algunas especies exhiben una mayor especificidad que sensibilidad, mientras que otras muestran la tendencia opuesta. Por ejemplo, especies como *P. putida* y *E. coli* demuestran una especificidad más pronunciada, con brechas que oscilan entre aproximadamente 0.19 y 0.22. Por el contrario, especies como *A. baumannii* y *K. aerogenes* muestran una mayor sensibilidad que especificidad, con brechas observadas de hasta 0.22. Esta variabilidad en el rendimiento del modelo entre especies subraya los desafíos para lograr una sensibilidad y especificidad equilibradas en la clasificación. El rango de estas brechas, desde diferencias sutiles de aproximadamente 0.01 hasta disparidades más significativas de alrededor de 0.22, enfatiza aún más la diversidad en las capacidades de detección en diferentes

contextos bacterianos. Además, especies como *E. coli*, *S. meliloti* y *P. putida* disminuyen notablemente la sensibilidad, cada una mostrando valores por debajo de 0.69, mientras que *B. japonicum* exhibe una sensibilidad de 0.77, aunque en menor medida. Por otro lado, la especificidad se ve más comprometida por *A. baumannii*, con una especificidad de 0.75, y, en menor medida, por *K. aerogenes*, que registra una especificidad de 0.80.

5.5. Sensibilidad en dataset CDS relacionado a sesgo de contenido GC

Se han identificado problemas en diversas tareas biotecnológicas asociadas con el sesgo de contenido de GC, incluyendo la detección de promotores [12]. Para corroborar estas observaciones, se realizará una proyección del espacio latente de los tokens de salida `[CLS]` de DNABERT utilizando UMAP (Uniform Manifold Approximation and Projection) [54]. UMAP reducirá la dimensionalidad de los datos, facilitando la visualización y el análisis de conjuntos de datos complejos y de alta dimensionalidad como parte del análisis del conjunto de datos de validación. Esta técnica de reducción de dimensionalidad ayuda a discernir patrones y estructuras dentro de los datos que podrían no ser evidentes en dimensiones más altas.

La Figura 5.2 integra varios análisis del contenido de GC y su impacto en la clasificación de secuencias promotoras utilizando DNABERT para *Bradyrhizobium japonicum USDA 110* usando el dataset CDS. La Subfigura A muestra los boxplots que presentan la distribución del contenido de GC en verdaderos positivos, verdaderos negativos, falsos positivos, falsos negativos, y secuencias positivas y negativas (groundtruth). La Subfigura B presenta una proyección UMAP de los tokens de salida `[CLS]` de DNABERT, codificados por colores para representar los resultados de la clasificación del modelo. Las Subfiguras C y D se centran en las proyecciones de las predicciones del modelo para no promotores y promotores, respectivamente, en el conjunto de datos de validación. Para representar el contenido medio de GC, se emplean gráficos de densidad hexagonal (hexbin), los cuales permiten visualizar

la concentración de puntos en áreas específicas, destacando regiones donde se observa una mayor superposición de datos. Esto facilita identificar patrones en las predicciones del modelo y su relación con el contenido de GC. Estas visualizaciones evalúan colectivamente cómo el contenido de GC afecta el rendimiento del modelo y la agrupación espacial de secuencias, proporcionando ideas sobre los fundamentos genómicos que influyen en la precisión del modelo al diferenciar promotores de no promotores basados en esta característica genómica.

El análisis de las Subfiguras C y D demuestra que la tarea de clasificación es más desafiante para las secuencias promotoras que para las secuencias no promotoras utilizando el modelo DNABERT, con claras variaciones en el contenido de GC entre los grupos. El grupo de no promotores (TN + FP), que oscila predominantemente entre el 80 % y el 60 %, está más claramente definido en comparación con el grupo de promotores (TP + FN), que muestra una mayor dispersión y abarca desde el 60 % hasta el 40 %. Este solapamiento es visible en la Subfigura B, donde muchos FN se superponen con TN, lo que destaca los desafíos en la diferenciación de categorías. Cabe destacar que muchas proyecciones clasificadas como promotores, especialmente aquellas con niveles más altos de contenido de GC, se encuentran dentro de la región de no promotores con bajo contenido de GC. Por el contrario, hay menos proyecciones de no promotores dentro del grupo de promotores, lo que indica una asimetría y dispersión significativas en la clasificación que sugieren la dificultad para distinguir secuencias promotoras. La prominencia de niveles más altos de GC entre las proyecciones de promotores mal clasificadas dentro del grupo de no promotores evidencia aún más un sesgo, impactando significativamente la precisión de la clasificación. Además, la Subfigura A muestra la distribución del contenido de GC en las clasificaciones, con una notable superposición entre falsos negativos y verdaderos negativos. Este patrón de distribución y la separación distinta entre verdaderos positivos y falsos positivos subrayan el impacto del contenido de GC en el rendimiento del modelo.

En la Subfigura A, los diagramas de caja ilustran una superposición significativa en la distribución del contenido de GC entre los TN y FN, con ambas categorías mostrando una considerable similitud en sus rangos intercuartiles. Los verdaderos negativos tienen un rango de contenido de GC que va aproximadamente del 62 % al 69 %, centrado en una mediana del

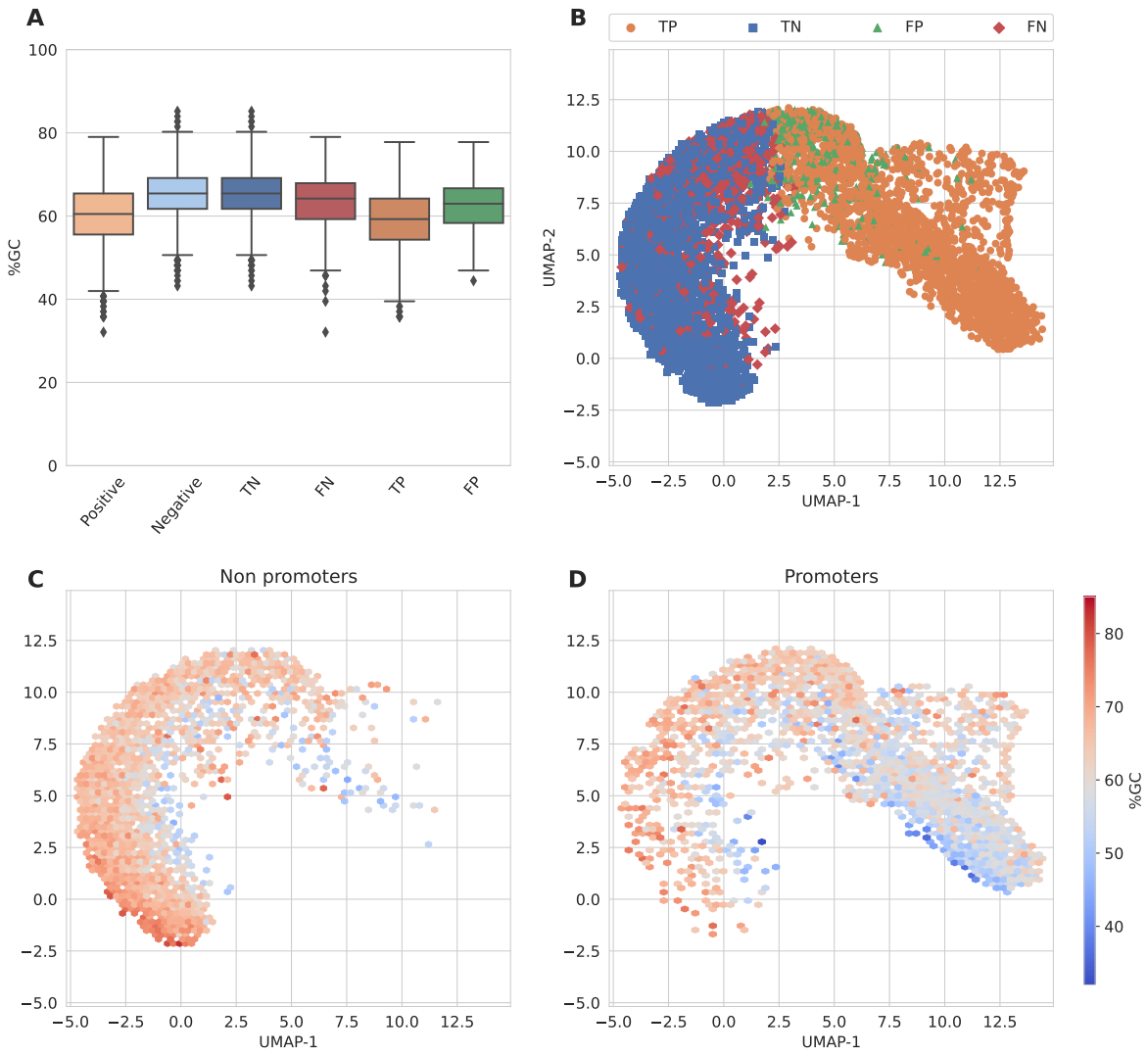


Figura 5.2: **Análisis comparativo de las clasificaciones de *B. japonicum* en el modelo basado en DNABERT (CDS)**. **A**: Diagramas de caja (boxplot) que muestran la distribución del contenido de GC para los conjuntos de datos positivos y negativos, junto con las categorías predichas (TP, TN, FP, FN). **B**: Proyecciones UMAP del token [CLS] de DNABERT, basadas en los datos de validación. **C,D**: Proyecciones UMAP de los valores reales para no promotores (TN+FP) y promotores (TP+FN), respectivamente, complementadas con mapas de densidad que ilustran el contenido GC.

65 %. Los falsos negativos muestran un rango intercuartílico ligeramente más estrecho, desde aproximadamente el 59 % hasta el 68 %, con una mediana del 64 %. Esta proximidad en la distribución sugiere que el modelo basado en DNABERT a menudo clasifica erróneamente las secuencias promotoras genuinas como no promotoras cuando su contenido de GC se

encuentra dentro de estos rangos superpuestos, lo que indica un sesgo en la capacidad del modelo para diferenciar en función del contenido GC. Es importante notar que los FN tienden a superponerse con los TN en otras especies, como se muestra en la Figura **??**. Las distribuciones de TP y FP presentan menos superposición y poseen un rango distintivo, lo que resalta diferencias en la sensibilidad del modelo hacia las características intrínsecas de los promotores.

5.6. Efectividad de generación sintética de secuencias para reducir sesgo de contenido GC

Como se discutió anteriormente, se ha observado un sesgo relacionado con el contenido de GC en tareas de clasificación utilizando secuencias codificantes como orígenes de secuencias no promotoras para modelos de una especie. En este sentido, se comprobará si la creación de secuencias no promotoras sintéticas, que forman una distribución de contenido de GC similar a la de los promotores por especie, mitiga dicho sesgo. Esta metodología tiene como objetivo alinear las características del contenido de GC entre categorías, mejorando así el rendimiento de los modelos. La Tabla 5.7 presenta las métricas de evaluación para DNABERT entrenado con el conjunto de datos SRS por cada especie.

Tabla 5.7: Métricas para el dataset SRS por especie usando DNABERT.

Especie	Sp	Sn	Pre	Acc	MCC	F_1	ROC AUC
A. baumannii	0.9268	0.9459	0.9333	0.9368	0.8734	0.9396	0.9779
A. tumefaciens	0.9531	0.9533	0.9597	0.9532	0.906	0.9565	0.9855
B. cenocepacia	0.9295	0.9789	0.9337	0.9544	0.9098	0.9558	0.9927
B. japonicum	0.9232	0.9661	0.9259	0.9446	0.89	0.9456	0.988
E. coli	0.9141	0.9605	0.9158	0.937	0.875	0.9376	0.9858
K. aerogenes	0.9565	0.9706	0.9612	0.9639	0.9277	0.9659	0.9907
P. putida	0.9153	0.938	0.9179	0.9267	0.8536	0.9278	0.9823
S. flexneri	0.9288	0.9526	0.9322	0.9409	0.8819	0.9423	0.9845
S. meliloti	0.9312	0.9824	0.9364	0.9572	0.9154	0.9588	0.9931
X. campestris	0.9382	0.9841	0.9409	0.9611	0.9232	0.962	0.9921

Se observa una mejora notable en comparación con el mismo modelo entrenado en el

conjunto de datos CDS; la especificidad y la sensibilidad son más cercanas, con una diferencia de no más de 0.05 puntos. Además, la sensibilidad generalmente supera a la especificidad, lo que indica una mejor identificación de promotores. Esto sugiere que las secuencias no promotoras sintéticas, que imitan la distribución del contenido de GC de los promotores, efectivamente conducen a una detección más precisa de promotores. De manera similar al análisis realizado con el conjunto de datos CDS, la Figura 5.3 muestra análisis del contenido de GC y su impacto en la clasificación de secuencias promotoras utilizando DNABERT para *Bradyrhizobium japonicum USDA 110*, utilizando un conjunto de datos SRS. Al igual que en el análisis de CDS, este enfoque implica el uso de UMAP para proyectar los tokens de salida `[CLS]` de DNABERT, permitiendo una visualización detallada de cómo las secuencias no promotoras sintéticas influyen en la capacidad del modelo para diferenciar entre secuencias promotoras y no promotoras en función de sus características de contenido GC.

Para el conjunto de datos SRS, la Subfigura A ilustra la distribución del contenido de GC en las diferentes clasificaciones, mostrando menos variabilidad en los rangos de contenido de GC que en el conjunto de datos CDS. Los verdaderos negativos (TN) y los falsos negativos muestran rangos intercuartiles superpuestos que van aproximadamente del 53 % al 62 %, lo que sugiere una brecha de distribución más estrecha. De manera similar, los verdaderos positivos y los falsos positivos tienen rangos intercuartiles que van del 56 % al 63 %, indicando distribuciones superpuestas sin un sesgo claro en el contenido de GC que diferencie estas clasificaciones. Este patrón sugiere un manejo más uniforme del contenido de GC entre categorías en comparación con el dataset CDS.

Las Subfiguras B, C y D muestran proyecciones UMAP para el dataset SRS, donde las subfiguras C y D representan los grupos de no promotores y promotores, respectivamente. Si bien algunos falsos negativos y falsos positivos se superponen dentro de sus respectivos grupos, no hay un patrón discernible de contenido de GC que distinga significativamente los grupos. Esta ausencia de un patrón claro de contenido de GC entre los grupos de promotores y no promotores es interpretable como una reducción del sesgo relacionado con el contenido de GC que se había observado previamente en el conjunto de datos CDS. El uso de secuencias que simulan de cerca las distribuciones de contenido GC de los promotores

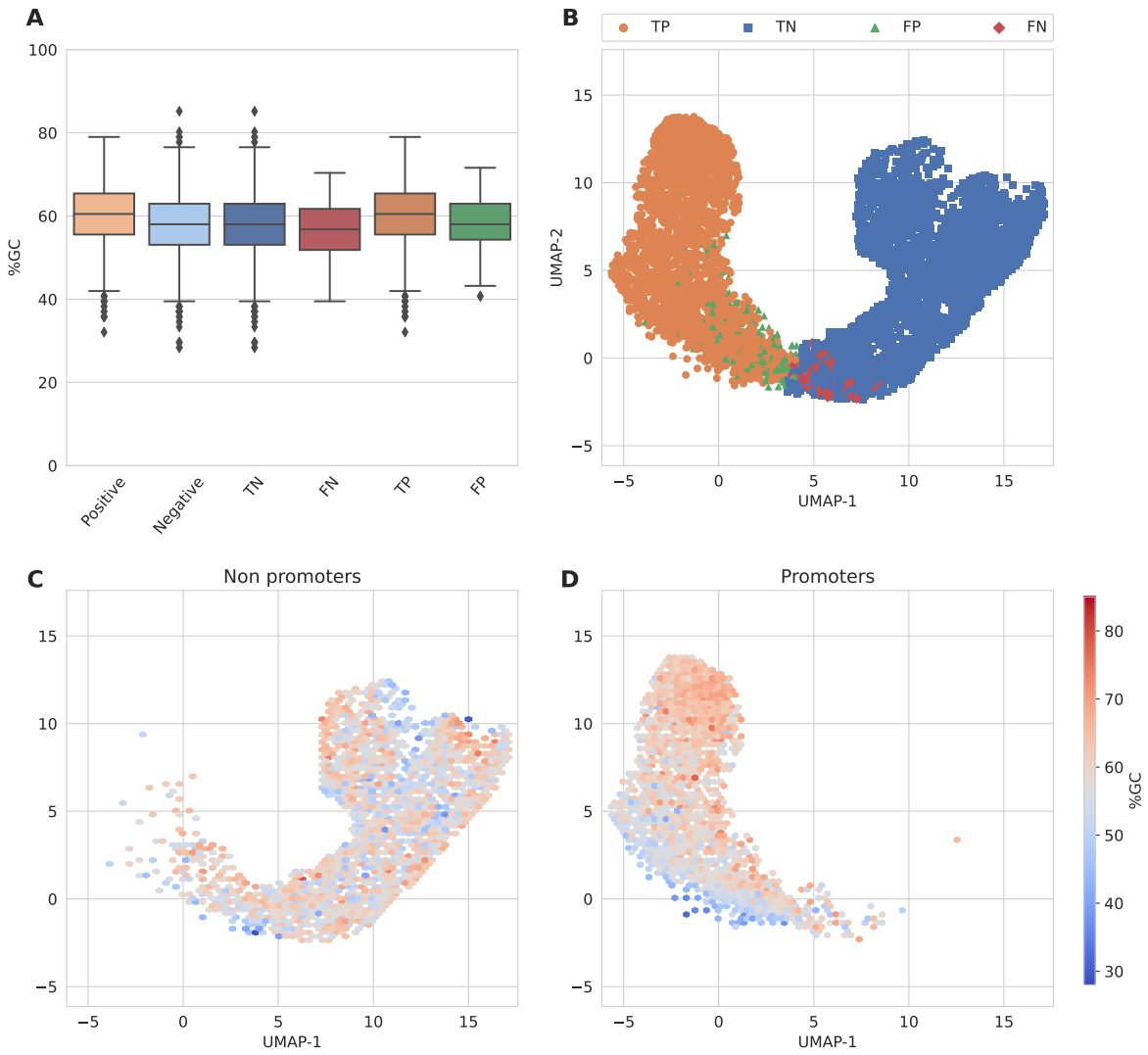


Figura 5.3: **Análisis comparativo de las clasificaciones de *B. japonicum* en el modelo basado en DNABERT (SRS)**. **A**: Diagramas de caja (boxplot) que muestran la distribución del contenido de GC para los conjuntos de datos positivos y negativos, junto con las categorías predichas (TP, TN, FP, FN). **B**: Proyecciones UMAP del token [CLS] de DNABERT, basadas en los datos de validación. **C,D**: Proyecciones UMAP de los valores reales para no promotores (TN+FP) y promotores (TP+FN), respectivamente, complementadas con mapas de densidad que ilustran el contenido GC.

contribuye a obtener resultados de clasificación más equilibrados y precisos por parte del modelo DNABERT. Estos hallazgos son consistentes en otras especies, como se muestran en la Figura 5.5

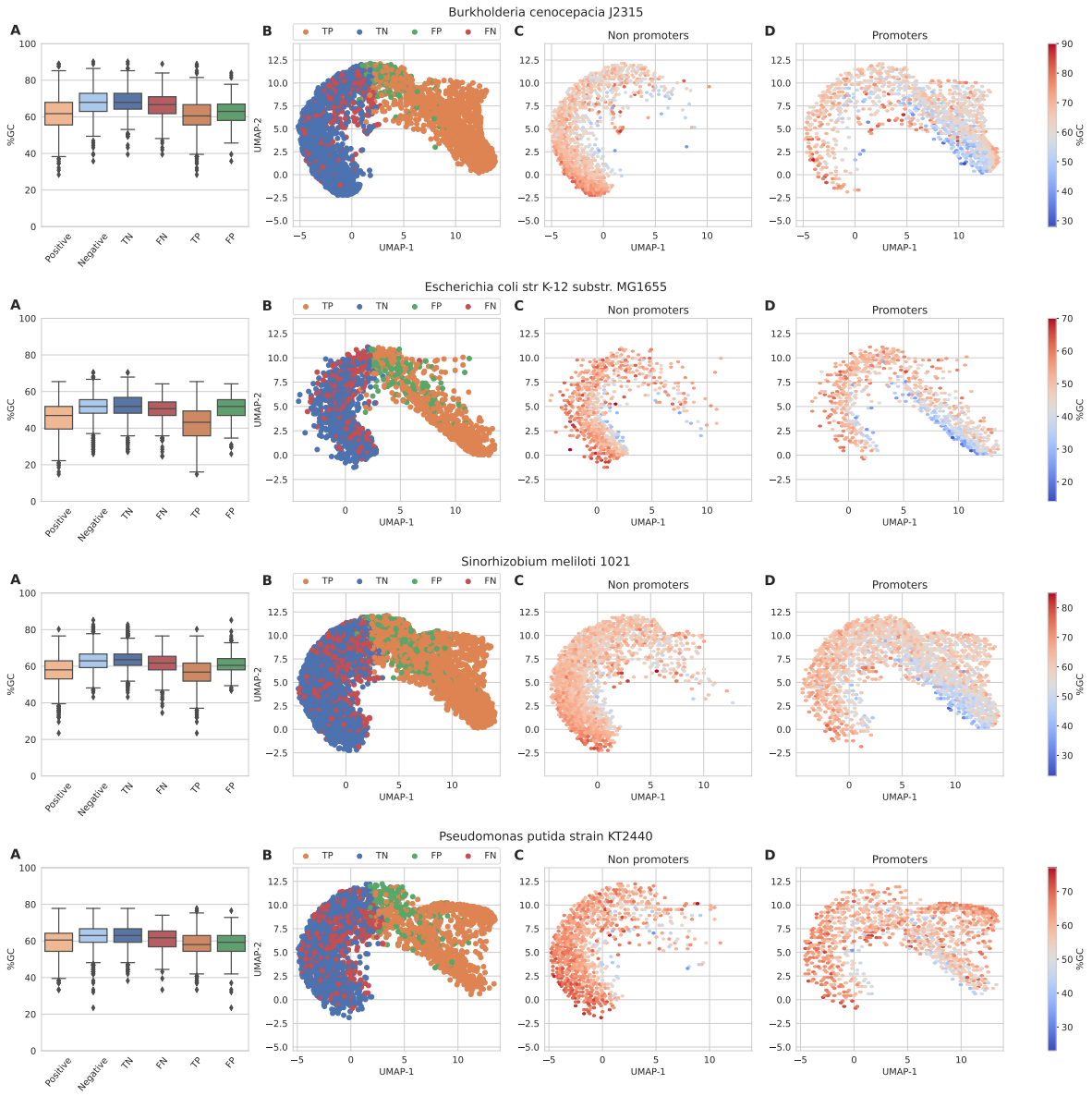


Figura 5.4: **Análisis Comparativo de Clasificaciones de Secuencias (CDS)**: (A) Diagramas de caja (boxplots) del contenido de GC para categorías reales y predichas; (B, D) Proyecciones UMAP de características de DNABERT con mapas de densidad que muestran variaciones en el contenido GC.

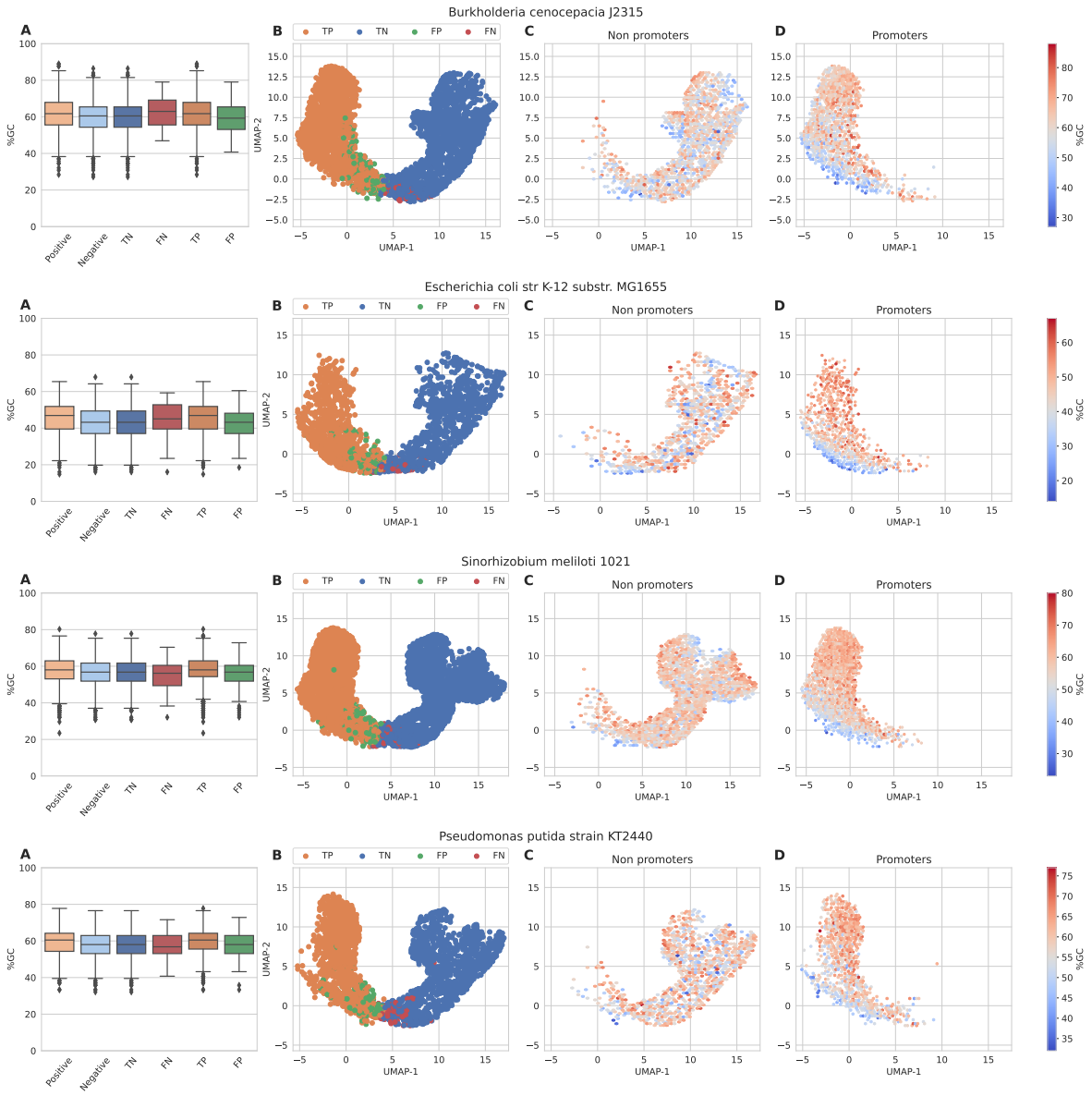


Figura 5.5: **Análisis Comparativo de Clasificaciones de Secuencias (SRS)**: (A) Diagramas de caja (boxplots) del contenido de GC para categorías reales y predichas; (B, D) Proyecciones UMAP de características de DNABERT con mapas de densidad que muestran variaciones en el contenido GC.

5.7. Evaluación de modelos en genoma

Aunque el conjunto de datos SRS contribuye a una clasificación más precisa y equilibrada en términos de especificidad y sensibilidad, estos resultados deben interpretarse con cautela debido a la posibilidad de sobreajuste del modelo. Para investigar este aspecto, se evaluó el comportamiento de los modelos DNABERT entrenados con CDS y SRS en datos genómicos reales de la especie con las mejores métricas, *X. campestris*.

Se analizaron diez promotores de *X. campestris* dentro de un intervalo de 5000 nucleótidos a la izquierda (extremo 5') y a la derecha (extremo 3') del sitio de inicio de transcripción (TSS), utilizando una ventana deslizante con un paso de 1. Esto permitió predecir regiones promotoras y no promotoras en función de la probabilidad promedio de softmax obtenida a partir del modelo, como se muestran en las Figuras 5.6, 5.7 y 5.8.

Ambos modelos mostraron un desempeño consistente en la identificación de promotores en datos genómicos reales. Sin embargo, el modelo basado en CDS presentó una mayor detección de ruido de fondo como señales promotoras, lo que afectó la precisión en la predicción de promotores.

En contraste, el modelo entrenado con SRS mostró un comportamiento distintivo: las predicciones de probabilidad permanecieron consistentemente altas en regiones generales del genoma, pero presentaron alteraciones en áreas que contenían promotores (indicadas en las áreas resaltadas). Este patrón sugiere la necesidad de reinterpretar los resultados del modelo entrenado con dicho dataset, enfocándose en la detección de bajadas en las probabilidades, en lugar de depender únicamente de un umbral fijo, como valores de softmax superiores a 0.5.

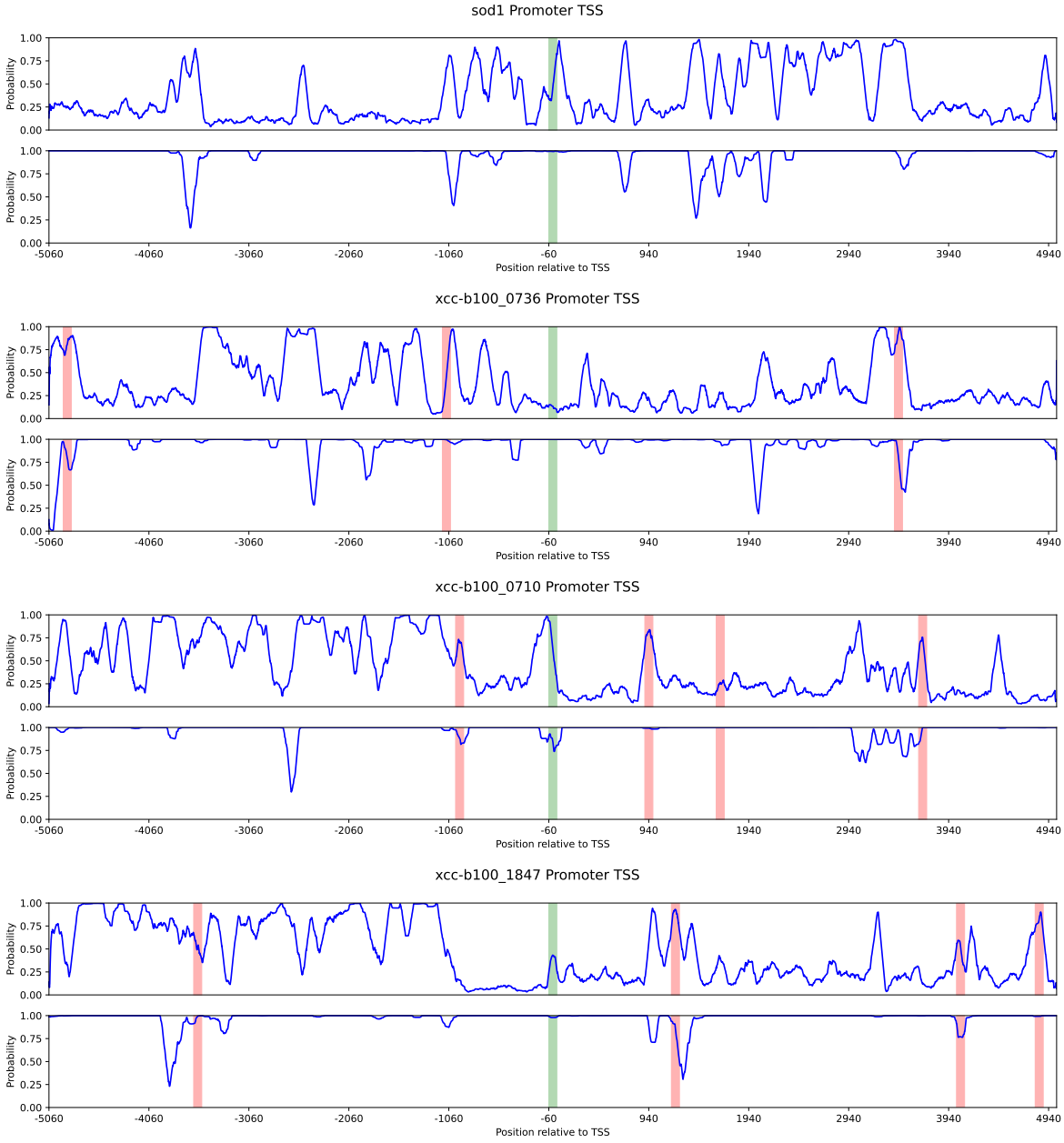


Figura 5.6: Probabilidades promedio por nucleótido predichas por DNABERT para un subconjunto de 10 promotores asociados con genes en *X. campestris*, utilizando modelos entrenados en los conjuntos de datos CDS (Superior) y SRS (Inferior). Cada gráfico representa un promotor vinculado a un gen específico. Las marcas en verde indican promotores del conjunto de datos de validación, mientras que las marcas en rojo señalan promotores adicionales incluidos en PPD.

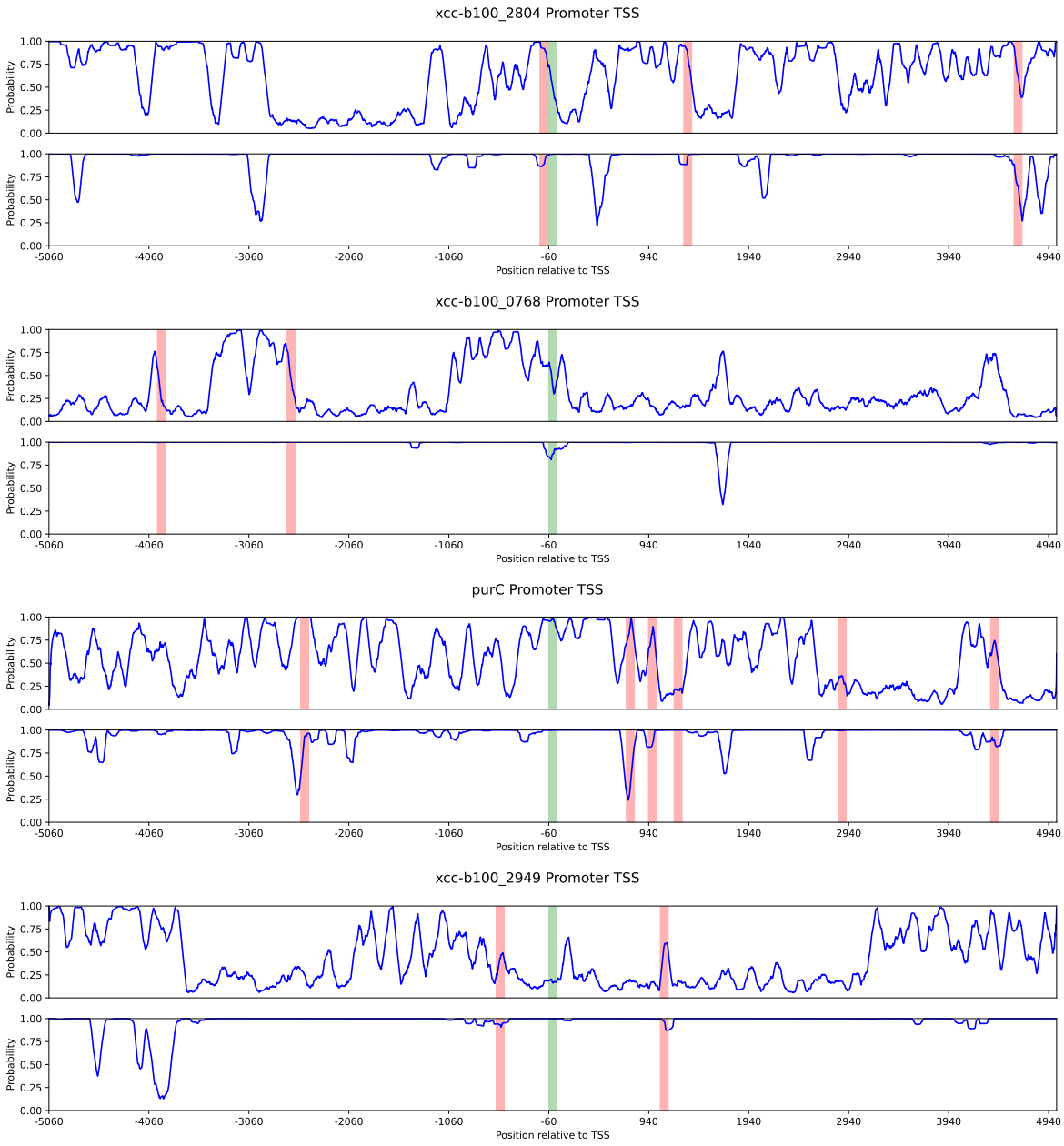


Figura 5.7: Probabilidades promedio por nucleótido predichas por DNABERT para un subconjunto de 10 promotores asociados con genes en *X. campestris*, utilizando modelos entrenados en los conjuntos de datos CDS (Superior) y SRS (Inferior). Cada gráfico representa un promotor vinculado a un gen específico. Las marcas en verde indican promotores del conjunto de datos de validación, mientras que las marcas en rojo señalan promotores adicionales incluidos en PPD.

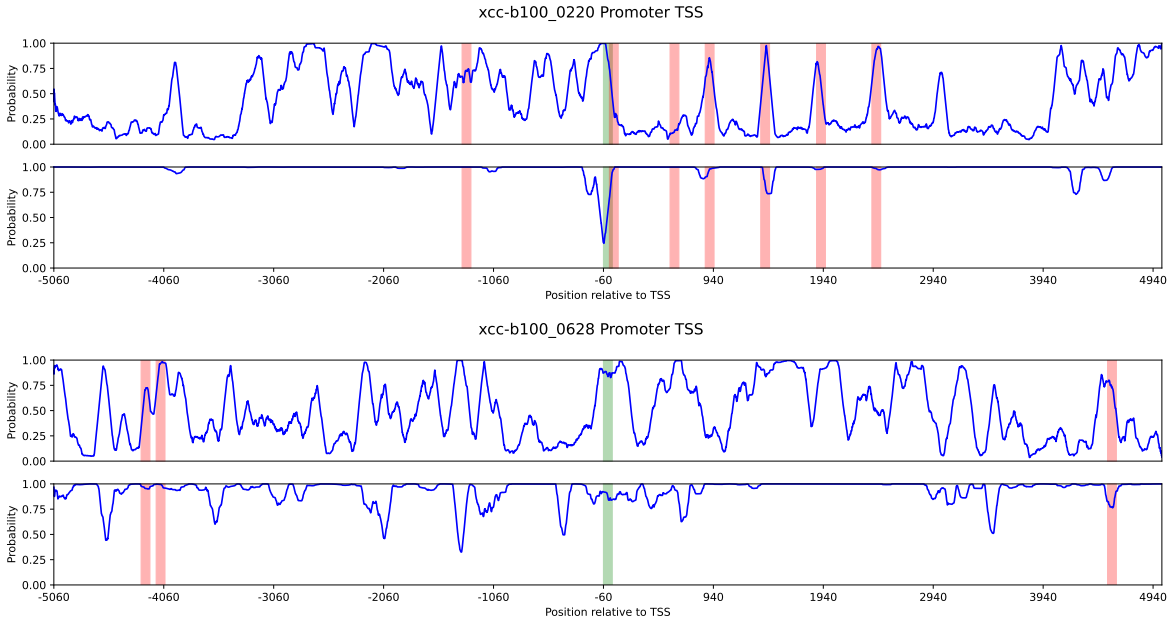


Figura 5.8: Probabilidades promedio por nucleótido predichas por DNABERT para un subconjunto de 10 promotores asociados con genes en *X. campestris*, utilizando modelos entrenados en los conjuntos de datos CDS (Superior) y SRS (Inferior). Cada gráfico representa un promotor vinculado a un gen específico. Las marcas en verde indican promotores del conjunto de datos de validación, mientras que las marcas en rojo señalan promotores adicionales incluidos en PPD.

Capítulo 6

Conclusiones y Trabajos futuros

Para el desarrollo y mejora de herramientas basadas en ML o en el análisis de secuencias, es fundamental profundizar en el conocimiento sobre los conjuntos de datos de entrenamiento utilizados para crear modelos predictivos. La limitada comprensión de las secuencias promotoras en múltiples especies bacterianas, con solo unas pocas cepas que cuentan con suficientes datos, dificulta identificar los factores que podrían generar sesgos.

Los sesgos no son poco comunes en las herramientas basadas en ML, ya que diversas etapas pueden introducir desigualdades en las tareas realizadas. Esto incluye la selección de conjuntos de datos, la optimización de hiperparámetros, los algoritmos o la arquitectura del modelo elegidos. En este trabajo, nos centramos en la curación de los conjuntos de datos, con especial énfasis en los sesgos impulsados por los datos negativos.

La variación en el contenido de GC es intrínseca a cualquier secuencia genómica, lo que representa un desafío para la creación de conjuntos de datos negativos adecuados, ya sea para una sola especie o para múltiples especies. Aunque la estandarización del contenido de GC para conjuntos de datos negativos ha sido considerada en algunos modelos [55], la corrección de conjuntos de datos negativos basada en el contenido de GC no ha sido investigada en relación con los resultados de los predictores. Los predictores para múltiples especies son inherentemente variables en su desempeño entre especies, dependiendo del conjunto de datos negativos elegido, siendo el contenido GC un factor relevante. En los modelos entrenados con

CDS, las brechas observadas entre sensibilidad y especificidad subrayan la necesidad de un mayor refinamiento. Las evaluaciones de desempeño en conjuntos de datos independientes de predictores para múltiples especies también evidenciaron este problema, con grandes brechas entre sensibilidad y especificidad por especie evaluada. En este contexto, el conjunto de datos SRS superó a los modelos entrenados con CDS, reduciendo las brechas entre las métricas mencionadas. Si bien el uso de conjuntos de datos sintéticos podría aumentar el riesgo de sobreajuste, la aplicabilidad y confiabilidad de estos modelos en datos reales necesitan ser evaluadas para garantizar su utilidad en aplicaciones del mundo real.

En los últimos años, muchas herramientas basadas en BERT han mejorado el desempeño de los predictores en el análisis de secuencias en comparación con otras técnicas de ML. En este estudio, DNABERT fue el mejor modelo en ambos conjuntos de datos. Para SRS, DNABERT mantuvo su posición como el mejor modelo; sin embargo, la normalización del contenido de GC permitió que arquitecturas menos complejas, como los modelos basados en CNN y RF, fueran competitivas en métricas de desempeño con los modelos basados en BERT, a pesar de su menor complejidad computacional. Aunque los modelos de NLP como los basados en BERT requieren grandes recursos computacionales, esto resalta la relevancia de los modelos CNN y RF para la predicción de promotores en múltiples especies con secuencias que igualan el contenido de GC.

Como trabajo futuro, se puede ampliar el análisis incorporando datasets negativos extraídos de otras regiones del genoma. Por otro lado, se propone crear un marco de evaluación más sistematizado para validar estos modelos en contextos reales, evaluándolos a nivel de genomas completos a través de un modelo único, tanto para una especie como para múltiples.

6.1. Contribuciones

- Marcelo González, Roberto E. Durán, Michael Seeger, Mauricio Araya, Nicolás Jara. GC-content bias in training datasets affects Deep Learning-based predictors in multiple species promoters. Jornada Técnica AC3E. Valparaíso, Chile. 08 de agosto, 2024. Poster

- Roberto E. Durán, Andrés Cumsille, Daryl Hernández, Marcelo González, Andrea Rodríguez-Delherbe, Vicente Saona-Urmeneta, Ester R. González, Camila Astorga-Alarcón, Óscar Guajardo-Menas, Fabián Guerrero-Maureira, Yeriel Paz, Claudia Clavero-León, Beatriz Cámara, Michael Seeger, Mauricio Araya, Nicolás Jara, Carlos Buil-Aranda. Desarrollo de software de visualización de datos, bases de datos y aplicaciones de aprendizaje de máquinas para el apoyo a la investigación genómica en microbiología. III Reunión Anual de la Sociedad de Bioinformática. Concepción, Chile. 26 al 29 de noviembre, 2024. Poster

- Marcelo González, Roberto E. Durán, Michael Seeger, Mauricio Araya, Nicolás Jara. GC-content bias in training datasets affects Deep Learning-based predictors in multiple species promoters. III Reunión Anual de la Sociedad de Bioinformática. Concepción, Chile. 26 al 29 de noviembre, 2024. Comunicación Oral

- González, M., Durán, R. E., Seeger, M., Araya, M., & Jara, N. Negative dataset selection impacts machine learning-based predictors for multiple bacterial species promoters. (Bioinformatics, JIF 2023 4.4 — Biochemical Research Methods; JCI Q1 — Biotechnology & Applied Microbiology; JCI Q1). En revisión

Apéndice A

Generación de secuencias sintéticas

Algorithm 1 Generation of a Random Sequence with Fixed GC Content

Require: L: Desired length of the sequence
Require: $GC\,\%$: Desired percentage of G and C nucleotides in the sequence
Ensure: $sequence$: A random sequence of length L with an approximate GC percentage

Round $GC\,\%$ to the nearest integer and assign to $gc\_rounded$
$at\_rounded \leftarrow 100 - gc\_rounded$
if $gc\_rounded = 0$ **then**
    $g\_percentage \leftarrow 0$
else
    $g\_percentage \leftarrow$ random integer between 0 and $gc\_rounded$
$c\_percentage \leftarrow gc\_rounded - g\_percentage$
if $at\_rounded = 0$ **then**
    $a\_percentage \leftarrow 0$
else
    $a\_percentage \leftarrow$ random integer between 0 and $at\_rounded$
$g\_count \leftarrow \left\lfloor L \cdot \frac{g\_percentage}{100} \right\rfloor$
$c\_count \leftarrow \left\lfloor L \cdot \frac{c\_percentage}{100} \right\rfloor$
$a\_count \leftarrow \left\lfloor L \cdot \frac{a\_percentage}{100} \right\rfloor$
$t\_count \leftarrow L - g\_count - c\_count - a\_count$
Construct the initial string $sequence$ as the concatenation of the following substrings:
    (i) A substring of g_count characters, each being 'g'.
    (ii) A substring of c_count characters, each being 'c'.
    (iii) A substring of a_count characters, each being 'a'.
    (iv) A substring of t_count characters, each being 't'.
Randomly shuffle the characters in $sequence$ to obtain the final sequence.
return $sequence$

Bibliografía

[1] B. Schmidt and A. Hildebrandt, "Next-generation sequencing: big data meets high performance computing," *Drug Discovery Today*, vol. 22, no. 4, pp. 712–717, 2017.

[2] S. Behjati and P. S. Tarpey, "What is next generation sequencing?," *Archives of Disease in Childhood - Education and Practice*, vol. 98, no. 6, pp. 236–238, 2013.

[3] Z. S. Guo, Q. Li, D. Bartlett, J. Yang, and B. Fang, "Gene transfer: The challenge of regulated gene expression," *Trends in molecular medicine*, vol. 14, pp. 410–8, Oct. 2008.

[4] A. Forrest, H. Kawaji, M. Rehli, K. Baillie, M. de Hoon, V. Haberle, T. Lassmann, I. Kulakovskiy, M. Lizio, M. Itoh, R. Andersson, C. Mungall, T. Meehan, S. Schmeier, N. Bertin, M. Jørgensen, E. Dimont, E. Arner, C. Schmidl, and Y. Hayashizaki, "A promoter-level mammalian expression atlas," *Nature*, vol. 507, pp. 462–70, Mar. 2014.

[5] W. Liu, J. Yuan, and C. Jr, "Advanced tools for plant biotechnology," *Nature reviews. Genetics*, vol. 14, Oct. 2013.

[6] S.-M. Yu, S.-S. Ko, C.-Y. Hong, H.-J. Sun, Y.-I. Hsing, C.-G. Tong, and T.-H. D. Ho, "Global functional analyses of rice promoters by genomics approaches," *Plant Molecular Biology*, vol. 65, no. 4, pp. 417–425, 2007.

[7] A. Moszyńska, M. Gebert, J. F. Collawn, and R. Bartoszewski, "Snps in microrna target sites and their potential role in human disease," *Open Biology*, vol. 7, p. 170019, Apr. 2017.

[8] R. Umarov and V. V. Solovyev, "Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks," *PLoS ONE*, vol. 12, 2016.

[9] B. Liu, F. Yang, D.-S. Huang, and K.-C. Chou, "iPromoter-2L: a two-layer predictor for identifying promoters and their types by multi-window-based PseKNC," *Bioinformatics*, vol. 34, pp. 33–40, Sept. 2017.

[10] H. Wenying, C. Jia, Y. Duan, and Q. Zou, "70ProPred: A predictor for discovering sigma70 promoters based on combining multiple features," *BMC Systems Biology*, vol. 12, Apr. 2018.

[11] M. S. Rahman, U. Aktar, M. R. Jani, and S. Shatabda, "iPro70-FMWin: identifying Sigma70 promoters using multiple windowing and minimal features," *Molecular Genetics and Genomics*, vol. 294, pp. 69–84, Feb. 2019.

[12] M. H. A. Cassiano and R. Silva-Rocha, "Benchmarking bacterial promoter prediction tools: Potentialities and limitations," *mSystems*, vol. 5, no. 4, 2020.

[13] R. Amin, C. R. Rahman, S. Ahmed, M. H. R. Sifat, M. N. K. Liton, M. M. Rahman, M. Z. H. Khan, and S. Shatabda, "ipromoter-bncnn: a novel branched cnn-based predictor for identifying and classifying sigma promoters," *Bioinformatics*, vol. 36, p. 4869–4875, July 2020.

[14] M. Shujaat, A. Wahab, H. Tayara, and K. T. Chong, "pcpromoter-cnn: A cnn-based prediction and classification of promoters," *Genes*, vol. 11, p. 1529, Dec. 2020.

[15] D. Hernández, N. Jara, M. Araya, R. E. Durán, and C. Buil-Aranda, "PromoterLCNN: A Light CNN-Based Promoter Prediction and Classification Model," *Genes*, vol. 13, no. 7, 2022.

[16] Y. Zhu, F. Li, X. Guo, X. Wang, L. Coin, G. Webb, J. Song, and C. Jia, "TIMER is a Siamese neural network-based framework for identifying both general and species-specific bacterial promoters," *Briefings in Bioinformatics*, May 2023.

[17] R. Chevez-Guardado and L. Peña-Castillo, "Promotech: a general tool for bacterial promoter recognition," *Genome biology*, vol. 22, p. 318, Nov. 2021.

[18] P. Zhang, H. Zhang, and H. Wu, "iPro-WAEL: a comprehensive and robust framework for identifying promoters in multiple species," *Nucleic Acids Research*, vol. 50, pp. 10278–10289, Sept. 2022.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[20] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri, "DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome," *Bioinformatics*, vol. 37, pp. 2112–2120, Feb. 2021.

[21] Z. Zhou, Y. Ji, W. Li, P. Dutta, R. Davuluri, and H. Liu, "DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome." arXiv:2306.15006, 2023.

[22] H. Dalla-Torre, L. Gonzalez, J. Mendoza-Revilla, N. L. Carranza, A. H. Grzywaczewski, F. Oteri, C. Dallago, E. Trop, B. P. de Almeida, H. Sirelkhatim, G. Richard, M. Skwark, K. Beguir, M. Lopez, and T. Pierrot, "The nucleotide transformer: Building and evaluating robust foundation models for human genomics," *bioRxiv*, 2023.

[23] K. Lee, T. Ho, N. Nui, and J.-S. Chang, "BERT-Promoter: An improved sequence-based predictor of DNA promoter using BERT pre-trained model and SHAP feature selection," *Computational Biology and Chemistry*, vol. 99, p. 107732, July 2022.

[24] S. Gama-Castro, H. Salgado, A. Santos-Zavaleta, D. Ledezma-Tejeida, L. Muñiz-Rascado, J. S. García-Sotelo, K. Alquicira-Hernández, I. Martínez-Flores, L. Pannier, J. A. Castro-Mondragón, A. Medina-Rivera, H. Solano-Lira, C. Bonavides-Martínez, E. Pérez-Rueda, S. Alquicira-Hernández, L. Porrón-Sotelo, A. López-Fuentes, A. Hernández-Koutoucheva, V. D. Moral-Chávez, F. Rinaldi, and J. Collado-Vides,

"RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond," *Nucleic Acids Research*, vol. 44, pp. D133–D143, Nov. 2015.

[25] M. Oubounyt, Z. Louadi, H. Tayara, and K. T. Chong, "DeePromoter: Robust Promoter Predictor Using Deep Learning," *Frontiers in Genetics*, vol. 10, 2019.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[27] S. Gupta, J. Dennis, R. Thurman, R. Kingston, J. Stamatoyannopoulos, and W. Noble, "Predicting human nucleosome occupancy from primary sequence," *PLoS computational biology*, vol. 4, p. e1000134, 02 2008.

[28] Z. Chen, P. Zhao, F. Li, T. T. Marquez-Lago, A. Leier, J. Revote, Y. Zhu, D. R. Powell, T. Akutsu, G. I. Webb, K.-C. Chou, A. I. Smith, R. J. Daly, J. Li, and J. Song, "ilearn: an integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of dna, rna and protein sequence data," *Briefings in Bioinformatics*, vol. 21, pp. 1047–1057, 04 2019.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018.

[30] V. Zouhar, C. Meister, J. L. Gastaldi, L. Du, T. Vieira, M. Sachan, and R. Cotterell, "A formal perspective on byte-pair encoding," 2024.

[31] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," 2022.

[32] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[33] W. Su, M.-L. Liu, Y.-H. Yang, J.-S. Wang, S.-H. Li, H. Lv, F.-Y. Dao, H. Yang, and H. Lin, "PPD: A Manually Curated Database for Experimentally Verified Prokaryotic Promoters," *Journal of Molecular Biology*, vol. 433, no. 11, p. 166860, 2021.

[34] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," 2013.

[35] M. Sethi, N. Tyagi, P. S. Kalsi, and P. Atchuta Rao, "Deep learning-based binary classification for spam detection in sms data: Addressing imbalanced data with sampling techniques," in *2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pp. 1–9, 2023.

[36] S. Jiang, J. Hu, C. L. Magee, and J. Luo, "Deep learning for technical document classification," *IEEE Transactions on Engineering Management*, vol. 71, p. 1163–1179, 2024.

[37] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[38] H. Teeling, A. Meyerdierks, M. Bauer, R. Amann, and F. Glöckner, "Application of tetranucleotide frequencies for the assignment of genomic fragments," *Environ Microbiol*, vol. 6, pp. 938 – 947, 01 2004.

[39] C. Moeckel, M. Mareboina, M. A. Konnaris, C. S. Chan, I. Mouratidis, A. Montgomery, N. Chantzi, G. A. Pavlopoulos, and I. Georgakopoulos-Soares, "A survey of k-mer methods and applications in bioinformatics," *Computational and Structural Biotechnology Journal*, vol. 23, pp. 2289–2303, 2024.

[40] C. Miller, T. Portlock, D. M. Nyaga, and J. M. O'Sullivan, "A review of model evaluation metrics for machine learning in genetics and genomics," *Frontiers in Bioinformatics*, vol. 4, 2024.

[41] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[42] M. Elgendy, *Deep Learning for Vision Systems*. Manning, 2020.

[43] E. U. H. Qazi, A. Almorjan, and T. Zia, "A one-dimensional convolutional neural network (1d-cnn) based deep learning system for network intrusion detection," *Applied Sciences*, vol. 12, no. 16, 2022.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[46] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," 2018.

[47] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," 2023.

[48] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[49] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li, "CD-HIT: accelerated for clustering the next-generation sequencing data," *Bioinformatics*, vol. 28, pp. 3150–3152, 10 2012.

[50] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

[51] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[53] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

[54] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[55] S. Zhang, A. Ma, J. Zhao, D. Xu, Q. Ma, and Y. Wang, "Assessing deep learning methods in cis-regulatory motif finding based on genomic sequencing data," *Briefings in Bioinformatics*, vol. 23, no. 1, 2021.