



UNIVERSIDAD NACIONAL DE COLOMBIA

Modelo basado en técnicas de machine learning para la clasificación de virus de ARN

Carolina Colmenares Celis

UNIVERSIDAD NACIONAL DE COLOMBIA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE SISTEMAS E INDUSTRIAL
BOGOTÁ, 2023



UNIVERSIDAD NACIONAL DE COLOMBIA

Trabajo Final de Maestría

**Modelo basado en técnicas de machine learning para
la clasificación de virus de ARN**

Carolina Colmenares Celis

Directora

CLARA ISABEL BERMUDEZ SANTANA, PH.D

Codirector

LUIS FERNANDO NIÑO VASQUEZ, PH.D

Línea de investigación

TECNOLOGÍAS COMPUTACIONALES EN BIOINFORMÁTICA

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE SISTEMAS E INDUSTRIAL

BOGOTÁ, 2023

Índice

Capítulos	Página
Resumen	IX
Agradecimientos	XIII
1. Introducción	15
1.1. Estructura del ARN	16
1.2. Clasificación y evolución de los virus de ARN	18
1.3. Metagenómica y Metatranscriptómica	20
2. Niveles de representación teórica	25
2.1. Estructuras del ARN	25
2.2. Estructura secundaria del ARN	26
2.2.1. Predicción de estructuras secundarias	28
2.3. Representaciones secundarias	31
2.3.1. Árbol ampliado	31
2.3.2. Representación HIT	31
2.3.3. Representación de árbol de <i>grano grueso</i>	32
3. Algoritmos del aprendizaje de máquina	35
3.1. Machine learning	36
3.1.1. Primera categoría: aprendizaje supervisado, no supervisados, semi-supervisado y por refuerzo	36
3.1.2. Segunda categoría: aprendizaje en línea o por lotes	37
3.1.3. Tercera categoría: aprendizaje basado en instancias o basado en modelos	37
3.2. Problema de clasificación	37
3.2.1. Desafíos en el aprendizaje de máquina	38
3.2.2. Métricas de desempeño	39
3.3. Algoritmos de clasificación	40
3.4. Ejemplos de implementaciones	49
4. Implementación de las aproximaciones de modelos y resultados	53
4.1. Datos de entrenamiento y prueba	53
4.2. Pre-procesamiento de los datos	57
4.2.1. Pre-procesamiento de los datos estándar o de entrenamiento	57
4.2.2. Pre-procesamiento datos metagenómicos o de prueba	59
4.3. Conversión de los datos a estructuras secundarias	61
4.4. Modelos	62
4.4.1. Modelo I: Perceptrón multicapa	62
4.4.2. Resultados modelo I	63
4.4.3. Modelo II: Árboles de sufijos	66
4.4.4. Resultados modelo II	67
4.4.5. Modelo III: Modelo oculto de Markov	69

4.4.6. Resultados modelo III	70
4.4.7. Modelo IV: CNN-LTSM	75
4.4.8. Resultados modelo IV	79
4.5. Resultados	83
4.6. Discusión	87
5. Conclusiones y perspectivas futuras	89
5.1. Conclusiones	89
5.2. Perspectivas futuras	90
5.2.1. Limitaciones	90
Apéndice	91
A. Grafo de De Bruijn	91
B. Matriz de Confusión	93
C. Truco del Kernel	94
D. Función escalonada	95
E. Distribución de Boltzmann	96
F. Puntuación de Información Mutua	97
G. Valor p	98
H. Algoritmo de Viterbi	99
I. Convolución	100
J. Distancia de Hamming	101
K. Puntuación de Covarianza	102
Referencias	103

“THERE IS AS YET INSUFFICIENT DATA FOR A MEANINGFUL ANSWER.”
Isaac Asimov, The Last Question

Resumen

Título: Modelo basado en técnicas de machine learning para la clasificación de virus de ARN

Los virus son las entidades biológicas más abundantes de la Tierra, pero detectarlos, aislarlos y clasificarlos ha sido todo un reto para la ciencia. Los virus de ARN patógenos causan numerosas muertes humanas, especialmente los implicados en la transmisión de enfermedades zoonóticas, lo que conduce a emergencias víricas y pandemias globales como la asociada al SARS-CoV-2. En este estudio, se explora y describen representaciones teóricas como la de árbol extendido, HIT y árbol de grano grueso para virus de ARN, basados en niveles de secuencia y estructura. Estas representaciones se utilizaron para determinar cuál de ellas demuestra un mejor potencial como entradas para un modelo de clasificación basado en técnicas de aprendizaje de máquina.

Para el diseño del modelo, se investigaron algoritmos de perceptrón multicapa, árboles de sufijos, modelos ocultos de Markov (HMM) y redes neuronales convolucionales con memoria de corto y largo plazo (CNN-LSTM). La aplicación de estos algoritmos se llevó a cabo utilizando dos conjuntos de datos. Los datos de entrenamiento consistieron en secuencias de familias de virus ARN, incluyendo *Orthomyxoviridae*, *Sedoreoviridae*, *Spinareoviridae*, *Retroviridae* y *Arteriviridae*, obtenidas de la base de datos del Centro Nacional para la Información Biotecnológica (NCBI). Los datos de prueba están comprendidos de metaviromas recolectados durante la “Expedición virológica en ecosistemas representativos de Colombia: selva húmeda tropical de la Sierra Nevada de Santa Marta”, un proyecto financiado por Colciencias en colaboración con el grupo de investigación teórica y computacional RNómica de la Universidad Nacional de Colombia.

Ambos conjuntos de datos se transformaron en las representaciones estructurales mencionadas utilizando el paquete ViennaRNA. La representación HIT mostró las mejores características para la extracción, y los modelos basados en HMMs y CNN-LSTM demostraron un rendimiento superior y potencial para clasificar metagenomas de virus ARN.

Palabras clave: Virus ARN, metagenómica, metavirómica, aprendizaje de máquina, estructuras secundarias, clasificación.

Abstract

Title: Model based on machine learning techniques for the classification of RNA viruses

Viruses are the most abundant biological entities on Earth, but detecting, isolating, and classifying them has posed a significant challenge for science. Pathogenic RNA viruses cause numerous human deaths, especially those involved in the transmission of zoonotic diseases, leading to viral emergencies and global pandemics like the one associated with SARS-CoV-2. In this study, theoretical representations such as extended tree, HIT, and coarse-grained tree are explored and described for RNA viruses, based on levels of sequence and structure. These representations were used to determine which of them demonstrates better potential as inputs for a classification model based on machine learning techniques.

For model design, algorithms including multilayer perceptrons, suffix trees, hidden Markov models (HMMs), and convolutional neural networks with short and long-term memory (CNN-LSTM) were investigated. The application of these algorithms was carried out using two datasets. The training data consisted of sequences from families of RNA viruses, including *Orthomyxoviridae*, *Sedoreoviridae*, *Spinareoviridae*, *Retroviridae*, and *Arteriviridae*, obtained from the National Center for Biotechnology Information (NCBI) database. The test data comprised metaviromes collected during the “Expedición virológica en ecosistemas representativos de Colombia: selva húmeda tropical de la Sierra Nevada de Santa Marta” a project funded by Colciencias in collaboration with the theoretical and computational research group RNomica at the National University of Colombia.

Both datasets were transformed into the mentioned structural representations using the ViennaRNA package. The HIT representation exhibited the most favorable features for extraction, and models based on HMMs and CNN-LSTM demonstrated superior performance and potential for classifying RNA virus metagenomes.

Keywords: RNA viruses, metagenomics, metaviromics, machine learning, secondary structures, classification.

Este Trabajo Final de maestría fue calificado en agosto de 2023 por el siguiente evaluador:

Fabio Augusto González Osorio PhD.
Profesor Facultad de Ingeniería
Universidad Nacional de Colombia

Agradecimientos

Ante todo, quiero dar las gracias a Dios, por sus bendiciones a lo largo de la maestría. A mi familia, especialmente mis padres y mi novio, cuyo apoyo y motivación durante este proceso me ayudaron a finalizar este trabajo. A mi hermano por su ayuda con las ilustraciones.

Me gustaría dar las gracias a mis supervisores, a la directora del trabajo, la profesora Clara Bermúdez por su constante apoyo, consejos y sugerencias a lo largo de la maestría, y durante el desarrollo del trabajo, le estoy muy agradecida por su orientación y ayuda con todos los temas académicos, que permitieron hacer de este trabajo lo que es. Agradezco también a mi co-director, el profesor Luis Fernando Niño, por su paciencia y guía con toda la parte computacional. Les agradezco a ambos la oportunidad de dejarme participar en sus grupos de investigación y laboratorios: Rnomica teórica y computacional, y el laboratorio de investigación en sistemas inteligentes - LISI. Aprecio todo el apoyo brindado en temas biológicos y computacionales, que fue fundamental para la elaboración de este trabajo.

También quiero agradecer a mis compañeros de la maestría, y especialmente a mis compañeros del grupo de Rnomica teórica y computacional, y LISI, quienes desde el inicio tuvieron la disposición de ayudarme con diferentes aspectos que eran desconocidos o nuevos para mí. Les estoy muy agradecida a todos mis compañeros de Rnomica teórica y computacional por contestar correos y mensajes en horas de la madrugada cuando estaba aprendiendo sobre alguna herramienta bioinformática nueva. Muchas gracias por sus comentarios, sugerencias y constante ayuda durante este proceso.

Agradezco a Ernesto Parra Rincón por su labor administrando el laboratorio de Biología Computacional de la Universidad Nacional de Colombia. Igualmente al Servicio Alemán de Intercambio Académico (DAAD) por la donación de equipos a la Facultad de Ciencias de la Universidad Nacional de Colombia, y al Centro de Excelencia en Computación Científica (CECC) por proveer recursos para la administración de los equipos.

Introducción

Los virus son las entidades biológicas más abundantes de la tierra, pero las dificultades para detectar, aislar y clasificarlos ha sido un reto para la ciencia. Por ejemplo, aún prevalecen diferentes sistemas de clasificación que varían desde el tipo de componente del genoma viral [142], hasta el establecido por el comité internacional de taxonomía de virus [117, 143]. Si bien la tarea de clasificación en categorías no es fácil para virus conocidos, para los virus emergentes o no clasificados el panorama es más complejo, lo cual impide realizar estudios exhaustivos del viroma mundial [49]. Los virus tienen diferentes funciones, en particular, los virus con genoma ARN son de suma importancia en la vigilancia epidemiológica viral ya que causan muchas enfermedades como: síndrome respiratorio agudo severo (SARS), influenza, resfriado común, hepatitis C, polio, entre otras [2]. Además, copian su genoma usando su propia polimerasa dependiente de ARN (dRdP por su siglas en inglés), que produce un tasa alta de mutaciones en las copias virales, haciendo que estos tengan una capacidad evolutiva única, la cual les permite adaptarse rápidamente a cambios ambientales, o farmacológicos [1]. De igual forma, los virus de ARN patógenos causan muchas muertes en humanos, ya que agrupan los virus que principalmente se han involucrado en la transmisión de enfermedades zoonóticas, causando emergencias virales y pandemias en todo el mundo como la asociada al SARS-CoV-2 [3].

Las enfermedades infecciosas emergentes (EIE) se han definido como enfermedades cuya incidencia aumentó en las últimas décadas o se prevé que aumenten en el futuro previsible [19]. Esta definición incluye infecciones conocidas con nuevas propiedades, o infecciones conocidas en nuevas regiones geográficas, infecciones que no se reconocieron previamente y nuevas infecciones que resultan del paso de patógenos desde un reservorio animal a los humanos. Se cree que la mayoría de las EIE se originan en animales, de los cuales los de vida silvestre son la fuente más importante para el posible salto de especie y despliegue de brotes infecciosos en humanos [20, 21]. Debido al efecto de la globalización, el cambio climático y la intensa movilización de la población mundial, es probable que nuevos virus emerjan y que se propaguen tan rápido como ocurrió con el SARS-CoV-2. Dentro de las estrategias de vigilancia molecular viral, la identificación temprana de los patógenos no solo puede ayudar a prevenir brotes, sino también ayudar al diseño rápido y eficaz de antivirales, vacunas y otras medicinas de tipo curativo [22, 23]. Hoy por hoy, parte de los programas de vigilancia en salud se basan en el secuenciamiento masivo de información genómica y metagenómica de virus y de muestras complejas que puedan acelerar el proceso de descubrimiento de nuevos virus. Sin embargo, los retos se mantienen desde el punto de vista de las validaciones de la información y de las dificultades de clasificación

de virus emergentes. Por ejemplo aplicaciones como VirMAP [144], sirven para identificar y clasificar virus de baja cobertura y altamente divergentes dentro de conjuntos de datos metagenómicos [144].

VirMAP no está basado en técnicas de machine learning, sino que utiliza una variedad de técnicas que incluyen un algoritmo de ensamblaje de mapeo que combina la información de alineación de nucleótidos y aminoácidos de una manera escalonada para construir super-scaffolds similares a virus, un sistema de fusión diseñado para hibridar un ensamblaje de mapeo y un ensamblaje de-novo, un algoritmo de mejora que fusiona y reconstruye iterativamente la información de contigs, y un algoritmo de clasificación taxonómica que se centra en torno a un sistema de puntuación de bits por base [144]. La forma en la que VirMAP realiza la clasificación, difiere con métodos como drVM [146], Vipie [145], ViromeScan [147], MetaPhlAn [148], FastViromeExplorer [149], VirusTAP [150] y Virus-Seeker [151] que se basan en la clasificación de lecturas individuales y/o contigs mediante alineamientos o Kaiju [152] y Kraken [153] que utilizan análisis de k -meros [144]. Todos estos enfoques se ven limitados por problemas similares, entre los que se incluyen: lecturas, contigs o k -meros que se asignan igual de bien a diferentes genomas, degeneración de codones que permite que dos secuencias de nucleótidos de baja identidad codifiquen polipéptidos idénticos, cobertura de secuencias muy variable que afecta a la heurística del ensamblador, grandes cantidades de secuencias contaminantes y bases de datos que carecen de una cobertura suficiente del árbol de la vida viral [144].

Es por ello, que en los últimos años se ha venido explorando nuevas metodologías computacionales de clasificación basadas en aprendizaje de máquina o machine learning (ML). Sin embargo, las aplicaciones existentes se basan en el análisis de secuencia de los genomas o partes de los genomas virales, que carecen de información sobre otros niveles de representación. En este trabajo final de maestría se exploran otras representaciones teóricas para proponer y revisar el uso de estos como descriptores de las secuencias. El desarrollo de este tipo de modelos es prometedor ya que pueden ayudar en el diagnóstico y clasificación de enfermedades, al seguimiento de enfermedades a nivel molecular y la investigación de biomarcadores de enfermedades potenciales [14]. De este modo, se busca analizar y comparar distintas variables que describan diferentes representaciones teóricas a nivel de secuencia y estructura para los virus de ARN. Dichas variables se toman de las representaciones teóricas de estructura de la molécula de ARN (o RNA por su sigla en inglés).

1.1. Estructura del ARN

El ácido ribonucleico (ARN) es una molécula esencial en diversas funciones biológicas de codificación, descodificación, regulación y expresión de genes. Al igual que el ADN, el ARN se ensambla como una cadena de nucleótidos, pero a diferencia del ADN, el ARN se encuentra en la naturaleza como una sola cadena plegada sobre sí misma, en lugar de una cadena doble emparejada. Los organismos celulares utilizan el ARN mensajero (ARNm) para transmitir información genética (utilizando las bases nitrogenadas de guanina, uracilo, adenina y citosina, denotadas por las letras G, U, A y C) que dirige la síntesis de proteínas específicas. La función principal del ARN es crear proteínas mediante la traducción. El ARN transporta información genética que los ribosomas traducen en diversas proteínas necesarias para diferentes procesos celulares. El ARNm, el ARNr y el

ARNt son los tres tipos principales de ARN que intervienen en la síntesis de proteínas. El ARN también sirve como material genético primario para los virus. Otras funciones son la edición del ARN, la regulación génica y la interferencia del ARN. Estos procesos son llevados a cabo por un grupo de pequeños ARN reguladores, entre los que se incluyen los pequeños ARN nucleares, los microARN y los pequeños ARN de interferencia [47].

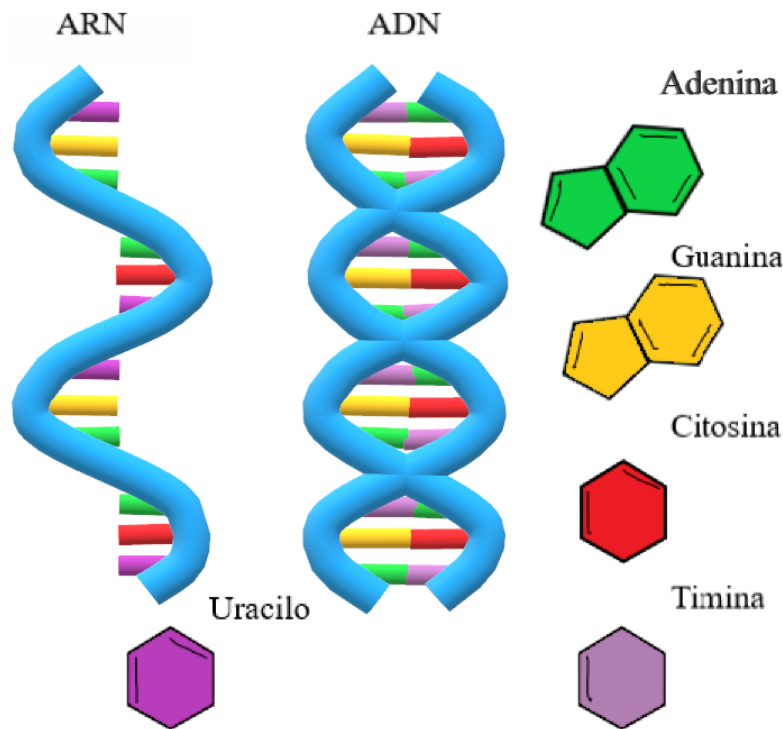


Figura 1: Estructura básica del ADN y ARN. Imagen propia.

La estructura química del ARN es muy similar a la del ADN, pero difiere en tres aspectos principales (Fig. 1):

- A diferencia del ADN bicatenario, el ARN suele ser una molécula monocatenaria o de una sola cadena (ssARN) [42] en muchas de sus funciones biológicas y está formado por cadenas de nucleótidos mucho más cortas. Sin embargo, puede formarse ARN bicatenario o de doble cadena (dsARN).
- Mientras que la *columna vertebral* de azúcar - fosfato del ADN contiene desoxirribosa, el ARN contiene ribosa [43].
- La base complementaria de la adenina en el ADN es la timina, mientras que en el ARN es el uracilo, que es una forma no metilada de la timina [44].

Al igual que el ADN, el ARN puede transportar información genética, dando lugar a que muchos virus codifiquen su información genética mediante un genoma de ARN. Los virus de ARN tienen genomas compuestos de ARN que codifican una serie de proteínas. Algunas de esas proteínas replican el genoma vírico, mientras que otras protegen el genoma a medida que la partícula vírica se desplaza a una nueva célula huésped. Los viroides son otro grupo de patógenos, pero están formados únicamente por ARN, no codifican ninguna

proteína y se replican mediante la polimerasa de una célula vegetal huésped [45].

1.2. Clasificación y evolución de los virus de ARN

En 1967 se habían propuesto clasificaciones de dos tipos principales: las destinadas a resolver problemas prácticos y que se basan en convenciones, y las destinadas a resolver problemas teóricos, que se basan en teorías y en las que las clases se comprueban mediante experimentos. Hasta ese momento se había intentado elaborar una clasificación preliminar de los virus del segundo tipo, la cual se basa en las teorías de la biología molecular, con el uso de comparaciones informáticas de los pesos moleculares y las relaciones de base de los ácidos nucleicos virales para asignar los virus a grupos que muestran un alto grado de correlación con las agrupaciones basadas en la hibridación de ácidos nucleicos, las reacciones cruzadas serológicas y las propiedades fenotípicas [35]. Unos años más tarde en 1971, [36] realiza una revisión sobre las diferentes clasificaciones que se habían llevado a cabo desde 1966 sobre cómo definir y organizar las características que un virus debía tener, dentro de las cuales estaban los siguientes rasgos distintivos: (i) Presencia de un único ácido nucleico, (ii) incapacidad para crecer y dividirse, (iii) reproducción únicamente a partir del material genético, (iv) ausencia de enzimas para el metabolismo energético, (v) ausencia de ribosomas, (vi) ausencia de información para la producción de enzimas en el ciclo energético, (vii) ausencia de información para la síntesis de las proteínas ribosómicas, y (viii) la ausencia de información para la síntesis del ARN ribosómico y del ARN de transferencia.

Existen otros sistemas de calificación como los propuestos por Baltimore, en la que se dividen todos los virus en siete grupos según como se produce el ARNm viral. En el caso de los grupos I, II, VI y VII de Baltimore, el genoma es o se convierte en dsDNA, que luego pasa a ser mRNA mediante la acción de la RNA polimerasa dependiente de DNA. En el caso de los grupos Baltimore III, IV y V, el genoma es o se convierte en +ssRNA, que es mRNA, mediante la acción de la RNA polimerasa dependiente de RNA. También hay un sistema que se basa en su tipo de nucleótido, es decir, esta clasificación divide los virus según su material genómico en virus de ADN y ARN. Los grupos virales de Baltimore I, II y VII se consideran virus de ADN, y los grupos virales restantes se consideran virus de ARN. También está la clasificación del dominio del huésped, en la cual se agrupan los virus según el dominio del huésped que infectan. Se forman tres grupos: virus eucariotas, bacterianos y arqueales [6].

Actualmente, la clasificación de los virus se basa en la recopilación y comparación de varios caracteres que describen el virus y que pueden utilizarse para distinguir un virus de otro [117, 118, 119]. Los caracteres pueden consistir en cualquier propiedad o característica del virus, e incluyen la composición molecular del genoma; la estructura de la cápside del virus y si está o no envuelto; el programa de expresión génica utilizado para producir proteínas virales; el rango de hospedadores; la patogenicidad; y la similitud de secuencia. Aunque todos los caracteres son importantes para determinar las relaciones taxonómicas, las comparaciones de secuencias utilizando tanto la similitud de secuencias por pares como las relaciones filogenéticas se han convertido en uno de los principales conjuntos de caracteres utilizados para definir y distinguir los taxones de virus [117].

Sin embargo, la clasificación taxonómica de los virus y la denominación de los taxones de virus es responsabilidad del Comité Internacional de Taxonomía de los Virus (ICTV o International Committee on Taxonomy of Viruses). La División de Virología de la Unión Internacional de Sociedades de Microbiología ha encomendado al ICTV la tarea de desarrollar, perfeccionar y mantener una taxonomía universal de los virus [117]. Este comité establece directrices para la clasificación taxonómica de los virus y aprueba la taxonomía y los nombres propuestos antes de que se conviertan en oficiales. El ICTV se creó en 1966 y está formado por una serie de directivos, presidentes de subcomités y miembros electos que supervisan la clasificación taxonómica y la denominación. Hay seis subcomités, cada uno de los cuales abarca un grupo diferente de virus, diferenciados por el tipo de huésped al que infecta el virus (animal, vegetal, fúngico y protista, o bacteriano y arqueano) y la composición molecular del genoma del virus. Cada subcomité contiene una serie de grupos de estudio, generalmente uno por familia de virus, que desarrollan los criterios de demarcación utilizados para definir cada nuevo taxón, y evalúan las propuestas utilizando esos criterios para la clasificación y la denominación. Estos criterios de demarcación se basan en las propiedades genéticas y biológicas de los miembros del taxón, y abarcan cada uno de los rangos taxonómicos reconocidos actualmente por el ICTV: orden, familia, subfamilia, género y especie [117]. Esta jerarquía coincide con sus historias evolutivas, con la diferencia de que los virus no pueden clasificarse normalmente por encima del nivel de familia [117]. Además, es probable que los virus tengan más de un origen. Los grandes virus de ADN pueden haberse originado en última instancia a partir de bacterias parásitas o eucariotas primitivos [117, 120, 121, 122], los retrovirus pueden derivar de elementos móviles en células de vertebrados [117, 123, 124], mientras que los virus con genomas de ARN pueden descender de una forma de vida basada únicamente en el ARN que precedió a la aparición de procariotas, arqueas y eucariotas [117, 125, 126].

El ICTV clasifica los virus de ARN como aquellos que pertenecen al Grupo III, Grupo IV o Grupo V del sistema de clasificación de Baltimore, este sistema de clasificación divide todos los virus en siete grupos de acuerdo a como se produce el ARNm viral. En el caso de los grupos I, II, VI y VII de Baltimore, el genoma es o se convierte en dsADN, que luego pasa a ser ARNm mediante la acción de la ARN polimerasa dependiente de DNA. En el caso de los grupos Baltimore III, IV y V, el genoma es o se convierte en +ssARN, que es ARNm, mediante la acción de la ARN polimerasa dependiente de ARN [5]. Además de esto, los virus de ARN de una sola cadena pueden clasificarse a su vez, según el sentido o polaridad de su ARN, en virus de ARN de sentido negativo y virus de ARN de sentido positivo o ambisentido. El ARN viral de sentido positivo es similar al ARNm y, por tanto, puede ser traducido inmediatamente por la célula huésped. El ARN viral de sentido negativo es complementario al ARNm y, por tanto, debe ser convertido en ARN de sentido positivo por una ARN polimerasa dependiente de ARN antes de su traducción [37]. Los virus de ARN de doble cadena representan un grupo diverso de virus que varían ampliamente en cuanto al rango de huéspedes (humanos, animales, plantas, hongos, y bacterias), el número de segmentos del genoma (de uno a doce) y la organización del virión (número de triangulación, capas de la cápside, etc.) [38].

Debido a que la clasificación de los virus se basa principalmente en el tipo de genoma y en el número y organización de los genes, la clasificación de los virus de ARN resulta difícil ya que suelen tener tasas de mutación muy elevadas en comparación con los virus de ADN [39] —lo cual es un resultado de que las ARN polimerasas virales carezcan de

la capacidad de corrección de pruebas de las ADN polimerasas [40, 41]—. A pesar de los avances que se han venido desarrollando en cuanto a la clasificación y análisis de secuencias virales, el análisis de secuencias metavirómicas es uno de los retos algorítmicos más demandantes en cuanto a la relación desempeño versus costo computacional, tiempo de ejecución, precisión y eficiencia de los resultados [4].

Los virus de ARN replican sus genomas utilizando la ARN polimerasa dependiente de ARN (RdRp) codificada por el virus. El genoma de ARN es la plantilla para la síntesis de cadenas adicionales de ARN (una molécula de ARN es la plantilla y se producen moléculas de ARN). Durante la replicación de los virus ARN, hay al menos tres tipos de ARN que deben sintetizarse: el genoma, una copia del genoma y los ARNm. Algunos virus ARN también sintetizan copias de ARNm subgenómicos. La RdRp es la pieza clave de todos estos procesos [46]. Los genomas de los virus ARN tienen algunas características generales comunes. En estos hay uno o más marcos de lectura abiertos que codifican las proteínas virales, pero también hay regiones de ARN que no codifican proteínas. Estas regiones no codificantes (NCR o non-coding regions) o regiones no traducidas (UTR o untranslated regions) suelen estar muy conservadas dentro de una familia de virus, lo que indica que tienen funciones importantes [46].

Puesto que un virus no solo puede haber evolucionado a partir de un sector determinado de ADN dentro de la célula huésped, sino también podría haberse obtenido del correspondiente ARN mensajero que contiene la misma información, tanto cualitativa como cuantitativa, en teoría, distintos virus podrían haberse originado a partir de ácidos nucleicos de naturaleza diferente pero complementaria. Los virus, como es generalmente aceptado, derivan del ácido nucleico de su huésped [36].

1.3. *Metagenómica y Metatranscriptómica*

La secuenciación completa del genoma ha estado estrechamente vinculada a los virus. Por ello, el primer genoma secuenciado en la historia de la humanidad correspondió a un virus con genoma ARN de cadena sencilla conocido como el bacteriófago MS2 [154] seguido por el bacteriófago φ X174 [155]. De allí en adelante muchos esfuerzos se han realizado para completar el secuenciamiento de muchos genomas, que en particular pueden ser cultivables pero que para muchas muestras complejas el secuenciamiento se vuelve un reto a nivel metatranscriptómico y metagenómico. La metatranscriptómica es la caracterización del contenido de ARN de las muestras, a diferencia del contenido de ADN, que se analiza mediante enfoques de metagenómica [157]. La metatranscriptómica puede utilizarse para identificar a los miembros microbianos de un ecosistema y, al mismo tiempo, obtener información funcional sobre una comunidad microbiana, dado que la metatranscriptómica utiliza transcritos de ARN en lugar de ADN, puede proporcionar información sobre la naturaleza dinámica de una comunidad y permitir extraer conclusiones sobre cómo los cambios en el entorno inducen alteraciones en la expresión génica de toda la comunidad [157, 158, 159]. Aunque esta es un área que ha ido transformando y ampliando el conocimiento de la virosfera conocida [156], el uso de esta para el descubrimiento de virus se enfrenta a importantes retos. Por lo tanto, hay algunas cuestiones que se deberían tener en cuenta en el descubrimiento de virus de ARN metatranscriptómico: (i) identificación de virus contaminantes, (ii) definición de asociaciones de huéspedes, (iii) clasificación de

metagenomas virales, y (iv) detección de virus altamente divergentes [156].

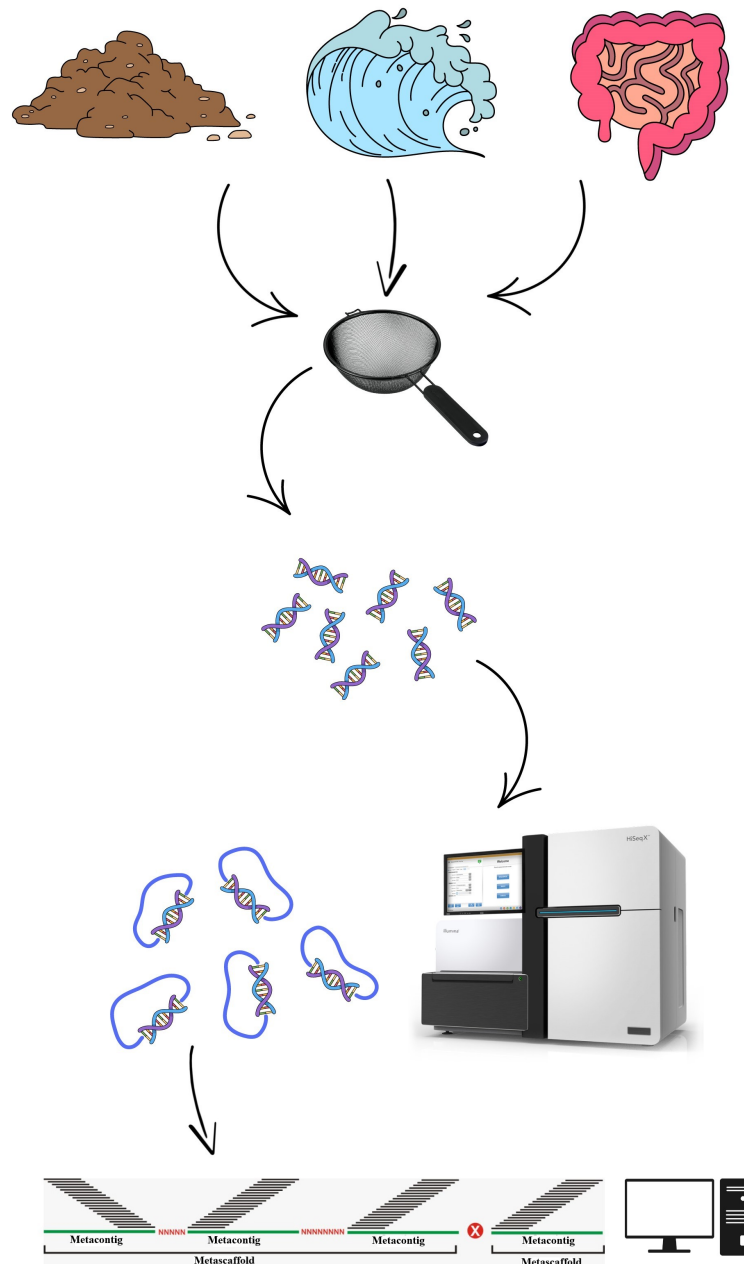


Figura 2: Ilustración del proceso de secuenciación de metagenomas.

La metagenómica se define como el análisis genético directo de los genomas contenidos en una muestra medioambiental o clínica mediante un método denominado secuenciación, el cual permite acceder a la composición genética funcional de las comunidades microbianas y, por tanto, ofrece una descripción mucho más amplia que los estudios filogenéticos, que a menudo se basan únicamente en la diversidad de un gen [70]. Este proceso consiste en extraer directamente muestras tomadas del medio ambiente posterior a un filtrado, para que luego, dichas muestras sean secuenciadas tras multiplicación por clonación en un

enfoque denominado *shotgun sequencing*¹ (Fig. 2). De este modo, estas secuencias cortas pueden unirse de nuevo mediante métodos de ensamblaje para deducir los genomas individuales o partes de genomas que constituyen la muestra medioambiental original. Esta información puede utilizarse para estudiar la diversidad de especies y el potencial funcional de la comunidad microbiana del entorno [48]. Los datos generados por los experimentos metagenómicos son bastante grandes e inherentemente ruidosos, ya que contienen datos fragmentados que representan múltiples especies [48, 49]. A pesar del reto que representa en la bioinformática la recopilación, conservación y extracción de información biológica a partir de estos conjuntos de datos [50], la secuenciación metagenómica es especialmente útil en el estudio de las comunidades víricas. Debido a que los virus carecen de un marcador filogenético universal compartido (como el ARN 16S para bacterias y arqueas, y el ARN 18S para eucariotas) [85], la única forma de acceder a la diversidad genética de la comunidad vírica a partir de una muestra ambiental es mediante la metagenómica, consiguiendo así proporcionar cada vez más información sobre la diversidad y la evolución viral [50, 52].

Esta área de estudio se ha visto muy beneficiada por las tecnologías de secuenciación de bajo costo y alto rendimiento, pero la enorme cantidad de datos generados hace que análisis como el ensamblaje de novo consuman demasiados recursos computacionales [127]. Motivados por esta necesidad, Li *et al.* [127] lanzaron en el 2014 MEGAHIT, el cual es un ensamblador de nueva generación que permite ensamblar datos metagenómicos grandes y complejos de forma eficiente en tiempo y costes [127, 128]. MEGAHIT ensambla los datos como un todo, es decir, no es necesario ningún procesamiento previo, como la partición y la normalización [127]. Este programa utiliza *grafos de De Bruijn*² sucintos (SDBG), que son representaciones comprimidas de grafos de De Bruijn, y está basado en un algoritmo paralelo rápido para la construcción de SDBG. El propósito del algoritmo es construir una estructura de datos específica SDBG, la cual representa las relaciones entre las secuencias de los datos metagenómicos, permitiendo el proceso de ensamblaje. El obstáculo de este algoritmo radica ordenar un conjunto de $(k + 1)$ -meros³, es decir, la clasificación de un conjunto de $(k + 1)$ -secuencias es el paso que consume más tiempo o recursos informáticos. En este tipo de grafos las aristas representan conexiones entre diferentes k -meros o subsecuencias. Estas aristas deben ordenarse en función de sus prefijos de longitud k en orden lexicográfico inverso, i.e., ordenarlos en la dirección opuesta, del valor más alto al más bajo [127, 128].

El análisis metagenómico de los virus sugiere nuevas pautas de evolución, que modifican las ideas previas sobre la composición del mundo vírico y revela cada vez más nuevos grupos de virus y agentes similares [53], a través de muestras de ADN ambiental o muestras de ADN clínico obtenidas de un huésped o reservorio natural [55, 56]. Los métodos metagenómicos pueden aplicarse al estudio de los virus en cualquier sistema y se han utilizado para describir diversos virus asociados a tumores cancerosos, entornos extremos,

¹*Shotgun sequencing* es un método utilizado para secuenciar cadenas aleatorias de ADN. En este método el ADN se fragmenta aleatoriamente en numerosos segmentos pequeños, que se secuencian utilizando el método de terminación en cadena para obtener lecturas. Al realizar varias rondas de esta fragmentación y secuenciación, se obtienen múltiples lecturas superpuestas para el ADN objetivo. Luego de esto, programas informáticos utilizan los extremos superpuestos de las distintas lecturas para ensamblarlas en una secuencia continua [51].

²Ver apéndice A.

³Los k -meros son subsecuencias cortas de longitud k derivadas de los datos de secuenciación de entrada. En este caso, $(k + 1)$ -meros se refiere a las subsecuencias con una longitud de $k + 1$.

ecosistemas terrestres, entre otros [57]. Por lo anterior, los enfoques metagenómicos virales ofrecen nuevas oportunidades para generar caracterizaciones objetivas de las poblaciones virales en diversos organismos y entornos [54]. Sin embargo, muchos métodos estándar de la bioinformática no son fácilmente aplicables en el análisis de metagenomas virales ya que tienden a subestimar la existencia de datos nuevos, lo cual es uno de los mayores limitantes si se considera que la mayoría de las secuencias virales no tienen una similitud significativa con nada conocido, y en donde de una muestra metavirómica puede encontrarse del 60 % al 90 % de secuencias no conocidas [4, 69]. El análisis metavirómico ha resultado ser más complejo que el análisis virómico, por ello, resulta necesario desarrollar enfoques que difieran a los métodos tradicionales, es decir, a métodos que suelen ser usados con datos genómicos [4, 70].

Los datos metagenómicos de genomas virales tipo ARN utilizados en este trabajo son producto de un trabajo previo realizado por [4]. Dichas secuencias fueron secuenciadas con recursos del proyecto financiado por Colciencias “Expedición virológica en ecosistemas representativos de Colombia: selva húmeda tropical de la Sierra Nevada de Santa Marta” desarrollado en colaboración con el grupo de investigación RNómica teórica y computacional de la Universidad Nacional de Colombia. La intención de este trabajo fue poder dar respuesta a la pregunta de investigación: ¿qué tipo de representación secundarias se puede utilizar para la clasificación de virus de ARN en modelos basados en técnicas de machine learning? Estudiar diferentes algoritmos y modelos basados en técnicas de machine learning, permite realizar un análisis y comparación de las diferentes formas de representación secundaria de los virus, de tal forma que se logre estudiar la viabilidad y utilidad de dichas estructuras secundarias en el proceso de clasificación de los virus de ARN.

Se dividieron en dos los conjuntos de datos a utilizar. El primero son los datos metagenómicos ya mencionados, y los segundos son datos tomados de la base de datos de genomas virales almacenados en el NCBI o Centro Nacional de la Información Biotecnológica, con el propósito de tener una base estándar para el entrenamiento y validación de los modelos revisados en este trabajo. Con ambos conjuntos de datos se implementaron diferentes metodologías para construir las representaciones teóricas de las estructuras ARN confinadas con herramientas bioinformáticas como: CD-HIT, Clustal-O, AliView, RNAz, entre otras. Por medio del paquete ViennaRNA se logró finalmente computar las estructuras secundarias virales ARN de interés bajo las representaciones de árbol ampliado, HIT y árbol de *grano grueso*.

Dado que dentro de los resultados conseguidos en [4] está que los metagenomas virales tienen representantes de las familias: *Orthomyxoviridae*, *Sedoreoviridae*, *Spinareoviridae*, *Retroviridae* y *Arteriviridae* se busca realizar una clasificación de estos metagenomas haciendo uso de otras notaciones de estructura, ayudado del entrenamiento de modelos basados en técnicas de machine learning con la base estándar. Por medio de este trabajo se pretendió mostrar un nuevo enfoque de clasificación de secuencias virales, tomando otras notaciones de estructura secundaria para romper el paradigma de uso de secuencias o estructura primaria del genoma viral como base para los métodos de lenguaje de procesamiento natural o de deep learning y por ende, nos interesa estudiar la viabilidad y utilidad de otras notaciones y niveles de representación teórica para la clasificación de virus.

Niveles de representación teórica

A causa de la alta variabilidad de los virus de ARN, estos suelen ser los principales culpables de emergencias virales, por ello surge un interés en estudiar diferentes formas de clasificación, ya que el desarrollo de modelos es muy recomendable para el diagnóstico y clasificación de enfermedades, el seguimiento de enfermedades a nivel molecular y la investigación de biomarcadores de enfermedades potenciales [14]. También ayudan a determinar la patogenicidad, el desarrollo de vacunas, el estudio de la epidemiología y estudiar la fármaco-resistencia [15]. De este modo, se hace necesario buscar, analizar y comparar distintas variables que describan diferentes representaciones teóricas a nivel de secuencia y estructura para los virus de ARN. Dichas variables se toman de las representaciones teóricas de la estructura.

2.1. Estructuras del ARN

La estructura del ácido nucleico hace referencia a la estructura de los ácidos nucleicos, como el ADN y el ARN, y la estructura de los ácidos nucleicos suele dividirse en cuatro niveles diferentes: primario, secundario, terciario y cuaternario.

- **Representación primaria:** Consiste en una secuencia lineal de nucleótidos unidos entre sí por enlaces fosfodiéster. La notación que se usa para esta representación en el ARN es como se observa en la figura 3.

```
GGGCUAUUAGCUCAGUUGGUUAGAGCGCACCCCUGAU
AAGGGUGAGGUCGCUGAUUCGAAUUCAGCAUAGCCCA
```

Figura 3: Representación primaria de ARN es la secuencia de nucleótidos (es decir, cuatro bases A, C, G y U) en el polímero monocatenario de ARN.

- **Representación secundaria:** Consiste en un único polinucleótido. El emparejamiento de bases en el ARN se produce cuando éste se pliega entre regiones complementarias. En las moléculas de ARN suelen encontrarse tanto regiones monocatenarias como bicatenarias. Esta estructura cuenta con múltiples representaciones

equivalente como punto-corchete o *dot-bracket*, mountain plot (Fig. 4), representaciones de árbol, la representación de Fontana [16] y la de Shapiro [17].

- **Representación terciaria:** La estructura terciaria se refiere a la ubicación de los átomos en el espacio tridimensional, teniendo en cuenta las restricciones geométricas y estéricas. Es un orden superior a la estructura secundaria, en la que se produce un plegamiento a gran escala en un polímero lineal y toda la cadena se pliega en una forma tridimensional específica [61].
- **Representación cuaternaria:** Se refiere a las interacciones entre moléculas de ácido nucleico separadas, o entre moléculas de ácido nucleico y proteínas. Mientras que muchas proteínas tienen estructura cuaternaria, la mayoría de las moléculas de ARN sólo tienen estructura primaria o secundaria y funcionan como moléculas individuales en lugar de como estructuras de varias subunidades [62].

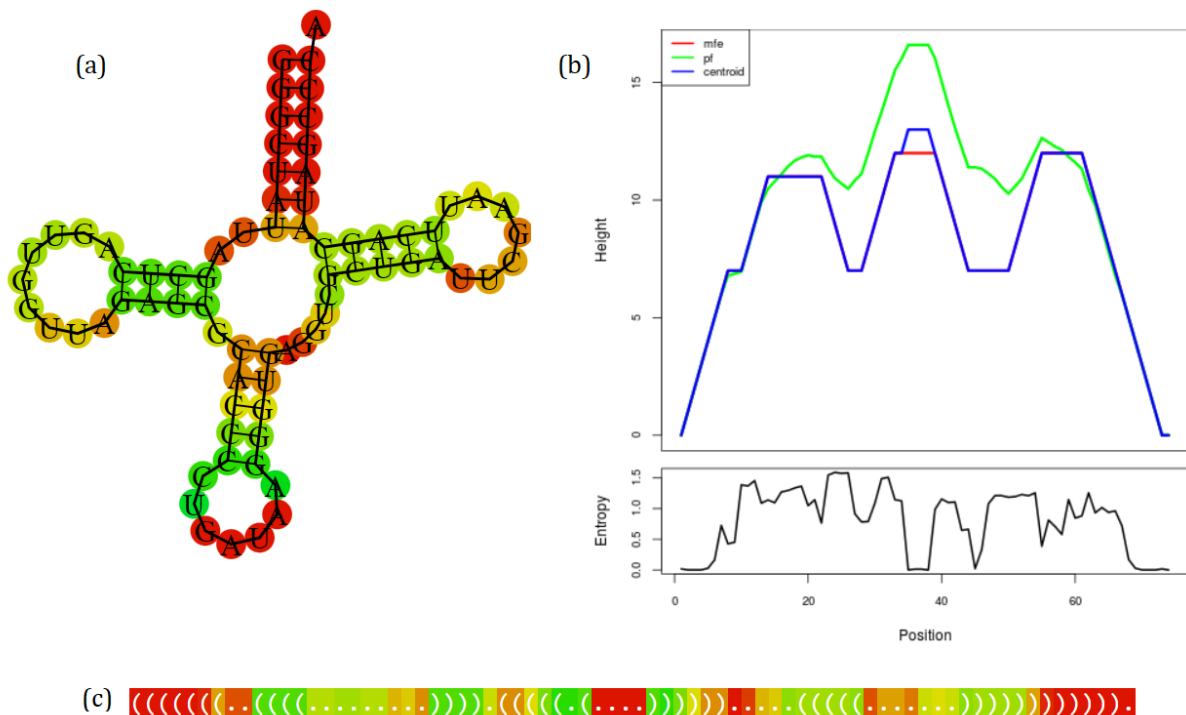


Figura 4: Representación secundaria de ARN (a) estructura de árbol, (b) mountain plot y (c) notación de dot-bracket. La imagen fue obtenida de RNAFold Server <http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi> para una cadena de ARN dada.

2.2. Estructura secundaria del ARN

Las secuencias de ácidos nucleicos monocatenarios contienen muchas regiones complementarias que pueden formar dobles hélices cuando la molécula se pliega sobre sí misma. El patrón resultante de tramos de doble hélice intercalados con bucles es lo que se denomina

estructura secundaria de ARN o ADN [60]. Estas estructuras son importantes porque pueden influir en su función y en sus interacciones con otras moléculas [79]. Las estructuras secundarias del ARN constan de dos clases distintas de residuos: los que se incorporan a las regiones de doble hélice (los llamados tallos o stacks) y los que no forman parte de las hélices. En el caso del ARN, las regiones de doble hélice están formadas casi exclusivamente por pares Watson-Crick C-G y A-U, así como por los pares G-U, ligeramente más débiles. Todas las demás combinaciones de nucleótidos emparejados, denominadas pares no canónicos, no se tienen en cuenta en la predicción de la estructura secundaria, aunque se dan, especialmente en los motivos de estructura terciaria [60].

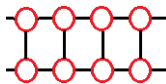
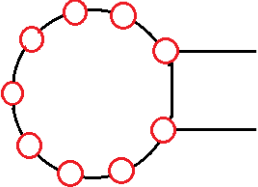
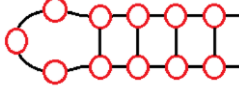
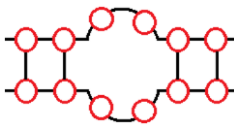
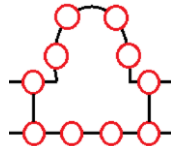
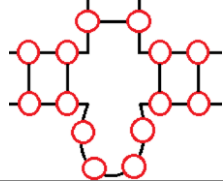
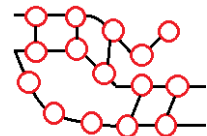
Estructura	Descripción	Representación
Stack	Región con bases apareadas.	
Loop o Bucle	Región incluida en un stack donde las bases no están apareadas.	
Hairpin loop o Bucle en horquilla	Estructura en donde regiones cercanas de bases complementarias se aparean, separadas por una región no apareada que permite que la secuencia se doble para formar una stack.	
Internal loop o Bucle interno	Estructura en donde hay regiones no apareadas en ambos lados de la hebra. Puede ser simétrico o asimétrico.	
Bulge o protuberancia	Estructura en donde hay una región no apareada en un solo lado de la hebra.	
Multiple-loop o Bucle múltiple	Región de la estructura en donde se juntan múltiples stacks.	
Pseudonudo	Variación de un loop o bucle en donde solo una parte del bucle sí está apareada.	

Tabla 1: Elementos estructurales de las representaciones secundarias.

Una estructura secundaria es principalmente una lista de pares de bases Ω . Para garantizar que la estructura es factible, una estructura secundaria válida debe cumplir las siguientes restricciones [60]:

1. Una base no puede participar en más de un emparejamiento de bases, es decir, Ω es un emparejamiento en el conjunto de posiciones de la secuencia.

2. Las bases emparejadas entre sí deben estar separadas por al menos 3 bases (no emparejadas).
3. No hay dos pares de bases (i, j) y $(k, l) \in \Omega$ *cruzados* en el sentido de que $i < k < j < l$. Los emparejamientos que no contienen bordes cruzados se conocen como emparejamientos de bucle o coincidencias circulares.

Las estructuras secundarias también pueden representarse fácilmente como árboles (Fig. 4, (a)), donde los nodos internos representan pares de bases y las hojas representan nucleótidos no apareados. La cadena de estructura dot-bracket es un árbol representado por una cadena de paréntesis (pares de bases) y puntos para los nodos de las hojas (nucleótidos no apareados) (Fig. 4, (c)). De forma alternativa, existen otros tipos de representaciones tales como: árbol ampliado, representación HIT o de Fontana [16] y representación de árbol de *grano grueso* o de Shapiro [17], siendo estas últimas tres las de interés en este trabajo, sobre las cuales se ahondará en la siguiente sección. Estas estructuras pueden ser descritas a partir de motivos o *motifs* estructurales que se pueden clasificar principalmente como: stack, loop, hairpin loop, internal loop, bulge, multi-loop o pseudonudo (ver Tabla 1). Un motivo es un patrón de secuencias cortas conservadas asociado a distintas funciones de una proteína o ADN. A menudo se asocia a un lugar estructural distinto que desempeña una función concreta [129].

2.2.1. Predicción de estructuras secundarias

Las predicciones precisas de las estructuras secundarias de los ARN pueden ayudar a descubrir las funciones de los ARN no codificantes funcionales [66]. Actualmente la mayoría de los métodos de predicción de estructuras secundarias de ácidos nucleicos se basan en un modelo termodinámico del vecino más cercano [63, 64]. Un método común para determinar las estructuras más probables dada una secuencia de nucleótidos utiliza un algoritmo de programación dinámica que busca estructuras con baja energía libre [65], sin embargo, estos métodos suelen prohibir los pseudonudos u otros casos en los que los pares de bases no se encuentren completamente anidados. También pueden utilizarse otros métodos, como las gramáticas estocásticas libres de contexto, para predecir la estructura secundaria de los ácidos nucleicos [60].

La mayoría de las moléculas funcionales de ARN tienen estructuras secundarias específicas conservadas en la evolución. Por lo tanto, calcular eficientemente la estructura consenso de una colección de estas moléculas de ARN resulta útil [60]. Dada una cantidad suficientemente grande de secuencias de ARN alineadas, se puede inferir directamente una estructura secundaria de consenso a partir de los datos, la idea es que las sustituciones en la secuencia respeten las restricciones estructurales comunes, de esta forma, las sustituciones en las regiones helicoidales tienen que estar correlacionadas, ya que en general sólo 6 (GC, CG, AU, UA, UG y GU) de las 16 combinaciones de dos bases pueden incorporarse a la hélice [60].

Las estructuras secundarias pueden descomponerse de forma única en bucles o loops. De manera formal, se llama a una posición k inmediatamente interior del par (i, j) si $i < k < j$ y no existe ningún otro par de bases (p, q) tal que $i < p < k < q < j$. Un bucle consiste

entonces en el par de cierre (i, j) y todas las posiciones inmediatamente interiores de (i, j) . Como caso especial, el bucle exterior contiene todas las posiciones no interiores a ningún par [60]. Un bucle se caracteriza por su longitud y grado. La longitud hace referencia a el número de nucleótidos no apareados en el bucle, y el grado está dado por el número de pares de bases que delimitan el bucle (incluido el par de cierre). Los bucles de grado 1 se denominan bucles de horquilla o hairpin loop, los bucles interiores tienen grado 2, y los bucles con grado > 2 se denominan bucles múltiples o multiloops (ver Tabla 1). Las protuberancias o bulge loops son casos especiales de bucles interiores en los que sólo hay bases no apareadas en un lado, mientras que los stacks corresponden a un bucle interior de tamaño cero. La descomposición en bucles constituye la base del modelo de energía estándar para las estructuras secundarias del ARN, en el que se supone que la energía libre total de una estructura es una suma de las energías de los bucles que la componen [60]. Dado que ahora la contribución energética de un par de bases en una hélice depende del par precedente y del siguiente, el modelo suele denominarse modelo termodinámico del vecino más próximo [60, 63, 64].

En los casos donde se sabe *a priori* que las secuencias alineadas deben plegarse en una estructura secundaria común, `RNAalifold` [71] resuelve este problema de forma bastante sencilla, ya que considera todo el alineamiento como una única secuencia y resuelve el problema de la estructura secundaria para esta secuencia generalizada, tomando el promedio de la contribución de energía sobre todas las secuencias [60]. `RNAalifold` funciona casi igual que `RNAfold` [18, 78], con la diferencia principal de que el modelo de energía se complementa con información de covarianza [60].

`RNAfold` calcula las estructuras secundarias de MFE, también conocidas como estructuras más estables u óptimas, se refieren a las estructuras secundarias de ARN que tienen la energía libre más baja entre todas las estructuras posibles para una secuencia de ARN dada. El concepto de estructuras MFE es especialmente relevante en el contexto del plegamiento del ARN y la predicción de las propiedades estructurales de las moléculas de ARN [79]. Las estructuras MFE se predicen mediante algoritmos y métodos computacionales basados en los principios de la termodinámica. Estos métodos calculan la energía libre asociada a diferentes estructuras secundarias de ARN e identifican la estructura con la mínima energía libre. Además de calcular las MFE, `RNAfold` también calcula la función de partición de los ARN [87]. Si se da la opción `-p`, también se calcula la función de partición (1) y la matriz de probabilidad de emparejamiento de bases, e imprime la energía libre del conjunto termodinámico, la frecuencia de la estructura MFE en el conjunto (2) y la diversidad del conjunto (3) [18].

$$Q = \sum_{s \in \Omega} \exp\left(\frac{-E(s)}{RT}\right) \quad (1)$$

A partir de la función de partición sobre el conjunto de todas las estructuras posibles Ω , con temperatura T y constante de gas R , `RNAfold` calcula la energía libre del conjunto $E = -RT \cdot \ln(Q)$, y la frecuencia de la estructura MFE o s_{mfe} dentro del conjunto:

$$p = \frac{\exp\left(\frac{-E(s_{mfe})}{RT}\right)}{Q} \quad (2)$$

Además, por defecto, la opción `-p` también activa el cálculo de las probabilidades de

emparejamiento de bases p_{ij} . Y a partir de estos datos, **RNAfold** determina la diversidad del conjunto, es decir, la distancia esperada entre dos estructuras secundarias cualesquiera, así como la estructura centroide o s_c con la menor distancia ponderada de Boltzmann⁴ (4) a todas las demás estructuras $s \in \Omega$ [87].

$$\langle d \rangle = \sum_{ij} p_{ij} \cdot (1 - p_{ij}) \quad (3)$$

$$d_{\Omega}(s_c) = \sum_{s \in \Omega} p(s) d(s_c, s) \quad (4)$$

En el caso del modelo de energía, las energías para los diferentes tipos de bucles se promedian individualmente, de tal forma que tanto la *puntuación de la información mutua* (o mutual information) (5) como la *covarianza* (6)⁵, asignan una bonificación a la mutación compensatoria.

$$MI_{ij} = \sum_{X,Y} f_{ij}(XY) \log \frac{f_{ij}(XY)}{f_i(X)f_j(Y)} \quad (5)$$

$$C_{ij} = \sum_{XY, X'Y'} f_{ij}(XY) \mathbf{D}_{XY, X'Y'} f_{ij}(X'Y') \quad (6)$$

Sin embargo, ninguna de las dos puntuaciones se ocupa de las secuencias inconsistentes, es decir, de las secuencias que no pueden formar un par de bases entre las posiciones i, j . Por lo tanto, para este propósito se cuenta el número de secuencias q_{ij} que no pueden formar un par de bases entre las columnas i y j . De acuerdo a [60] las combinaciones de un nucleótido y un hueco se cuentan como inconsistentes, mientras que las combinaciones de hueco y hueco (es decir, las deleciones de un par de bases) se ignoran. En el caso de alineamientos múltiples de un gran número de secuencias, está la posibilidad de que se produzcan errores de secuenciación ocasionales y errores en el alineamiento, en cuyo caso no se puede marcar un par de posiciones como no apareadas si una única secuencia es incoherente; de este modo, lo estándar es definir un valor umbral para la puntuación combinada $B_{ij} = C_{ij} - \phi_1 q_{ij}$, tal que i, j son posiciones no emparejadas si B_{ij} es demasiado pequeño.

Lo anterior permite establecer de forma básica la manera en la que se suelen predecir las estructuras secundarias. En el caso de este trabajo, se implementó tanto **RNAfold** como **RNAz** [67] con el propósito de predecir dichas estructuras secundarias. Este programa combina un enfoque comparativo, el cual asigna una puntuación a la conservación de la estructura secundaria, teniendo en cuenta que los ARN no codificantes son termodinámicamente más estables de lo esperado por azar [60, 68, 72]. Este exceso de estabilidad se mide en términos de la puntuación z :

$$z = \frac{E - \bar{E}}{\sigma}$$

donde \bar{E} y σ son la media y la desviación estándar de la distribución de secuencias, respectivamente. En lugar de tratar con secuencias individuales, **RNAz** utiliza alineamientos

⁴Ver apéndice (E) para más información sobre la distribución de Boltzmann.

⁵Ver apéndice (F) para mayor información sobre ambas puntuaciones.

múltiples de ARN potenciales de diferentes especies como entrada.

Además, la conservación estructural también se cuantifica en términos termodinámicos, para esto se define un índice de conservación estructural SCI como la relación entre la energía media de la estructura de consenso y la media de las energías de plegamiento sin restricciones de las secuencias individuales [60]. Por un lado, una alineación de secuencias idénticas tiene $SCI = 1$, mientras que secuencias que no se encuentran en absoluto relacionadas no podrán formar una estructura consenso, es decir, $SCI = 0$ [60]. Por otro lado, las secuencias que forman una secuencia consenso bien conservada en presencia de covariaciones de secuencia tendrán las mismas contribuciones de energía en el consenso y en los pliegues individuales. Además, como la energía de consenso contiene las contribuciones adicionales por covariaciones de secuencia, se obtiene para estas secuencias que $SCI > 1$.

Teniendo en cuenta este índice de conservación de la estructura y la puntuación z , **RNAz** utiliza máquinas de soporte vectorial (SVM) para determinar si un alineamiento de secuencias múltiples dado es un ARN conservado estructuralmente [60]. El resultado original de la máquina de soporte vectorial se le denomina *valor de decisión*. Este número de valor real es positivo si la predicción es ARN y negativo en caso contrario. A partir de este valor se calcula un valor llamado *RNA class probability* o *valor p* , que es 0.5 para un valor de decisión de 0. Estos resultados, en particular las secuencias consenso, resultan de utilidad ya que éstas serán las representantes que se convertirán a las diferentes notaciones de interés de estructura secundaria, para así ser utilizadas en el proceso de entrenamiento y prueba del modelo de clasificación.

2.3. Representaciones secundarias

2.3.1. Árbol ampliado

La representación de árbol ampliado tiene dos etiquetas nuevas, P para nucleótidos emparejados y U para no emparejados; en este caso, un punto se sustituye por (U) , y a cada corchete cerrado se le asigna un identificador adicional P [18]. En la figura 5 se evidencia la equivalencia entre la notación de dot-bracket y la representación de árbol ampliado.

$$\begin{aligned} & .(((\underline{((\dots))})..((\dots)))) \\ & ((U)((U)(U)((\underline{((U)(U)(U)P)P)P})(U)(U)((U)(U)P)P)P)(U)R) \end{aligned}$$

Figura 5: Ejemplo de la representación de árbol ampliado [18].

2.3.2. Representación HIT

La representación HIT o de árbol homeomórficamente irreducible (homeomorphically irreducible tree), también conocida como la representación de Fontana [16]. En esta notación, un *stack* se representa como un único par de corchetes etiquetados como P y ponderados por el número de pares de bases. De forma correspondiente, bases no emparejadas se

muestra como un par de corchetes etiquetados como (U) y ponderados por su longitud [18].

$$\begin{aligned} & \cdot((\cdot(((\underline{(\dots)}))\cdot((\cdot))))). \\ & ((U)((U)(U)((U)(U)(U)P)P)P)(U)(U)((U)(U)P)P)P(U)R \\ & ((U1)((U2)((U3)P3)(U2)((U2)P2)P2)(U1)R \end{aligned}$$

Figura 6: Ejemplo de la equivalencia con la representación HIT [18].

En la figura 6 se muestra la equivalencia entre la notación de dot-bracket, la representación de árbol ampliado y la representación HIT.

2.3.3. Representación de árbol de *grano grueso*

La representación coarse grained o de *grano grueso*, es una representación propuesta por Bruce Shapiro [17]. Esta notación no conserva toda la información de la estructura secundaria, y representa los distintos elementos de la estructura mediante paréntesis simples coincidentes y se etiquetan como sigue [18]:

- H (hairpin loop),
- I (interior loop),
- B (bulge),
- M (multi-loop), y
- S (stack)

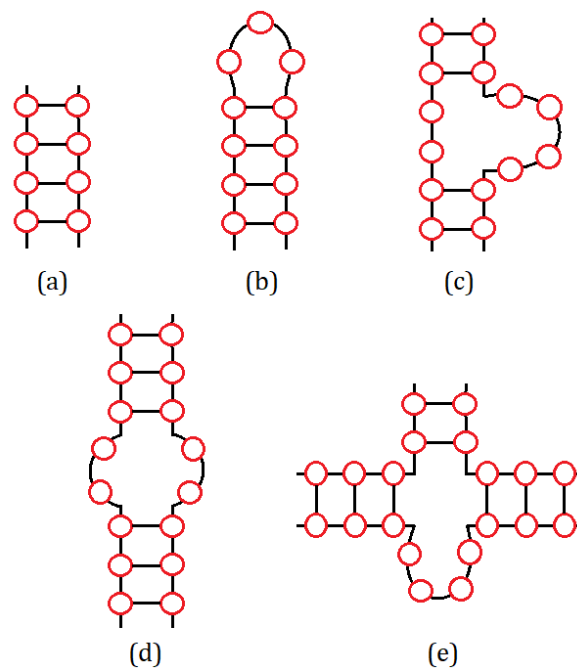


Figura 7: (a) Stack, (b) hairpin loop, (c) bulge, (d) interior loop, y (e) multi-loop.

En la figura 7 se pueden ver como se interpretan las diferentes etiquetas de la estructura secundaria, de acuerdo a la representación propuesta.

- a) $((H)((H)M)R)$
- b) $(((((H)S)((H)S)M)S)R)$
- c) $((((((H)S)((H)S)M)S)E)R)$
- d) $((((((H3)S3)((H2)S2)M4)S2)E2)R)$

Figura 8: Ejemplo de la representación de Shapiro. a) Es la representación corta (con nodo raíz R , sin nodos madre S), b) es una representación completa (con nodo raíz R), c) es una representación extendida (con nodo raíz R , con nodos externos E), y d) es una representación ponderada (con nodo raíz R , con nodos externos E) [18].

Este alfabeto tiene una letra adicional para los elementos externos E . Además, los identificadores pueden ir seguidos de un peso correspondiente al número de bases o pares de bases no apareadas en el elemento de la estructura [18]. Dentro de esta representación existen otras formas de mostrar la estructura secundaria de la secuencia (ver Fig. 8).

A todas las representaciones mencionadas previamente (excepto la forma dot-bracket) se les puede añadir la etiqueta adicional R , indicando la raíz donde finaliza la estructura de la secuencia.

Algoritmos del aprendizaje de máquina

El crecimiento exponencial en la cantidad de datos biológicos disponibles plantea dos problemas: por un lado, el almacenamiento y la gestión eficientes de la información y, por otro, la extracción de información útil a partir de estos datos. El segundo problema es uno de los principales retos en la bioinformática y biología computacional, que requiere el desarrollo de herramientas y métodos capaces de transformar todos estos datos en conocimiento biológico interpretable [28, 29]. Para el caso del estudio de información de virus el panorama de la búsqueda de nuevos métodos es también demandante dada masiva acumulación de información viral de virus como el del HIV, la influenza y los de coronavirus en los últimos años. Los estudios computacionales relacionados con la estructura, función y evolución de los virus siguen siendo un área de investigación desatendida [2], y algunos de los retos que se evidencian en la literatura suelen estar relacionados con la extracción de características de las secuencias virales [23]; precisamente por la variabilidad de estos virus, resulta complejo lograr hacer una recopilación y análisis de características que no sean redundantes y que además brinden información útil para la clasificación. Esta es un área que se sigue estudiando ya que se presentan casos donde existen características similares entre virus o subtipos de virus distintos, lo cual dificulta hacer una distinción clara entre las clases que se desean caracterizar en modelos basados en técnicas de machine learning.

Por ende, en este capítulo se profundiza en el concepto de machine learning o aprendizaje de máquina, asimismo como en los desafíos que se deben tener en cuenta a la hora de trabajar con cualquier algoritmo de esta área. Se trata en particular el tipo de problema que se aborda con este trabajo: la clasificación, y los diferentes métodos que permiten dar soluciones a este problema. Al final se describen algunos ejemplos de aplicaciones de aprendizaje de máquina como: BERTax, CHEER y pysster que utilizan algunas de las técnicas descritas para realizar la clasificación de secuencias algunas previas al ensamblaje genómico o metagenómico y otras posterior al ensamblaje.

“A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E .”

Tom Mitchell, 1997

3.1. Machine learning

El aprendizaje de máquina o machine learning es una rama de la *ciencia computacional*⁶ que pretende que los computadores o máquinas aprendan sin ser programados directamente [30, 31]. Las técnicas basadas en esta área se han aplicado con éxito en diversos campos que van desde el reconocimiento de patrones, la visión por ordenador, la ingeniería de naves espaciales, las finanzas, el entretenimiento y la biología computacional hasta las aplicaciones biomédicas y médicas [26].

Hoy en día áreas como la bioinformática y la biología computacional se ven beneficiadas de las actuales innovaciones en inteligencia artificial y machine learning, esto se debe a que mientras la bioinformática aplica principios de la informática para ayudar a comprender los vastos, diversos y complejos datos de las ciencias de la vida y hacerlos así más útiles, las técnicas de machine learning permiten aplicar enfoques computacionales que ayuden a abordar cuestiones teóricas o experimentales [32]. El aprendizaje de máquina se puede clasificar en categorías generales, basándose en los siguientes criterios [81]:

- Si se entrenan o no con supervisión humana, es decir, que se entrenan utilizando un conjunto de datos que han sido etiquetados por humanos (supervisados, no supervisados, semi-supervisados y de aprendizaje por refuerzo).
- Si pueden o no aprender de forma incremental sobre la marcha (aprendizaje en línea o por lotes⁷).
- Si funcionan simplemente comparando nuevos puntos de datos con puntos de datos conocidos, o detectando patrones en los datos de entrenamiento y construyendo un modelo predictivo (aprendizaje basado en instancias frente a aprendizaje basado en modelos⁸).

3.1.1. Primera categoría: aprendizaje supervisado, no supervisados, semi-supervisado y por refuerzo

Aprendizaje supervisado: En el aprendizaje supervisado el valor del resultado, es decir, la variable dependiente, a menudo denominada *etiqueta* (o label), se conoce para cada observación, y los datos con valores de resultado especificados se denominan *datos etiquetados* [31]. En este tipo de aprendizaje una tarea típica es la clasificación, y algunos de los algoritmos de aprendizaje supervisado más importantes son: *k*-vecinos más cercanos, regresión lineal y logística, máquinas de soporte vectorial (SVM), árboles de decisión y bosques aleatorios, y redes neuronales [31, 81].

Aprendizaje no supervisado: En el aprendizaje no supervisado, el algoritmo intenta identificar relaciones y agrupaciones naturales dentro de los datos sin referencia a ningún resultado o a la *respuesta correcta*. Los enfoques de aprendizaje no supervisado comparten similitudes en cuanto a objetivos y estructura con los enfoques estadísticos que intentan identificar subgrupos no especificados con características similares [31]. Los algoritmos de agrupación (clustering), que agrupan observaciones basándose en características similares de los datos, son implementaciones comunes del aprendizaje no supervisado. Algunos ejemplos son la agrupación por

⁶En inglés *computer science*.

⁷En inglés es online versus batch learning

⁸En inglés es instance-based versus model-based learning

k -means, SVM de una clase, análisis de componentes principales (PCA), entre otros [31, 33, 34, 81].

Aprendizaje semi-supervisado: Etiquetar datos suele llevar mucho tiempo y tiende a ser costoso, además, se presentan casos donde hay mucho sin etiquetar y poco que sí se encuentra etiquetado. En estos casos, hay algunos algoritmos que pueden trabajar con datos parcialmente etiquetados. A esto se denomina aprendizaje semi-supervisado, y la mayoría de los algoritmos de este tipo de aprendizaje son combinaciones de algoritmos supervisados y no supervisados [81].

Aprendizaje por refuerzo: En el aprendizaje por refuerzo el sistema de aprendizaje llamado *agente*, puede observar el entorno, seleccionar y realizar acciones, y obtener recompensas a cambio o *penalizaciones* en forma de recompensas negativas. Con esta información, debe aprender por sí mismo cuál es la mejor estrategia, denominada política o *policy*, para obtener una mayor recompensa a lo largo del tiempo. Una política ayuda a definir qué acción debe elegir el agente cuando se encuentra en una situación determinada [81].

3.1.2. Segunda categoría: aprendizaje en línea o por lotes

Aprendizaje en línea: En el aprendizaje en línea, el sistema se entrena de forma incremental alimentándolo con instancias de datos de forma secuencial, ya sea individualmente o en pequeños grupos denominados *mini-lotes* o *mini-batches*. Cada paso de aprendizaje es rápido y barato, de modo que el sistema puede aprender sobre la marcha, a medida que llegan los nuevos datos. También es una buena opción si se dispone de recursos informáticos limitados, ya que una vez el sistema ha aprendido sobre nuevas instancias de datos, ya no las necesita, por lo que puede descartarlas [81].

Aprendizaje por lotes: En el aprendizaje por lotes, el sistema es incapaz de aprender de forma incremental: debe entrenarse utilizando todos los datos disponibles. En este el sistema se entrena y luego funciona sin aprender más, ya que sólo aplica lo que ha aprendido [81].

3.1.3. Tercera categoría: aprendizaje basado en instancias o basado en modelos

Aprendizaje basado en instancias: En este aprendizaje el sistema aprende los ejemplos de memoria y luego generaliza a nuevos casos utilizando una medida de similitud para compararlos con los ejemplos aprendidos (o un subconjunto de ellos) [81].

Aprendizaje basado en modelos: Con este aprendizaje el sistema aprende otra forma de generalizar a partir de un conjunto de ejemplos, para esto se contruye un modelo de esos ejemplos y se utiliza para hacer predicciones [81].

3.2. Problema de clasificación

Se mencionó que dentro de los problemas de aprendizaje supervisado están aquellos asociados a la clasificación. Recordemos que el objetivo del aprendizaje supervisado es construir

un modelo conciso de la distribución de las etiquetas en términos de características predictoras [82, 83], es decir, poder asignar datos a categorías específicas. Por ello, la idea detrás de los problemas de clasificación es poder reconocer, comprender y agrupar objetos en categorías pre-establecidas, de tal manera que se logre de reconocer patrones, para así encontrar el mismo patrón en futuros conjuntos de datos.

En el mundo real, no siempre se tienen atributos o características que contribuyan a la determinación de las etiquetas. Por ello es necesario tener en cuenta que: 1) un exceso en la cantidad de características ralentizan el proceso de aprendizaje y hacen que el clasificador se ajuste en exceso a los datos de entrenamiento y comprometa la generalización del modelo [84], y 2) características irrelevantes no aportan al aprendizaje. Esto resulta esencial ya que en el caso de este trabajo, precisamente el interés radica en evaluar la relevancia de las estructuras secundarias de árbol ampliado, HIT y árbol de grano grueso, para estudiar su posible potencial en el proceso de selección de características en la clasificación de virus de ARN.

3.2.1. Desafíos en el aprendizaje de máquina

1. **Cantidad insuficiente de datos de entrenamiento:** La mayoría de los algoritmos requieren de muchos datos para un correcto funcionamiento, y entre más complejo sea el problema, mayor cantidad de datos y ejemplos se necesitan para obtener resultados óptimos.
2. **Desbalance en los datos:** La clasificación desequilibrada es un problema común en el aprendizaje de máquina. Esto ocurre cuando el conjunto de datos de entrenamiento tiene una distribución desigual de las clases, lo que conduce a un sesgo potencial en el modelo entrenado [81].
3. **Datos de entrenamiento no representativos:** Para generalizar bien, es fundamental que los datos de entrenamiento sean representativos de los nuevos casos a los que se quiere generalizar. Se debe tener en cuenta algunos errores que se pueden dar: que la muestra sea demasiado pequeña, lo cual genera ruido de muestreo, es decir, datos no representativos, o se puede dar que la muestra sea muy grande pero no es representativa, lo cual indica que el método de muestreo es defectuoso [81].
4. **Datos de baja calidad:** Este tipo de inconvenientes se presentan cuando los datos de entrenamiento están llenos de errores, valores atípicos y ruido, lo cual resulta en una mayor dificultad a la hora de detectar los patrones subyacentes [81].
5. **Características irrelevantes:** Debido a que el modelo sólo será capaz de aprender si los datos de entrenamiento contienen suficientes características relevantes y no demasiadas irrelevantes, es indispensable encontrar un buen conjunto de características con las que entrenar. Para esto, se deben tener en cuenta los siguientes pasos: (i) selección de características útiles, (ii) extracción de características para combinar las características existentes y para obtener una más útil, y (iii) crear nuevas características mediante la recopilación de nuevos datos [81].
6. **Sobreajuste de los datos de entrenamiento:** Sobreajustar los datos significa que el modelo funciona bien con los datos de entrenamiento, pero no generaliza bien. Esto ocurre cuando el modelo es demasiado complejo en relación con la cantidad y ruido de los datos de entrenamiento [81].

7. **Subajuste de los datos de entrenamiento:** El subajuste de datos es lo opuesto al sobreajuste, ocurre cuando el modelo es demasiado simple para aprender la estructura subyacente de los datos [81].

3.2.2. Métricas de desempeño

La evaluación de un modelo es importante para poder determinar la eficacia del mismo. Para esto, hay diferentes métricas que ayudan a estimar cuán bueno es un resultado. A continuación se presentan algunas de estas métricas.

1. **Matriz de confusión:** Una forma de evaluar el rendimiento de un clasificador es observar la matriz de confusión. La idea es contar el número de veces que instancias de la clase A se clasifican como clase B [81]. Esta matriz se utiliza para problemas de clasificación en donde la salida puede ser de dos o más tipos de clases. La matriz de confusión (ver Fig. 9), es una tabla con dos dimensiones: la real y la predicha. Cada fila de una matriz de confusión representa una clase real, mientras que cada columna representa una clase predicha⁹.

		Valor predicho	
		Positivo	Negativo
Valor real	Positivo	Verdadero positivo (VP)	Falso negativo (FN)
	Negativo	Falso positivo (FP)	Verdadero negativo (VN)

Figura 9: Esta matriz se utiliza para evaluar la exactitud o *precision* de un clasificador, y permite ilustrar los valores VP , VN , FP y FN (para la definición de dichos valores, revisar el apéndice (B)).

2. **Accuracy o exactitud:** La exactitud en problemas de clasificación es el número de predicciones correctas realizadas por el modelo sobre todos los tipos de predicciones realizadas [81]. Se calcula como la suma de las predicciones correctas dividida por el número total de datos (7), donde la mejor precisión es 1 y la peor 0 [86].

$$ACC = \frac{VP + VN}{VP + VN + FP + FN} \quad (7)$$

3. **Precisión:** Esta métrica es la relación entre los verdaderos positivos y todos los positivos predichos por el modelo. Se dice que la precisión del modelo es baja a medida que se presenta una mayor cantidad de falsos positivos [81, 86]. La precisión

⁹También se puede representar la matriz de forma traspuesta, es decir, que en las filas se encuentren las clases predichas, y en las columnas, las clases reales.

se calcula como el número de predicciones positivas correctas, dividido por el número total de predicciones positivas (8). La mejor precisión es 1 y la peor 0 [86].

$$\text{Precisión} = \frac{VP}{VP + FP} \quad (8)$$

4. **Sensibilidad o *recall***: También se le conoce como la tasa de verdaderos positivos. Este valor es la proporción de casos positivos detectados correctamente por el clasificador [81], y se calcula como el número de predicciones positivas precisas (VP) dividido por el número total de valores positivos (9). La mejor tasa es 1 y la peor 0 [86].

$$\text{Sensibilidad} = \frac{VP}{VP + FN} \quad (9)$$

5. **F-score**: La puntuación F es una métrica que representa la relación entre la sensibilidad y la precisión, y se obtiene considerando la media armónica entre las dos medidas (10) [86].

$$\text{F-score} = \frac{2 \cdot \text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \quad (10)$$

3.3. Algoritmos de clasificación

Se mencionó previamente que en el aprendizaje supervisado una tarea usual es la de clasificación, y algunos de los algoritmos más importantes son: k -vecinos más cercanos, regresión logística, máquinas de soporte vectorial (SVM), árboles de decisión y bosques aleatorios, y redes neuronales [31, 81]. A continuación se elabora un poco más sobre algunos de estos algoritmos.

- **K -vecinos más cercanos**: Este algoritmo es un método no paramétrico que se utiliza para problemas de clasificación y regresión, y hace uso de la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual. Para esto, se estima el valor de la función de densidad de probabilidad $F(x/C_j)$ luego de que el elemento x pertenezca a la clase C_j . La etiqueta de la clase a la que pertenece un objetivo se asigna por mayoría, es decir, se utiliza la etiqueta que está representada con más frecuencia en un punto de datos determinado [89].

Los ejemplos de entrenamiento son vectores en un espacio de características multidimensional, cada uno con una etiqueta de clase. La fase de entrenamiento del algoritmo consiste únicamente en almacenar los vectores de características y las etiquetas de clase de las muestras de entrenamiento. En la fase de clasificación, k es una constante definida, y un vector no etiquetado, es decir, un punto de prueba, se clasifica asignándole la etiqueta más frecuente entre las k muestras de entrenamiento más cercanas a ese punto de consulta [90] (ver Fig. 10). La mejor elección para el valor de k depende de los datos. Generalmente, si k corresponde a valores altos se reduce el ruido en la clasificación, pero hace que los límites entre las clases sean menos claros [91].

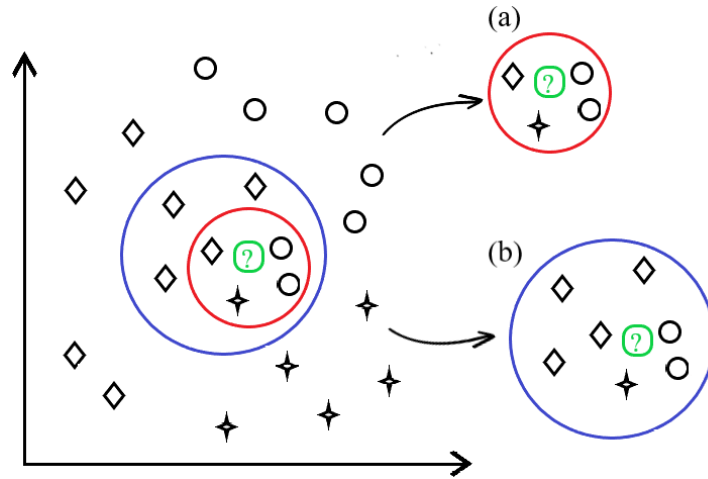


Figura 10: Se desea clasificar el círculo verde. (a) Para $k = 4$ este es clasificado con la clase círculo, ya que hay solo un rombo, una estrella y dos círculos, dentro del círculo rojo. (b) Para $k = 7$ este es clasificado con la clase rombo, ya que hay dos círculos, una estrella y cuatro rombos, dentro del círculo azul.

- Regresión logística:** La regresión logística se utiliza en muchos casos para estimar la probabilidad de que una instancia pertenezca a una clase determinada. Si la probabilidad estimada es superior al 50 %, el modelo predice que la instancia pertenece a esa clase, denominada clase positiva, etiquetada como 1 y, en caso contrario, predice que no, es decir, que pertenece a la clase negativa, etiquetada como 0 [81]. Este modelo calcula una suma ponderada de las características de entrada más un término de sesgo, pero en lugar de mostrar el resultado directamente muestra la *logística* de este resultado [81] (ver (11)).

$$\hat{p} = h_{\theta}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta}) \quad (11)$$

donde \mathbf{x} es el vector de características de instancias, que contiene x_0 hasta x_n , y $\boldsymbol{\theta}$ es el vector de parámetros del modelo, que contiene el término de sesgo θ_0 y los pesos de las características θ_1 hasta θ_n . La *logística* (ver (12)) es una función sigmoidea (ver Fig. 11) que arroja un número entre 0 y 1 [81].

$$\sigma(t) = \frac{1}{1 + e^{-t}} \quad (12)$$

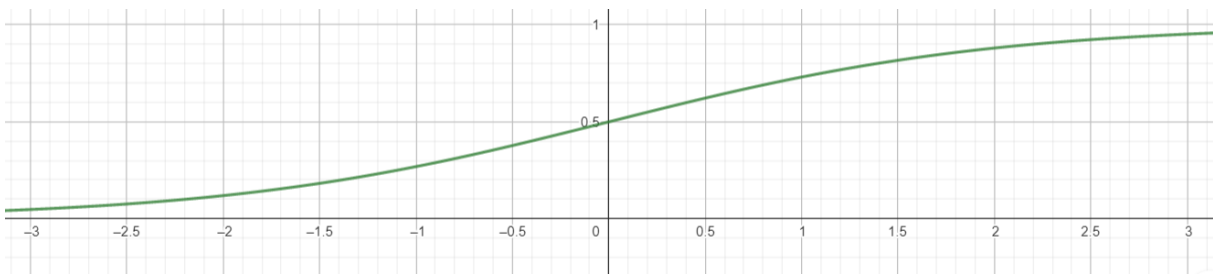


Figura 11: Gráfica de la función $\sigma(t)$ con la herramienta GeoGebra.

Una vez que el modelo ha estimado la probabilidad $\hat{p} = h_{\theta}(x)$ de que una instancia \mathbf{x} pertenezca a la clase positiva, puede realizar su predicción \hat{y} de la siguiente forma: el modelo predice $\hat{y} = 1$ si $\mathbf{x}^T \boldsymbol{\theta}$ es positivo, y esto ocurre cuando $\sigma(t) \geq 0.5$ para $t \geq 0$, y el modelo predice $\hat{y} = 0$ si $\mathbf{x}^T \boldsymbol{\theta}$ es negativo, es decir, cuando $\sigma(t) < 0.5$ para $t < 0$.

- Máquinas de soporte vectorial (SVM):** Las máquinas de soporte vectorial (o Support Vector Machines) son un modelo capaz de realizar clasificaciones lineales o no lineales, cuyo objetivo es crear una frontera de decisión (también denominado hiperplano) que pueda separar el espacio n -dimensional en clases, de modo que se pueda ubicar de forma sencilla los nuevos puntos de datos en la categoría correcta. El algoritmo elige los puntos o vectores que ayudan a crear el hiperplano, dichos puntos se llaman vectores de soporte. Hay muchos hiperplanos, sin embargo, se suele escoger el que consigue la mayor separación entre las clases [81, 92].

La clasificación lineal se utiliza para datos linealmente separables, lo que significa que si un conjunto de datos puede clasificarse en dos clases utilizando una única línea recta, entonces dichos datos se denominan datos linealmente separables, y el clasificador utilizado se denomina clasificador lineal (ver Fig. 12). Para este caso suponga que se da un conjunto de datos de entrenamiento de n puntos de la forma: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$, donde y_i es 1 o -1, cada uno de estos indicando la clase a la cual pertenece el punto \mathbf{x}_i , y cada uno de estos es un vector real p -dimensional. Como se quiere encontrar el hiperplano¹⁰ de margen máximo que divide el grupo de puntos en aquellos para los cuales $y_i = 1$ y $y_i = -1$, se define una distancia entre el hiperplano y el punto \mathbf{x}_i más cercano de cada grupo de tal forma que dicha distancia sea la máxima.

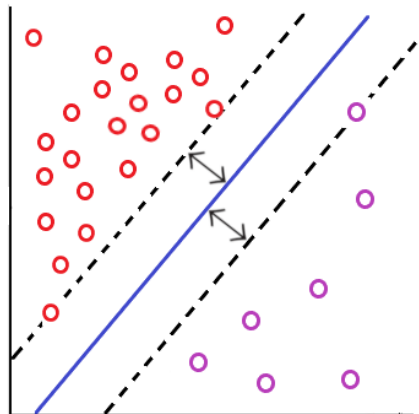


Figura 12: La línea azul corresponde al hiperplano de margen máximo y las líneas punteadas a los márgenes para una SVM entrenada con muestras de dos clases.

Para datos que no son linealmente separables, se propuso una forma de crear clasificadores no lineales aplicando el *truco del kernel*¹¹ a los hiperplanos de margen

¹⁰Cualquier hiperplano se puede escribir como el conjunto de puntos \mathbf{x} que satisface: $\mathbf{w}^T \mathbf{x} - b = 0$, donde \mathbf{w} es el vector normal al hiperplano.

¹¹Ver apéndice C.

máximo [94]. En este algoritmo cada producto punto se sustituye por una función kernel no lineal, lo cual permite ajustar el hiperplano de margen máximo a un espacio de características transformado.

- Árboles de decisión:** Un árbol de decisión es un algoritmo no paramétrico que tiene una estructura jerárquica de árbol, que consta de un nodo raíz, ramas, nodos internos y nodos hoja (ver Fig. 13). El nodo raíz o nodo de decisión, representa una elección que dará lugar a la subdivisión de todos los registros en dos o más subconjuntos mutuamente excluyentes. Los nodos internos, representan una de las posibles elecciones disponibles en ese punto de la estructura del árbol, y los nodos hoja, representan el resultado final de una combinación de decisiones [95]. Este algoritmo emplea una estrategia de división hasta identificar los puntos de *corte* óptimos dentro del árbol; el proceso de división se repite de forma descendente y recursiva hasta que todos o la mayoría de los registros se han clasificado con etiquetas de clases específicas.

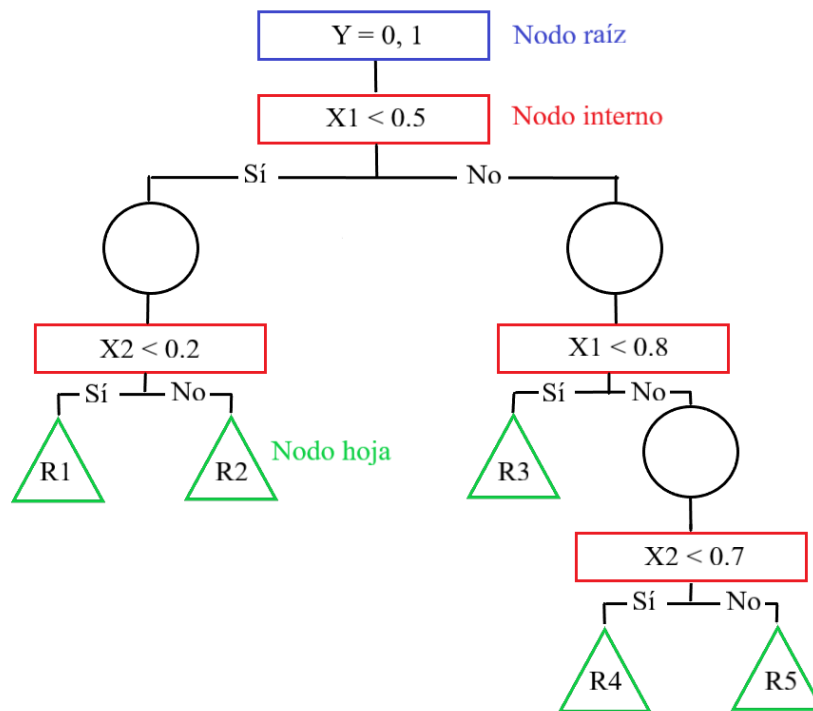


Figura 13: Estructura de un árbol de decisión, que incluye una única variable objetivo binaria Y (0 ó 1) y dos variables continuas, X_1 y X_2 . La imagen fue adaptada de [95].

Se puede encontrar en los árboles de decisión el atributo **gini** de un nodo (ver (13)). Este mide su impureza, es decir, un nodo es puro o **gini** = 0 si todas las instancias de entrenamiento a las que se aplica pertenecen a la misma clase [81].

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2 \quad (13)$$

donde $p_{i,k}$ es la proporción de instancias de la clase k entre las instancias de entrenamiento en el i -ésimo nodo. Además, también se puede estimar la probabilidad de

que una instancia pertenezca a una clase concreta k . Eso se consigue recorriendo el árbol para encontrar el nodo hoja de la instancia de interés, con lo cual se devuelve la proporción de instancias de entrenamiento de la clase k en dicho nodo [81].

- Bosques aleatorios:** Los bosques aleatorios o de decisión aleatorios son un método que funciona construyendo una multitud de árboles de decisión en el momento del entrenamiento [81, 96]. Para tareas de clasificación, la salida del bosque aleatorio es la clase seleccionada por la mayoría de los árboles (ver Fig. 14). La idea en los bosques aleatorios es mejorar la reducción de la varianza disminuyendo la correlación entre los árboles, sin aumentar demasiado la varianza. Esto se consigue en el proceso de crecimiento del árbol mediante la selección aleatoria de las variables de entrada [96]. Esta técnica de agregar las predicciones de un grupo de predictores (como clasificadores) se denomina aprendizaje conjunto y un algoritmo de aprendizaje conjunto se denomina método conjunto [81]. Este algoritmo se puede aplicar entrenando un grupo de clasificadores de árbol de decisión, con distintos subconjuntos aleatorios del conjunto de entrenamiento, para así obtener las predicciones de todos los árboles individuales y, de esta forma, predecir la clase que obtiene la mayor cantidad de votos [81].

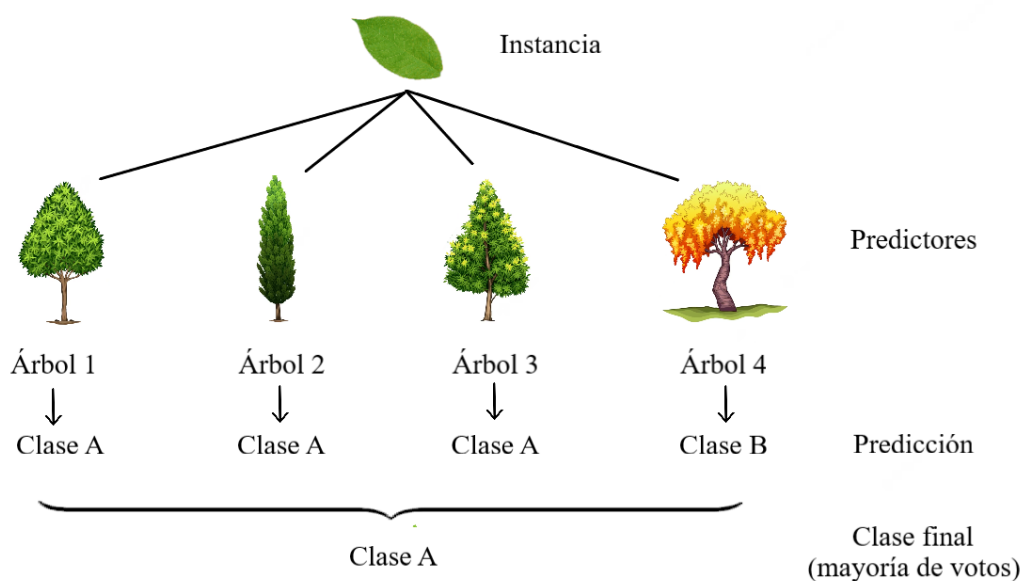


Figura 14: En esta imagen se muestra una representación de un bosque aleatorio, donde cada árbol representa un predictor, de por ejemplo regresión logística, SVM, entre otros; y al agregar las predicciones de cada clasificador se obtiene la clase predicha por mayoría de votos.

Una forma de obtener un conjunto diverso de clasificadores es utilizar algoritmos de entrenamiento muy diferentes. Otro método consiste en utilizar el mismo algoritmo de entrenamiento para cada predictor y entrenarlos en diferentes subconjuntos aleatorios del conjunto de entrenamiento. Cuando el muestreo se realiza con sustitución, este método se denomina *bagging* (abreviatura de bootstrap aggregating). Cuando el muestreo se realiza sin reemplazo, se denomina *pasting*. En otras palabras, tanto el bagging como el pasting permiten muestrear instancias de entrenamiento varias

veces para múltiples predictores, pero sólo el bagging permite muestrear instancias de entrenamiento varias veces para el mismo predictor [81].

- Redes neuronales:** El desarrollo de modelos basados en redes neuronales artificiales o redes neuronales está inspirado en las redes neuronales biológicas de nuestro cerebro [81]. El cerebro es un sistema de procesamiento de información altamente complejo, no lineal y paralelo, que tiene la capacidad de organizar sus componentes estructurales, conocidos como neuronas, para realizar ciertos cálculos, por ejemplo: reconocimiento de patrones, percepción y control motor. Si se considera la visión humana, que es una tarea de procesamiento de información, la función del sistema visual es proporcionar una representación del entorno que nos rodea y, aún más importante, suministrar información necesaria para interactuar con el entorno [97].

En [97] se define una red neuronal¹² como un procesador distribuido, *masivamente* paralelo, formado por unidades de procesamiento simples que tiene una propensión natural a almacenar el conocimiento experiencial y ponerlo a disposición para su uso. Estas redes se parecen al cerebro en dos aspectos:

1. La red adquiere los conocimientos de su entorno mediante un proceso de aprendizaje.
2. La intensidad de las conexiones entre neuronas, denominadas pesos sinápticos, se utiliza para almacenar los conocimientos adquiridos.

Una de las arquitecturas de redes neuronales más sencillas es el perceptrón, inventada en 1957 por Frank Rosenblatt [81]. El perceptrón está basado en una neurona artificial llamada *threshold logic unit* o TLU, en esta las entradas y salidas son números (en lugar de valores binarios on/off), y cada conexión de entrada está asociada a un peso. Esta red calcula una suma ponderada de sus entradas $z = w_1x_1 + w_2x_2 + \dots + w_nx_n = \mathbf{x}^T\mathbf{w}$, luego aplica una *step function* o función escalonada¹³ a esa suma y arroja el resultado: $h_{\mathbf{w}}(\mathbf{x}) = \text{step}(z)$, donde $z = \mathbf{x}^T\mathbf{w}$ (ver Fig. 15).

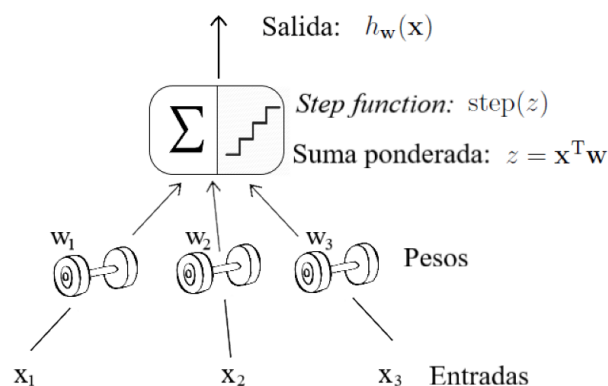


Figura 15: Neurona que calcula una suma ponderada de sus entradas y, aplica una función escalonada.

¹²Salvo que se indique lo contrario al hablar de redes neuronales, se hace referencia a redes neuronales computacionales, no biológicas.

¹³Ver apéndice D.

La forma en la que se estructuran las neuronas de una red neuronal está ligada al algoritmo de aprendizaje utilizado para entrenar la red [97]. El algoritmo de aprendizaje tiene como función modificar los pesos sinápticos de la red de manera ordenada para alcanzar un objetivo específico, esto lo consigue reforzando las conexiones que ayudan a reducir el error, es decir, se le da como entrada una instancia a la vez, y por cada instancia hace una predicción, y por cada salida para la que produzca una predicción incorrecta, se refuerza los pesos de la conexión de las entradas que hubiesen contribuido a una predicción correcta [61]. En la ecuación (14) se establece la regla de aprendizaje:

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta (y_i - \hat{y}_j) x_i \quad (14)$$

donde $w_{i,j}$ es el peso de la conexión entre la neurona de entrada i -ésima y la neurona de salida j -ésima, x_i es el i -ésimo valor de entrada de la instancia de entrenamiento actual, \hat{y}_j es la salida de la j -ésima neurona de salida de la instancia de entrenamiento actual, y_j es la salida objetivo de la neurona de salida j -ésima para la instancia de entrenamiento actual y η es la tasa de aprendizaje.

Por ende, un perceptrón se compone simplemente de una sola capa de TLUs, con cada TLU conectada a todas las entradas. Cuando todas las neuronas de una capa están conectadas a todas las neuronas de la capa anterior, es decir, a sus neuronas de entrada, la capa se denomina capa totalmente conectada o capa densa. Cuando un perceptrón tiene más de una entrada, se utiliza la ecuación (15) para computar las salidas de varias instancias al mismo tiempo.

$$h_{\mathbf{W},\mathbf{b}} = \phi(\mathbf{X}\mathbf{W} + \mathbf{b}) \quad (15)$$

\mathbf{X} representa la matriz de características de entrada, donde hay una fila por cada instancia y una columna por característica. La matriz de pesos \mathbf{W} contiene todos los pesos de conexión excepto los de la neurona de sesgo. El vector de sesgo \mathbf{b} contiene todos los pesos de conexión entre la neurona de sesgo y las neuronas artificiales. La función ϕ se denomina *función de activación*; cuando las neuronas artificiales son de tipo TLU, esta función de activación es una función escalonada.

Otra arquitectura conocida son las redes neuronales multicapa o perceptrón multicapa. Estas se componen de una capa de entrada, una o más capas de TLUs, llamadas capas ocultas, y una capa final de TLUs, llamada capa de salida (ver Fig. 16). Las capas cercanas a la capa de entrada suelen denominarse capas inferiores, y las cercanas a las salidas suelen llamarse capas superiores. Cada capa, excepto la capa de salida incluye una neurona de *sesgo*¹⁴ que está totalmente conectada a la siguiente capa (esta no se suele mostrar en la arquitectura de la red) [81]. Además, la función de las capas ocultas es intervenir entre la entrada y salida de la red. Al añadir una o varias capas ocultas, la red puede extraer más estadísticas e información de la entrada [97].

El perceptrón multicapa es un algoritmo de aprendizaje supervisado que aprende una función $f : R^m \rightarrow R^k$ mediante el entrenamiento en un conjunto de datos,

¹⁴En inglés se conocen como *bias neuron*.

donde m es el número de dimensiones de entrada y k es el número de dimensiones de salida. Dado un conjunto de características $X = x_1, x_2, \dots, x_m$ y un objetivo y , puede aprender un aproximador de función no lineal para la tarea de clasificación [99]. Modelos que hacen uso de este tipo de algoritmos utilizan un conjunto de entradas emparejadas y salidas deseadas, de tal forma que la tarea consiste en producir la salida deseada para cada entrada. Los valores de los hiperparámetros como la tasa de aprendizaje, el número de capas ocultas y el tamaño del lote o *batch size* se obtienen mediante aprendizaje [98].

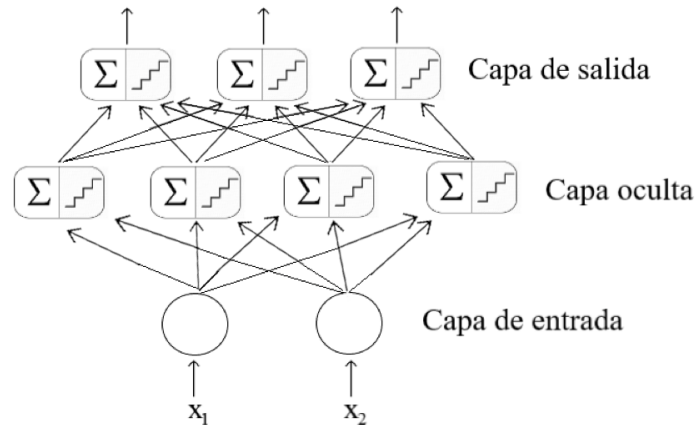


Figura 16: Arquitectura de un perceptrón multicapa con dos entradas, una capa oculta de cuatro neuronas y tres neuronas de salida.

La mayoría de las aplicaciones actuales de machine learning se basan en el aprendizaje supervisado, que hacen uso de los algoritmos descritos previamente, sin embargo, hay una gran cantidad de datos que no están etiquetados [81]. Si bien este trabajo utiliza métodos de aprendizaje supervisado, es importante tener presente que existen algoritmos basados en aprendizaje no supervisado que pueden ser útiles para esta tarea. Gran parte de estos algoritmos se apoyan del concepto de *clustering* o agrupación, el cual se encarga de encontrar los grupos más adecuados dentro de un espacio de características determinado [130], es decir, agrupar es la tarea de identificar instancias similares y asignarlas a clusters, o grupos de instancias similares. Al igual que en la clasificación, cada instancia se asigna a un grupo, aunque a diferencia de la clasificación el clustering es una tarea no supervisada [81]. A continuación se mencionan dos de los algoritmos de agrupación más populares: *K*-Means y DBSCAN.

- ***K*-means:** Este algoritmo fue propuesto por Stuart Lloyd en los laboratorios Bell en 1957 como técnica para la modulación por impulsos codificados, pero no se publicó fuera de la empresa hasta 1982 [81]. El algoritmo comienza con un grupo aleatorio de centroides, considerando cada centroide como un cluster, y realiza cálculos repetitivos para ajustar la posición de los centroides, de otra forma, se inicia colocando los centroides de forma aleatoria, por ejemplo, eligiendo k instancias al azar y utilizando sus ubicaciones como centroides, para luego etiquetar las instancias, actualizar los centroides, etiquetar las instancias, actualizar los centroides, y así sucesivamente (ver Fig. 17). El algoritmo detiene la optimización de los clusters cuando los centroides son estables, es decir, no hay cambios en sus valores, o se alcanza un número

definido de iteraciones [81, 130].

Aunque se sabe que el algoritmo converge (esto es porque el proceso repetitivo de etiquetar y actualizar los centroides no oscila eternamente, debido a que la distancia *mean squared* entre las instancias y su centroide más cercano van disminuyendo con cada iteración), puede que no converja a la solución correcta, de otra manera, puede que converja a un óptimo local, y esto depende de la inicialización del centroide [81].

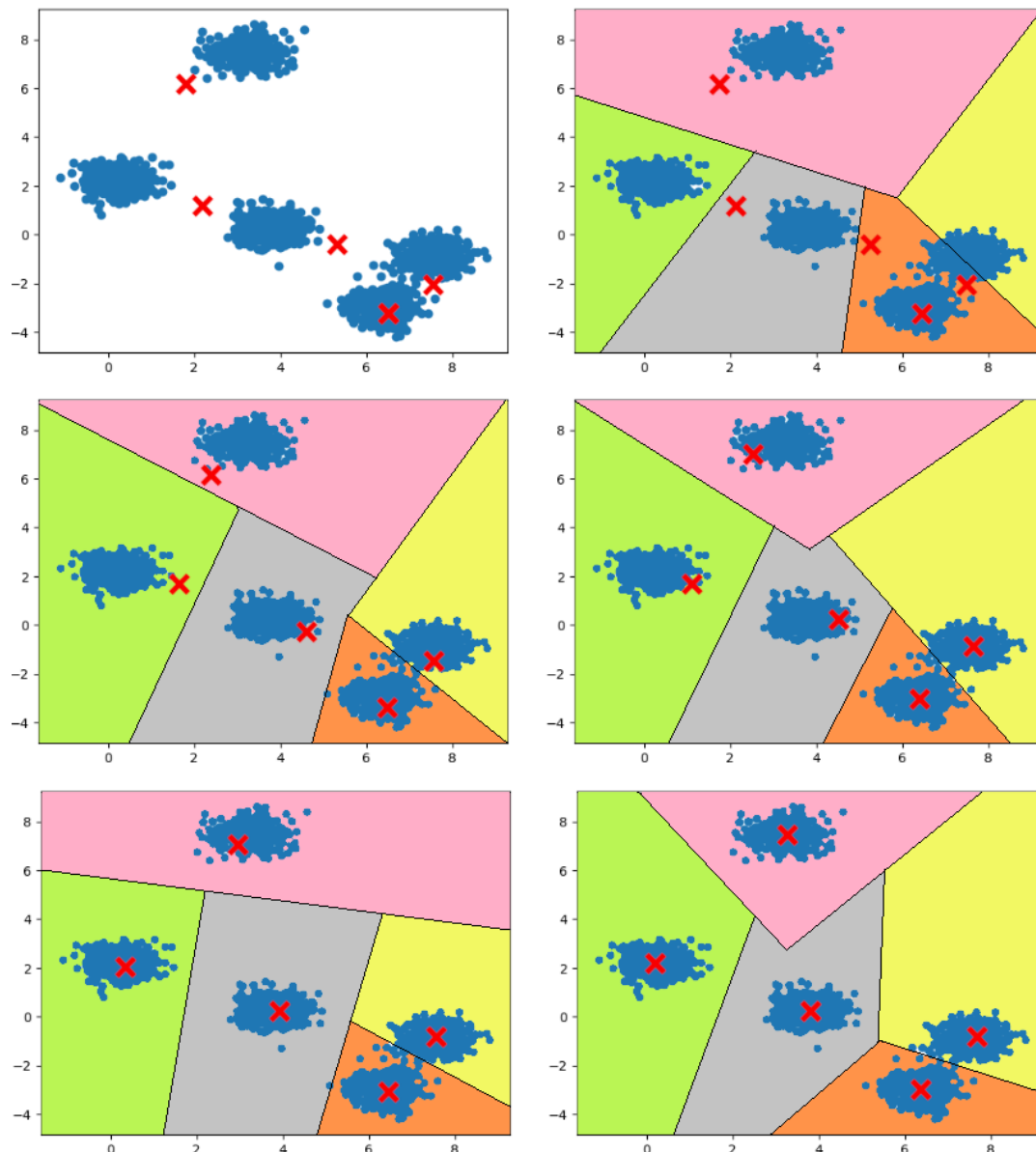


Figura 17: Algoritmo de K -means. Imagen basada en [81].

- **DBSCAN:** La agrupación espacial de aplicaciones con ruido basada en la densidad o *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) encuentra las áreas de mayor densidad en el dominio de características dado y expande esas áreas, formando clusters del espacio de características. El DBSCAN encuentra

vecindarios de un punto de datos que superan un umbral de densidad especificado. Este umbral viene definido por el número mínimo de puntos de datos necesarios dentro de un radio de vecindad (`minPts`) y el radio de la vecindad (`eps`). Ambos parámetros se inicializan manualmente al inicio del algoritmo [130].

Este algoritmo define los clusters como regiones continuas de alta densidad. La forma en la que lo hace es como sigue [81]:

- Para cada instancia, el algoritmo cuenta cuántas instancias se encuentran a una distancia ϵ . Esta región se denomina vecindario ϵ de la instancia.
- Si una instancia tiene al menos un mínimo de instancias (`min_muestras`) en su vecindad ϵ , incluida ella misma, se considera una instancia central. En otras palabras, las instancias centrales son aquellas que se encuentran en regiones densas.
- Todas las instancias de la vecindad de una instancia central pertenecen al mismo cluster. Esta vecindad puede incluir otras instancias centrales; por lo tanto, una larga secuencia de instancias centrales vecinas forma un único cluster.
- Cualquier instancia que no sea central y no tenga ninguna otra instancia en su vecindario se considera una anomalía.

Este algoritmo funciona bien si todos los clusters son lo suficientemente densos y si están bien separados por regiones de baja densidad [81].

3.4. Ejemplos de implementaciones

Teniendo ya las bases asociadas a los algoritmos o técnicas de machine learning, se hablará sobre tres herramientas que hacen uso de algunos de estos para llevar a cabo la clasificación de secuencias.

CHEER: Es una herramienta publicada en mayo del 2021 por [132], la cual realiza una clasificación taxonómica jerárquica de datos metagenómicos virales mediante aprendizaje profundo, cuyo nombre viene de *hierarCHical taxonomic classification for viral mEtagEnomic data via deep leaRning* (CHEER). Para utilizar CHEER, es necesario disponer de dos tipos de datos: (1) un conjunto de lecturas o *reads*, como los datos metagenómicos virales que contienen lecturas de virus y, (2) un archivo `pkl`¹⁵, que son los parámetros para el modelo. Es un modelo de clasificación jerárquico que puede realizar una clasificación taxonómica a nivel de lectura desde el orden hasta el género para nuevas especies. La combinación de la codificación basada en *embedding* de *k*-meros, redes neuronales convolucionales (CNN) organizadas jerárquicamente y una capa de rechazo cuidadosamente entrenada, hace que sea capaz de asignar etiquetas taxonómicas correctas para lecturas de nuevas especies [132].

Este método proporciona agrupación de lecturas, cuyo objetivo es clasificar/agrupar las lecturas en intervalos de modo que las lecturas del mismo intervalo pertenezcan al mismo grupo taxonómico [132]. El agrupamiento de lecturas metagenómicas

¹⁵Los archivos `pkl` también conocidos como archivos `pickle`, son un formato de archivo binario utilizado en Python para serializar y deserializar objetos Python.

puede proporcionar información en el análisis de la composición y también ayuda a conseguir un mejor ensamblaje y comparación del metagenoma, que son retos computacionales para los datos metagenómicos [132, 134]. CHEER está estrechamente relacionado con el agrupamiento dependiente de la taxonomía, el cual lleva a cabo la clasificación comparando lecturas con datos de entrenamiento con orígenes filogenéticos conocidos [132].

El componente clave de esta herramienta es un modelo de árbol que consiste en múltiples clasificadores desde el orden hasta el género. Para llevar a cabo la clasificación filogenética de las lecturas de nuevas especies, o incluso de nuevos géneros, la clasificación se realiza utilizando un enfoque descendente desde la raíz hasta el nodo hoja [132]. La capa superior es una CNN entrenada que puede rechazar lecturas que no pertenezcan a virus de ARN, lo cual es un paso de pre-procesamiento importante para los datos metagenómicos virales debido a la contaminación del genoma del huésped. Luego de este paso, las lecturas filtradas, que en su mayoría son lecturas de ARN viral, se introducen en el modelo de clasificación jerárquica, en donde el clasificador a nivel de orden se encarga de clasificar las lecturas en sus órdenes de origen. Luego, cada orden tiene un clasificador entrenado por separado para asignar las lecturas de entrada a las familias dentro de ese orden. Además, se tiene que para todas las lecturas asignadas a una familia, el clasificador de familia asigna las lecturas a diferentes géneros dentro de esa familia [132]. CHEER también implementa funciones de *early stop* o parada temprana en cada nivel para detener la ruta de clasificación en rangos superiores. Esta función puede acomodar la clasificación taxonómica para especies de nuevos géneros o incluso rangos superiores, y sirve además como medida de control de calidad para evitar clasificaciones erróneas si la confianza de la predicción es baja [132].

Cada clasificador en el modelo de árbol se implementa utilizando redes neuronales convolucionales, cuyos filtros de convolución sirven para aprender características de secuencias bien conservadas en diferentes clases [132]. Por otro lado, para incorporar tanto la composición de k -meros como la información sobre su orden, construyeron una capa de embedding basada en Skip-Gram [135], que puede aprender qué k -meros tienden a aparecer cerca unos de otros. Este embedding se utiliza en el procesamiento del lenguaje natural para aprender las relaciones semánticas y sintácticas de las frases. De este modo, una red neuronal con una capa oculta se entrena para asignar una palabra a un vector n -dimensional, de forma que las palabras que suelen aparecer juntas estén más cerca en el espacio n -dimensional. Para la tarea de clasificación de lecturas de ADN, los k -meros son las palabras y la capa de embedding mapea los k -meros próximos a vectores de alta similitud [132]. La alta acumulación de datos metagenómicos virales, tiene como consecuencia que hayan casos en los que una nueva especie no pertenece a ningún género existente y, por tanto, la asignación de etiquetas debería detenerse en los rangos superiores.

BERTax: Esta aplicación fue desarrollada por [131] y publicada en el 2022. Está basada en una red neuronal profunda o *deep neural networks* (DNN), que hace uso del procesamiento del lenguaje natural para clasificar taxonómicamente con precisión el super-reino y el filo de secuencias de ADN sin necesidad de un pariente

representativo conocido de una base de datos [131]. Para usar BERTax se requiere inicialmente de un modelo BERT pre-entrenado, aunque si se desea re-entrenar o afinar BERTax en datos específicos o con etiquetas taxonómicas personalizadas, se puede necesitar archivos adicionales que proporcionen los datos de entrenamiento y sus correspondientes etiquetas taxonómicas. Además se necesita una base de datos de referencia que contenga información taxonómica para la tarea de clasificación; esta base de datos suele incluir información sobre diferentes rangos taxonómicos, como especie, género, familia, etc., junto con sus correspondientes etiquetas o identificadores. Finalmente, se toman datos de secuencias de ADN como entrada para la clasificación taxonómica. Estas secuencias suelen ser lecturas cortas obtenidas de muestras metagenómicas, que pueden proporcionarse como un archivo en formato FASTA o FASTQ [131].

La clasificación de secuencias de ADN se puede realizar en tres niveles taxonómicos diferentes: super-reino (arqueas, bacterias, eucariotas y virus), filo y género. La novedad fundamental de este trabajo fue asumir que el ADN es un “lenguaje”, para así clasificar el origen taxonómico basándose en la comprensión de este lenguaje en lugar de hacerlo por la similitud local con genomas conocidos en una base de datos [131]. BERTax no se limita a regiones codificantes o a super-reinos específicos como bacterias o virus, sino que puede clasificar potencialmente cualquier región del genoma, de cualquiera de los cuatro super-reinos, y no requiere secuencias similares en la base de datos [131]. Este programa se basa en la arquitectura de procesamiento de lenguaje natural (PLN) de última generación BERT (codificador-representación bidireccional de transformadores) [133], que se adapta a la tarea de clasificación taxonómica mediante capas adicionales. La clasificación por medio de este enfoque necesita menos representantes exactos en los datos de entrenamiento, ya que las secuencias de entrenamiento no se memorizan [131]. BERT se apoya en un transformador que emplea el mecanismo de *auto-atención* [133]. La auto-atención es un método que determina, de forma autónoma, qué partes de las entradas son relevantes entre sí. Esto permite a la arquitectura del transformador procesar los datos secuenciales no en un orden predefinido, lo cual hace que el proceso de formación sea más rápido [133].

El proceso de formación de BERTax se divide en dos partes: En primer lugar, se pre-entrena un modelo BERT de forma no supervisada, lo que significa que las variables objetivo -qué secuencia pertenece a qué clase taxonómica- no son conocidas por el modelo, con el objetivo de aprender la estructura general del lenguaje del ADN genómico. Luego de esto, el modelo BERT pre-entrenado se combina con capas taxonómicas y se ajusta a la tarea específica de predecir las clases taxonómicas [131]. De este modo, BERTax es una herramienta para la clasificación taxonómica de secuencias de ADN (reads, contigs o scaffolds) utilizando redes neuronales profundas. Muestra un gran potencial para secuencias sin especies estrechamente relacionadas en la base de datos de referencia o en el conjunto de datos de entrenamiento. Debido a su alta sensibilidad y especificidad, relativamente independientes de las especies estrechamente relacionadas en los datos de entrenamiento, BERTax es muy adecuado con muestras metagenómicas [131].

Pysster: Es una paquete de python publicado en el 2018 por [141], el cual se creó pa-

ra entrenar redes neuronales convolucionales en datos de secuencias biológicas. El formato específico y la estructura de los datos de entrada dependen de la tarea que esté realizando. Por ejemplo, si se usa para descubrir motivos de secuencias, que son patrones cortos conservados dentro de las secuencias de ADN. Los datos de entrada suelen ser un conjunto de secuencias de ADN o un archivo que contiene secuencias de ADN en formato FASTA. Si se desea hacer tareas de clasificación de secuencias, como distinguir entre diferentes elementos funcionales o regiones genómicas basándose en sus secuencias de ADN, los datos de entrada para las tareas de clasificación suelen consistir en secuencias etiquetadas, en las que cada secuencia está asociada a una clase o etiqueta específica. Estas secuencias y etiquetas pueden proporcionarse como archivos separados o como un único archivo con pares secuencia-etiqueta. Otro uso puede ser para extraer características de las secuencias de ADN que puedan utilizarse para análisis posteriores o en tareas de aprendizaje de máquina. Estas características pueden incluir frecuencias k -mer, pesos posicionales u otras representaciones que capturen las características de las secuencias, en cuyo caso los datos de entrada serían las propias secuencias de ADN [141].

Las secuencias se clasifican aprendiendo motivos de secuencia y estructura, y el paquete ofrece un procedimiento automatizado de optimización de hiperparámetros y opciones para visualizar los motivos aprendidos junto con información sobre su enriquecimiento posicional y de clase. Por medio de esta herramienta se puede hacer clasificaciones multiclase, mono-etiqueta o multi-etiqueta, interpretación de los motivos aprendidos en términos de enriquecimiento posicional y de clase y de co-ocurrencia de motivos, también funciona para secuencias de entrada sobre alfabetos definidos por el usuario, por ejemplo, sirve con datos de ADN, ARN y proteínas [141].

La arquitectura básica de la red consiste en un número variable de capas convolucionales y de agrupamiento máximo (o pooling layers), seguidas de un número variable de capas densas. Estas capas se intercalan con capas de *dropout* después de la capa de entrada y después de cada capa de agrupamiento máximo y capa densa. Tiene una ventaja en cuanto a la información de estructura, es decir, para ARN en forma de cadenas de puntos y corchetes u otro tipo de notación, esta se puede incorporar a la red codificando la cadena de secuencia dada sobre un alfabeto de tamaño N y la cadena de estructura correspondiente sobre un alfabeto de tamaño M en una única cadena nueva utilizando un alfabeto ampliado de tamaño $N \cdot M$. Luego, se entrena la red con estas nuevas cadenas, que pueden descodificarse de nuevo en las cadenas originales tras el entrenamiento para permitir la visualización de dos motivos como matrices de posición-peso [141].

Estos algoritmos sirven de base para entender qué tipo de modelo se desea y qué clase de parámetros y condiciones se deben tener en cuenta considerando el tipo de datos que se espera clasificar. Aunque se mostraron algunos de los algoritmos más usados, no se deben dejar de lado otros que puedan ser igual o más útiles a la hora de extraer características. También resulta útil tener en cuenta otros modelos y enfoques que ayuden a explorar otras formas de abordar el problema de este trabajo. En un capítulo posterior se explican un par de algoritmos adicionales que fueron utilizados como base para algunos de los modelos que se probaron para el problema de clasificación.

Implementación de las aproximaciones de modelos y resultados

Este capítulo se encarga inicialmente de profundizar en los dos conjuntos de datos utilizados en este trabajo: el primero siendo los datos obtenidos del proyecto “Expedición virológica en ecosistemas representativos de Colombia: selva húmeda tropical de la Sierra Nevada de Santa Marta”, y el segundo, los datos recolectados a través del Centro Nacional para la Información Biotecnológica (NCBI). Dichos datos hacen parte del grupo de secuencias que se utilizaron para el entrenamiento y prueba del modelo de clasificación. Además, se mencionan y explican de forma breve las distintas herramientas bioinformáticas que se emplearon en la limpieza, procesamiento y conversión de las secuencias, según correspondiera.

Además del pre-procesamiento de datos, se ahonda en la implementación de los modelos y algoritmos que se utilizaron y probaron con el conjunto de datos, además de presentar y analizar los diferentes resultados que se obtuvieron para cada modelo. En la implementación y elaboración de los modelos, se consideraron otras formas en las cuales se podía abordar el problema de interés. Debido a que la pregunta de investigación tiene como objetivo dar respuesta a qué tipo de representación secundaria se puede utilizar para la clasificación de virus de ARN, se consideraron otro tipo de algoritmos que además de la etiqueta, usaran los patrones existentes dentro de las secuencias para así hacer una predicción o clasificación de los metagenomas.

De este modo, se consideraron cuatro algoritmos, que por su naturaleza fueran capaces de extraer información útil para poder llevar a cabo la clasificación. Estos algoritmos son: un perceptrón multicapa, árboles de sufijos, modelos ocultos de Markov (Hidden Markov Model o HMM) y una red neuronal convolucional con memoria de largo y corto plazo o CNN-LSTM. Cada uno de los modelos que se explicarán se implementaron para cada una de las tres diferentes representaciones de estructura secundaria: árbol ampliado, HIT y árbol de *grano queso* (para esta última representación se utilizó la representación extendida, ver Fig. 8 (b)).

4.1. Datos de entrenamiento y prueba

Este trabajo consideró los resultados obtenidos en [4] tanto para realizar la base de datos de entrenamiento, como para crear la base de datos de prueba a partir de los datos me-

tavirómicos experimentales. Los datos metavirómicos fueron muestreados en el marco del proyecto “Expedición virológica en ecosistemas representativos de Colombia: selva húmeda tropical de la Sierra Nevada de Santa Marta”. Estos muestreos se realizaron en tres lugares diferentes de la Sierra Nevada de Santa Marta, los cuales eran cercanos a bordes de fragmentos de bosque. El material genético fue extraído con el kit “Viral RNA Mini Kit” y fue secuenciado con la tecnología de secuencia Illumina, para un total de tres librerías. En [4] se utilizaron además cinco librerías pertenecientes al metaviroma de *Drosophila suzukii* como datos de referencia, para el control y verificación del procesamiento.

Librería	Número de metacontigs	Número de segmentos
Primera	180	5714
Segunda	83	5125
Tercera	15334	473903

Tabla 2: Librerías obtenidas en [4] por secuenciación, junto con la cantidad respectiva de metacontigs y finalmente, el conjunto de los segmentos obtenidos de tamaño 90 para cada secuencia en sus respectivas librerías.

La tabla 2 contiene los valores de los *metacontigs*¹⁶ obtenidos originalmente para cada una de las librerías. Para procesar estas secuencias se implementó un código en `Perl` que permitió extraer cada 20 pasos, ventanas de tamaño 90 para cada uno de los metacontigs, es decir, se cortaron las secuencias en segmentos de tamaño 90. Esto se realizó con el objetivo de conservar la mayor cantidad de información posible que pudiera ser utilizada posteriormente en el proceso de predicción de estructuras secundarias. Estas librerías son metagenomas virales y por ende no todo lo que se encuentra allí contiene necesariamente información funcional.

La base de datos de entrenamiento se consolidó teniendo en cuenta el análisis de datos metavirómicos hecho por [4], que evidencia que en la clasificación del metagenoma de *Culex sp.* se encuentran las familias *Orthomyxoviridae*, *Reoviridae*, *Retroviridae* y *Arteriviridae* [4], por lo tanto, surgió un interés en revisar si diferentes representaciones secundarias de estas secuencias metagenómicas resultarían útiles para realizar una clasificación dentro de este grupo de familias. A continuación se presenta una breve descripción-definición de las familias de interés:

- ***Orthomyxoviridae***: Familia de virus ARN monocatenarios que tienen un virión esférico o filamentosos con numerosas proyecciones de glicoproteínas superficiales, una nucleocápside helicoidal y un genoma formado por seis a ocho segmentos de ARN, y que incluyen los virus causantes de la influenza *A*, la influenza *B* y la influenza *C*, así como un género (*Thogotovirus*) de virus transmitidos por garrapatas que infectan ocasionalmente a los seres humanos [9].
- ***Retroviridae***: Familia de virus ARN monocatenarios que producen transcriptasa inversa mediante la cual se sintetiza ADN utilizando su ARN como molde y se

¹⁶Un cóntigo o contig es un conjunto de segmentos o secuencias de ADN o ARN que se superponen parcialmente de forma tal que colectivamente dan una representación continua de una región genómica. Un metacóntigo es un conjunto de cóntigos.

incorpora al genoma de las células infectadas, que a menudo son tumorigénicos y que incluyen los virus espumosos, el VIH, el HTLV-I, el virus del sarcoma de Rous, el VIS y los agentes causantes de la leucosis aviar, la anemia infecciosa equina, la neumonía progresiva ovina y el jaagsiekte [10].

- ***Arteriviridae***: Familia de virus de ARN de cadena positiva, envueltos, del orden de los Nidovirales, y con forma icosaédrica que infectan a vertebrados. Entre los organismos hospedadores se encuentran équidos, cerdos, zarigüeyas, primates no humanos y roedores [11].
- ***Sedoreoviridae***: Familia de virus que incluye virus que tienen partículas centrales relativamente lisas, que contienen 10-12 segmentos de dsARN lineal. Los hospedadores son mamíferos, aves, crustáceos, artrópodos, algas y plantas [12].
- ***Spinareoviridae***: Familia de virus que contienen 9-12 segmentos de dsARN lineal y son icosaédricas. Los hospedadores son mamíferos, animales acuáticos (peces, mamíferos, crustáceos, moluscos), aves, reptiles, artrópodos, hongos y plantas [13].

Este resultado inicial por parte de [4] permitió filtrar la cantidad de familias con las cuales se esperaba realizar la clasificación de los metagenomas. Posterior a esto, se inició una búsqueda de secuencias que se utilizarían como referencia en el modelo; dichos datos se tomaron de la base de datos del Centro Nacional para la Información Biotecnológica (National Center for Biotechnology Information) o NCBI, de allí se tomaron las secuencias de virus que correspondieran a las familias *Arteriviridae*, *Orthomyxoviridae*, *Retroviridae*, *Sedoreoviridae*, y *Spinareoviridae*. Las secuencias de estas familias se transformaron a diferentes representaciones secundarias para luego ser utilizadas en el entrenamiento del modelo de clasificación, es decir, dichas secuencias serán parte de la base de datos estándar con la cual se entrenará el modelo.

En la tabla 3 se encuentra un resumen de los datos que resultaron de la búsqueda y limpieza inicial. Al descargar todas las secuencias correspondientes a las cinco familias de interés, se mantuvieron las secuencias que fueran genomas completos y que estuvieran verificadas, esto ayuda a mejorar la redundancia en los datos que se descargan. De este modo, la base de datos estándar o de entrenamiento contiene 9768 secuencias: 81.6 % siendo de la familia *Retroviridae*, 15.8 % de la familia *Arteriviridae*, 1 % de la familia *Sedoreoviridae*, 0.8 % a la familia *Spinareoviridae*, y 0.5 % a la familia *Orthomyxoviridae*.

La familia *Orthomyxoviridae* cuenta con 56 secuencias de virus, donde todas pertenecen al género *Alphainfluenzavirus*, 42.8 % pertenece a influenza A en gallinas, 41 % corresponde a influenza A en patos, 14.2 % corresponde a influenza A en cerdos y 1.8 % es de una especie desconocida. En el caso de la familia *Retroviridae*, se encontraron 7977 secuencias de virus pertenecientes a 11 géneros (ver Tabla 3), donde 81.6 % de las secuencias corresponde a humanos, 8.8 % a primates, 3.2 % de aves, 1.6 % a bovinos, 1.2 % a felinos domésticos, 1 % a cabras, 0.8 % a ratas, 0.4 % a ovinos, 0.12 % a porcinos, 0.11 % a koalas y pumas, 0.08 % a peces, 0.07 % a especies no especificadas. Para la familia *Arteriviridae*, se obtuvieron 1552 secuencias, cuyos virus se asociaron a 10 géneros de ésta familia; 89.6 % de las secuencias corresponden a porcinos, el 2.5 % a primates, 5.8 % a equinos, diferentes roedores sumaron un 0.9 %, y varios tipos de mamíferos juntaron 0.8 %.

Familia	Género	Tipo de genoma	Número de secuencias
<i>Orthomyxoviridae</i>	<i>Alphainfluenzavirus</i>	ssARN (-)	56
<i>Retroviridae</i>	<i>Alpharetrovirus</i> <i>Betaretrovirus</i> <i>Deltaretrovirus</i> <i>Epsilonretrovirus</i> <i>Gammaretrovirus</i> <i>Lentivirus</i> <i>Bovispumavirus</i> <i>Equispumavirus</i> <i>Felispumavirus</i> <i>Prosimiispumavirus</i>	ssARN (+)	7977
<i>Arteriviridae</i>	<i>Muarterivirus</i> <i>Alphaarterivirus</i> <i>Lambdaarterivirus</i> <i>Deltaarterivirus</i> <i>Epsilonarterivirus</i> <i>Iotaarterivirus</i> <i>Thetaarterivirus</i> <i>Betaarterivirus</i> <i>Gammaarterivirus</i> <i>Kappaarterivirus</i>	ssARN (+)	1552
<i>Sedoreoviridae</i>	<i>Orbivirus</i> <i>Phytoreovirus</i> <i>Rotavirus</i> <i>Seadornavirus</i>	dsARN	102
<i>Spinareoviridae</i>	<i>Aquareovirus</i> <i>Coltivirus</i> <i>Fijivirus</i> <i>Orthoreovirus</i>	dsARN	81

Tabla 3: Familias de la base estándar con los correspondientes géneros de las secuencias, tipo de genoma, y la cantidad de secuencias que contienen un genoma completo.

La familia *Sedoreoviridae* tiene 102 secuencias representates, de las cuales 22.5% corresponden al género *Seadornavirus*, 32.3% al género *Orbivirus*, 33.3% al género *Rotavirus* y 11.7% al género *Phytoreovirus*. Finalmente, la familia *Spinareoviridae* tiene 81 secuencias, donde 9.8% pertenecen al género *Aquareovirus*, 14.8% a *Coltivirus*, 25.9% a *Fijivirus*, y 49.3% a *Orthoreovirus*. Estas dos últimas familias se analizaron teniendo en cuenta el género y no la especie, esto se debió a que en los datos que provee el NCBI, dicha información no se encuentra disponible.

Estos datos hacen parte del conjunto de datos con los que se entrenará y validará el modelo. Es importante notar que se presenta un desequilibrio en cuanto a la cantidad de datos representativos para cada familia, lo cual puede crear problemas en el desempeño de los modelos. La otra parte del conjunto de datos son los datos de metagenomas virales del proyecto “Expedición virológica en ecosistemas representativos de Colombia:

selva húmeda tropical de la Sierra Nevada de Santa Marta” (ver tabla 2). Ya que se tiene cuáles datos se utilizarán en el entrenamiento y prueba, resulta necesario ahora enfocarse en el procesamiento de los mismos para obtener datos que estén limpios y cumplan ciertas condiciones para asegurar un nivel de conservación alto, y así evitar errores tanto en el proceso de la transformación a las estructuras secundarias de interés, como en el entrenamiento y prueba del modelo.

4.2. Pre-procesamiento de los datos

4.2.1. Pre-procesamiento de los datos estándar o de entrenamiento

Para el procesamiento de los datos descargados del NCBI se utilizaron distintas herramientas bioinformáticas que permitieran la extracción, limpieza y transformación de estas secuencias, para que más adelante pudiesen ser utilizados en el entrenamiento del modelo. Luego de la limpieza inicial, que permitió omitir secuencias que no fuesen genomas completos o no estuvieran verificadas, se pasó a estudiar y analizar cada una de las familias para evaluar qué metodología aplicar, ya fuese de forma colectiva o individual.

Para la familia *Orthomyxoviridae* y la división de especie de humanos en la familia *Retroviridae*, fue necesario utilizar CD-HIT. CD-HIT es un programa utilizado para agrupar secuencias biológicas con el fin de reducir la redundancia de secuencias y mejorar el rendimiento de posteriores análisis. De dicho paquete, se implementó en particular el script CD-HIT-EST, este agrupa un conjunto de datos de nucleótidos en clusters que cumplen un umbral de similitud definido, normalmente una identidad de secuencia. La entrada es un conjunto de datos de ADN/ARN en formato fasta y la salida son dos archivos: un archivo fasta de secuencias representativas y un archivo de texto con la lista de agrupaciones [24, 25].

Para estudiar si se presentaban diferencias significativas, se agruparon las secuencias asociadas a la familia *Orthomyxoviridae* y, por otro lado, las secuencias asociadas a la especie de humanos de la familia *Retroviridae* con un umbral de identidad de 0.80, es decir, del 80%; este umbral se calcula tomando el número de bases idénticas en el alineamiento y se divide por la longitud total de la secuencia más corta. Este primer paso, para estas dos familias, resultó en una reducción significativa de la redundancia, ya que pasó de 56 y 6510 secuencias, respectivamente, a 10 y 151, es decir, se mantuvo el 17.8% y el 2.3% de la secuencias originales. La decisión de aplicar CD-HIT a estas dos familias se debió a que durante el proceso de análisis se evidenciaron dificultades cuando se continuaba con la metodología de las otras familias en los procesos posteriores, por ello, se tomó la decisión de usar este programa, el cual ayudó a mejorar y garantizar un buen nivel en la conservación y reducción de redundancia en las secuencias para los posteriores análisis.

Subsiguiente a la mejora en la redundancia de las secuencias, se continuaron los análisis con la herramienta AliView y Clustal-0. AliView es un visor y editor de alineamientos diseñado para satisfacer los requisitos de los conjuntos de datos filogenéticos de la era de la secuenciación de nueva generación. Esta herramienta facilita la inspección, clasificación, eliminación, fusión y realineación de secuencias como parte del proceso de filtrado manual de grandes conjuntos de datos [27]. Esta herramienta ayudó a re-evaluar las divisiones que se tenían originalmente para los alineamientos, por lo cual, finalmente se decidió agrupar

por especie o género a las familias, facilitando así no solo el alineamiento múltiple de las familias, sino también mejorando la visualización de los resultados. Para llevar a cabo el alineamiento se utilizó Clustal Omega o `Clustal-0`, el cual es un programa que realiza alineamientos múltiples (multiple sequence alignments o MSAs) de propósito general para proteínas y ADN/ARN. Utiliza *seeded guide trees* o árboles guía sembrados y técnicas de los modelos ocultos de Markov (HMM) para generar alineamientos entre tres o más secuencias [58, 59].

Después de obtener alineamientos múltiples funcionales, es decir, alineamientos sin secuencias repetidas, o con demasiados huecos, se procedió a utilizar **RNAz**. **RNAz** es un paquete informático ampliamente utilizado para la detección de novo de ARN no codificantes estructurados en datos de genómica comparativa [67], es un método que combina el análisis comparativo de secuencias y la predicción de estructuras. Consta de dos componentes básicos: (i) una medida de la conservación de la estructura secundaria del ARN basada en el cálculo de una estructura secundaria de consenso, y (ii) una medida de la estabilidad termodinámica que se normaliza con respecto tanto a la longitud de la secuencia como a la composición de bases [68].

El primer enfoque calcula el MFE (Minimum Free Energy) como medida de la estabilidad termodinámica de una secuencia utilizando, por ejemplo, `RNAfold`. Sin embargo, la MFE depende de la longitud y de la composición de bases de la secuencia y es, por tanto, difícil de interpretar en términos absolutos [67, 68]. Por ello, **RNAz** calcula una medida normalizada de la estabilidad termodinámica comparando el MFE E de una secuencia dada con los MFEs de un gran número de secuencias aleatorias de la misma longitud y composición de bases. Recordemos que la puntuación z o z -score se calcula como $z = (E - \bar{E})/\sigma$, donde \bar{E} y σ son la media y la desviación estándar, respectivamente, de las MFE de las muestras aleatorias.

Las puntuaciones z negativas indican que una secuencia es más estable de lo esperado por azar [67, 68]. En el segundo enfoque **RNAz** predice una estructura secundaria de consenso E_A , donde compara este MFE de consenso con el MFE medio de las secuencias individuales \bar{E} y calcula un índice de conservación de la estructura: $SCI = E_A/\bar{E}$. El SCI será alto si las secuencias se pliegan juntas igual de bien que si se pliegan individualmente. Por otro lado, el SCI será bajo si no se encuentra un plegamiento de consenso. Los valores de la puntuación z y SCI se utilizan para clasificar un alineamiento como ARN estructural u otro. Para ello, **RNAz** utiliza un algoritmo de aprendizaje de máquina de soporte vectorial (Support Vector Machine o SVM) [60, 67, 68].

Es importante entender la metodología que **RNAz** utiliza, ya que este paso resulta de gran importancia debido a las distintas variables y parámetros que se deben considerar en su aplicación, para así poder obtener estructuras secundarias que sean representativas de cada una de las familias. En este paso de la metodología se consideraron varias posibilidades, sin embargo, se optó que para los alineamientos se tomaran ventanas superpuestas de tamaño 90, con un paso de tamaño 20, y no se considerara en ningún caso secuencias de referencia; esto ayudó a mejorar el pre-procesamiento y filtrado de las secuencias. Además, fue indispensable considerar -por la cantidad de secuencias- la opción donde se daba un puntuación en ambos sentidos, ya que esta información no se tenía de antemano. **RNAz** fue usado con todas las familias, excepto por la familia *Orthomyxoviridae*. Esto se debió a que

al aplicarlo a ésta, se observó que no le era posible encontrar secuencias consenso, esto quiere decir que el programa no logró encontrar secuencias conservadas, sin embargo, la no presencia de secuencias conservadas no implica que no existan estructuras secundarias en las secuencias de virus de esta familia, por ello, para obtener las estructuras secundarias se determinó que era mejor utilizar *RNAfold* [18, 78], teniendo en cuenta la metodología usada -explicada más adelante- con los datos metagenómicos.

Para el resto de familias, se tomaron los resultados obtenidos con *RNAz* y se escogieron las secuencias consenso considerando lo siguiente: las puntuaciones z , los valores de *SCI* y el valor p^{17} de *RNAz*. Para todas las familias se consideraron las estructuras secundarias cuya puntuación z fuera menor o igual a -0.80 , el valor de *SCI* fuese mayor o igual a 0.80 y se evaluó el SVM RNA-class probability -este valor es el *valor p* de *RNAz*- teniendo en cuenta las dos condiciones anteriores. Después de realizar los diferentes procesos, ya nombrados, se obtuvieron 14813 secuencias que sirven como representantes de las diferentes familias (ver Fig. 18).

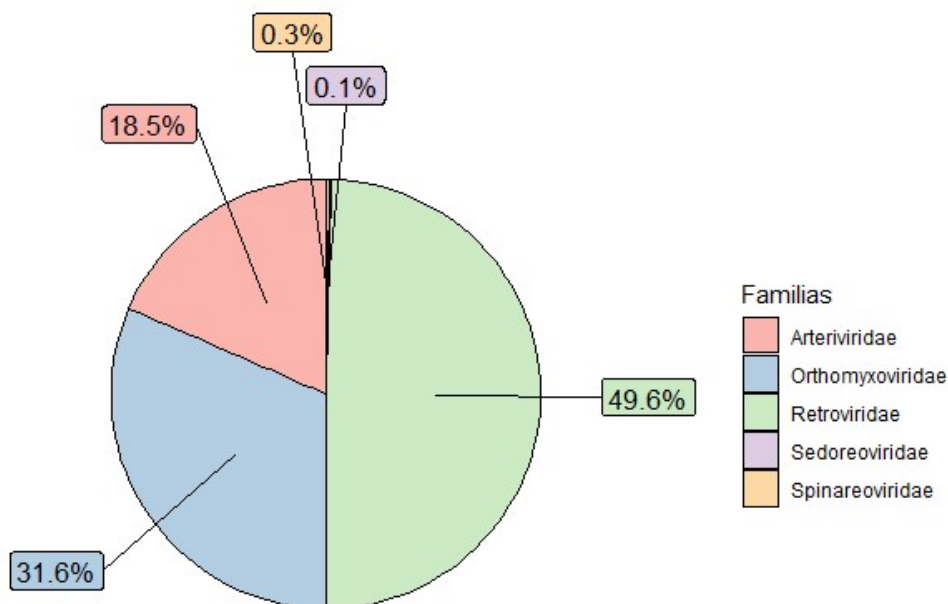


Figura 18: Porcentajes correspondientes a las secuencias que quedaron representando a las cinco familias.

4.2.2. Pre-procesamiento datos metagenómicos o de prueba

Como se mencionó en la primera parte del capítulo, los datos originales obtenidos en [4] están compuestos por las tres primeras librerías de la tárrado 2. abla 2 y, las otras cinco corresponden a datos de referencia. Las secuencias que se encuentran en dichas librerías son metagenomas virales, que fueron sometidos a diferentes métodos y procesos que arrojaron datos limpios y completamente secuenciados. Sin embargo, a dichos datos era necesario realizarles una modificación adicional antes de iniciar con el proceso de conversión.

La modificación que se llevó a cabo con los metagenomas virales consistió en partir las

¹⁷Ver apéndice G.

secuencias originales en segmentos de secuencia más pequeños, esta elección se debió a que en principio, los metagenomas con los cuales se está trabajando varían bastante de tamaño, desde 500 a 20000 nucleótidos. Por lo tanto, se quería por un lado estandarizar los tamaños de los metagenomas y, por otro lado, se tenía la intención de mejorar la posibilidad de que se encontraran estructuras secundarias funcionales o informativas. Al tener secuencias muy grandes, puede perderse información al momento de predecir estructuras secundarias. Este proceso se realizó teniendo en cuenta los parámetros utilizados con *RNAz*.

Para ello, se implementó una metodología similar a lo que hace *RNAz*. Se evaluó qué tamaño de ventana sería el más propicio para éstos datos, de tal manera que se mantuviera la mayor cantidad posible de información dentro de las secuencias, es decir, al tomar ventanas de tamaños pequeños, se corría el riesgo de partir o ignorar segmentos de secuencia que fueran informativos, así mismo, si se tomaban tamaños de ventanas muy grandes, se agregaba ruido a la información, causando que los segmentos no produjeran representaciones secundarias precisas. La decisión de emplear ventanas de tamaño 90 en el código de *Perl* se basó en un análisis de las secuencias de los metagenomas virales, cuyos tamaños promediados indicaron que este tamaño sería el más favorable dentro de las opciones que se estaban considerando. Al igual que con los otros datos, se partieron las secuencias originales en ventanas de tamaño 90, y este proceso se realizó de forma iterativa cada 20 pasos.

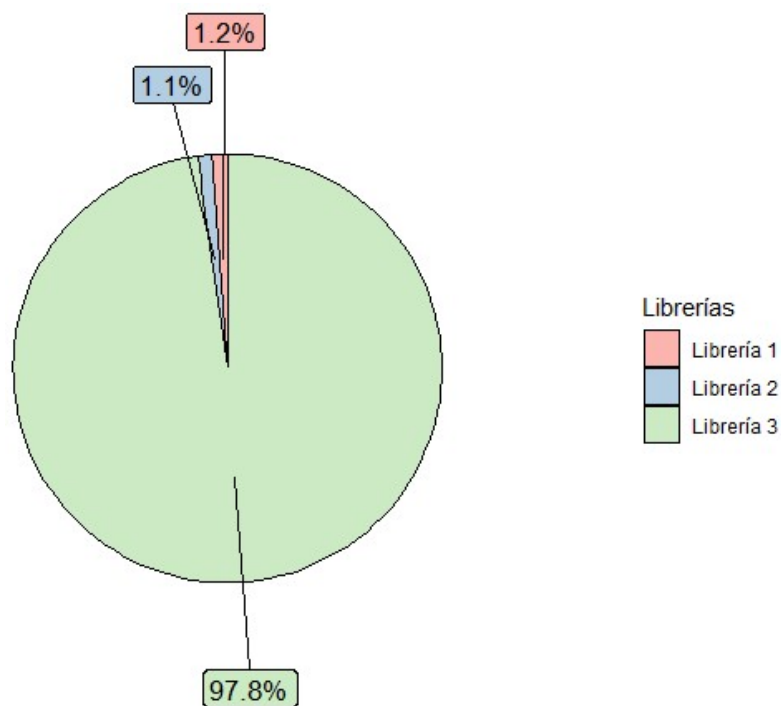


Figura 19: Distribución correspondiente a las tres librerías de secuencias metagenómicas de la Sierra Nevada de Santa Marta.

Más allá de la separación que se hizo para cada secuencia dentro de cada librería, no fue necesario hacer más cambios a las secuencias. El conjunto total de metagenomas con el cual se va a realizar la prueba es de 484742 secuencias, correspondientes a los metagenomas virales de la Sierra Nevada de Santa Marta. Teniendo esto, se pasó a utilizar *RNAfold*

[18, 78]. `RNAfold` es la principal herramienta de predicción de estructuras secundarias que calcula la estructura de mínima energía libre (MFE) e imprime la estructura en notación de punto-corcheo con su respectivo valor de MFE [18, 79, 80]. Los parámetros considerados para `RNAfold` fueron los que vienen por defecto en el programa. Debido a que los datos se encontraban limpios, no era necesario agregar ningún proceso adicional. En la tabla 4 se encuentra la información de los datos utilizados en la parte del entrenamiento de los modelos, y en la tabla 5 están los datos de los metagenomas de la Sierra Nevada de Santa Marta, que son utilizados en el proceso de prueba de los modelos.

Familia	CD-HIT	Clustal-0	AliView	RNAz	RNAfold	N° secuencias
<i>Orthomyxoviridae</i>	Sí	Sí	Sí	No	Sí	4677
<i>Arteriviridae</i>	No	Sí	Sí	Sí	No	2738
<i>Retroviridae</i>	Sí	Sí	Sí	Sí	No	7340
<i>Sedoreoviridae</i>	No	Sí	Sí	Sí	No	19
<i>Spinareoviridae</i>	No	Sí	Sí	Sí	No	39

Tabla 4: Tabla con información resumida sobre los programas utilizados con cada familia, y el número de secuencias obtenidas para realizar el proceso de conversión.

Librería	RNAfold	Número de secuencias
Primera	Sí	5714
Segunda	Sí	5125
Tercera	Sí	473903

Tabla 5: Tabla con información resumida sobre los programas utilizados con cada librería, y el número de secuencias obtenidas para realizar el proceso de conversión.

4.3. Conversión de los datos a estructuras secundarias

Al tener todas las estructuras secundarias de notación dot-bracket, correspondiente a las secuencias de ambos conjuntos de datos, se procedió a implementar un código en `Python`, el cual utiliza el paquete de `ViennaRNA` y las funciones que este ya tiene, para hacer el cambio de la representación secundaria básica a las representaciones de árbol ampliado, HIT y de árbol de *grano grueso*.

Seguido a esta conversión, se generaron los archivos `csv` que une todas las secuencias en sus múltiples representaciones, para cada familia. En este paso se generaron dos archivos, uno corresponde al que se utiliza en el entrenamiento del modelo; este tiene las diferentes representaciones de estructuras secundarias junto con la etiqueta de familia, es decir, junto a cada representación, se encuentra el nombre de la familia a la cual pertenece (ver Fig. 20). El otro archivo, se ve similar, éste también contiene las distintas representaciones secundarias de los datos metagenómicos, la diferencia radica en que no hay una columna correspondiente a la familia. El archivo de los datos sin etiqueta, es decir, de las secuencias metagenómicas, es el que se utilizará en la prueba del modelo de clasificación.

primeras capas, y la función de activación utilizada en la última capa es softmax. El modelo usa `categorical_crossentropy` o entropía cruzada categórica como función de pérdida, `adam` como optimizador y la precisión como métrica. Estos conceptos se explican brevemente a continuación:

- **ReLU (Unidad lineal rectificada):** Es una función de activación utilizada en redes neuronales. Se define como la parte positiva de su argumento, es decir,

$$ReLU(z) = \max(0, z) = \begin{cases} z & \text{si } z > 0 \\ 0 & \text{de lo contrario.} \end{cases}$$

Esta función establece a todos los valores negativos en cero y permite el paso de todos los valores positivos sin cambios [81].

- **Softmax:** Es una función utilizada para normalizar la salida de una red y representar una distribución de probabilidad sobre las clases predichas. La salida de la función softmax es un vector de valores que deben sumar uno [81].
- **Entropía cruzada categórica:** Es una función de pérdida utilizada para calcular la diferencia entre la distribución de probabilidad predicha y la distribución de probabilidad real de las etiquetas [81].
- **Adam:** Es un algoritmo de optimización utilizado para actualizar los pesos de una red neuronal durante el entrenamiento. Adapta la tasa de aprendizaje de cada peso en función de la media de la magnitud de los gradientes recientes para ese peso, lo que permite una convergencia más rápida y un mejor rendimiento [81].

El modelo se entrenó con los datos de la base estándar, los cuales se dividieron en 80 % para entrenar y 20 % para validar. Posteriormente se probó el modelo con los datos metagenómicos. Este modelo contaba con un número de épocas de 100 y un tamaño de lote de 32. Dichos parámetros se escogieron considerando algunos aspectos como la complejidad del problema, la capacidad de memoria, recursos y la variabilidad de los datos, además se probó con múltiples parámetros para evaluar cuáles serían los más apropiados para la versión final del mismo.

4.4.2. Resultados modelo I

Árbol ampliado: Para esta primera representación se obtuvo que los datos metagenómicos corresponden únicamente a dos de las cinco familias: 177710 secuencias se clasificaron como *Orthomyxoviridae* y 307032 de *Retroviridae* (ver Fig. 21). Estos resultados no fueron sorprendentes teniendo en cuenta dos aspectos: el tipo de modelo y la cantidad de datos de entrenamiento. Estos últimos eran solo 14813, mientras que la cantidad de datos metagenómicos o de prueba son 484742, donde menos del 20 % corresponde a las familias de *Arteriviridae*, *Sedoreoviridae* y *Spinareoviridae*. Por lo tanto, se esperaba que no hubiesen o que fueran muy pocas las secuencias clasificadas para las familias menos representadas. En cuanto a las métricas, se obtuvo para los datos de prueba que el valor de la exactitud o accuracy, precisión, sensibilidad o recall y puntuación F fue de 1.

Por otro lado, el modelo obtuvo una precisión de entrenamiento del 48.68 % y una precisión de validación del 51.40 %. Esto sugiere que el modelo no se está ajustando

en exceso a los datos de entrenamiento y que tiene cierto nivel de generalización a datos no vistos. Sin embargo, los valores relativamente bajos indican que el modelo puede no estar captando correctamente los patrones y características subyacentes en los datos. Considerando estos resultados y los de prueba, se optó por aplicar una técnica de regularización. La regularización se refiere al uso de técnicas para evitar el sobreajuste y mejorar la capacidad de generalización de un modelo. Los métodos de regularización añaden un término de penalización a la función de pérdida del modelo, lo que reduce la complejidad del modelo disminuyendo los pesos hacia cero o limitándolos a un rango determinado. En este caso particular, se escogió usar la regularización L2; esta añade un término de penalización a la función de pérdida que es proporcional al cuadrado de los pesos. Esto promueve que el modelo use pesos más pequeños para evitar así el sobreajuste.

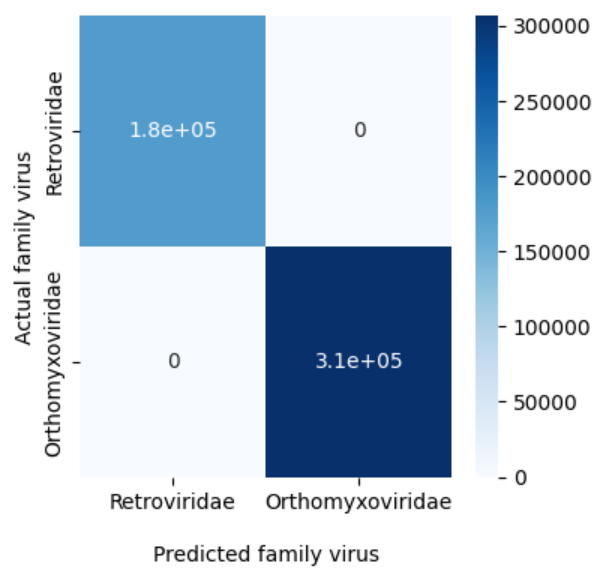


Figura 21: Matriz de confusión con resultados del modelo I para la representación de árbol ampliado con los datos de prueba.

A pesar de haber utilizado la regularización y ajustar diferentes parámetros, el rendimiento del modelo no mejoró con el tiempo. Estos problemas pueden deberse a que hay clases desequilibradas en el conjunto de datos de entrenamiento, haciendo que el modelo tenga dificultades para aprender y generalizar bien.

HIT: Para esta representación no fue posible usar este modelo. El problema radica en el diccionario que se define; en este, a cada carácter de la secuencia se le asigna un valor numérico, sin embargo, la naturaleza de esta representación secundaria resulta problemática ya que como se mencionó en un capítulo previo, la representación HIT a pesar de ser similar a la representación de árbol ampliado en cuanto a las etiquetas del alfabeto, tiene la gran diferencia de que se le añaden pesos, los cuales son numéricos, y es allí donde hay inconvenientes. No solo hay muchos pesos, es decir, muchos números dependiendo de la cantidad de nucleótidos apareados o no apareados, sino que además al ser números, esto representa un problema fundamental en la definición del diccionario porque no se puede hacer de forma efectiva y correcta

la asignación de valores.

Con esta representación se evidenciaron dos problemas principales: la memoria y la representación dispersa. El problema de la memoria radica en que cada carácter único debe estar representado por un índice distinto en el diccionario, y a medida que aumenta el número de caracteres únicos, también aumenta la memoria necesaria para almacenar el diccionario y los recursos para procesarlo. La representación dispersa hace referencia a que al tener secuencias con un gran número de caracteres únicos, pero sólo un pequeño subconjunto de ellos aparece con frecuencia, la representación del diccionario será dispersa. Esto significa que la mayoría de las entradas del diccionario tendrían muy pocas o ninguna ocurrencia, lo que podría ocasionar problemas a la hora de generalizar a partir de los datos disponibles. Por lo tanto, no fue posible establecer un diccionario con esta notación.

Árbol de grano grueso: Los resultados para esta representación fueron similares a los de árbol ampliado. Se obtuvo una clasificación de los datos metagenómicos únicamente para dos de las cinco familias, donde 190799 secuencias se clasificaron como *Retroviridae* y 293943 como secuencias de *Orthomyxoviridae* (ver Fig. 22). Al realizar la prueba del modelo con los datos metagenómicos, se obtuvo un valor de 1 para la exactitud, precisión, sensibilidad y puntuación F. De igual forma, el modelo obtuvo una precisión de entrenamiento del 48.59 % y una precisión de validación del 55.45 %, lo cual es una buena señal respecto a problemas de generalización que se habían presentado inicialmente. A este modelo también se le aplicó la regularización L2, sin embargo, los resultados con los datos de prueba no mejoraron.

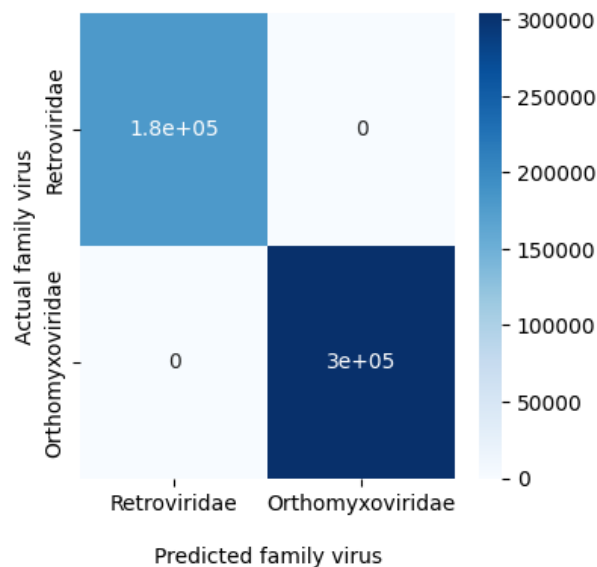


Figura 22: Matriz de confusión con resultados del modelo I para la representación de árbol de *grano grueso* de los datos de prueba.

De este modo, se puede concluir que el modelo I no capta de forma útil las características de ninguna de las representaciones de la estructura secundaria, lo cual induce a pensar que por la naturaleza del modelo dichas representaciones no son las mejores para llevar a cabo la tarea de clasificación con este tipo de modelos.

4.4.3. Modelo II: Árboles de sufijos

Antes de proceder a hablar del modelo, se explicará qué son los árboles de sufijo. Un árbol de sufijos es un árbol de prefijos comprimido, es decir, contiene todos los sufijos del texto dado como sus claves y las posiciones en el texto como sus valores. La construcción de un árbol de este tipo facilita la localización de subcadenas, o encontrar coincidencias de un patrón de expresión regular. El árbol de sufijos para una cadena S de longitud n se define como un árbol tal que [100]:

- El árbol tiene exactamente n hojas numeradas de 1 a n .
- Salvo por la raíz, cada nodo interno tiene al menos dos hijos.
- Cada arista está etiquetada con una subcadena no vacía de S .
- Dos aristas que parten de un nodo no pueden tener etiquetas de cadena que empiecen por el mismo carácter.
- La cadena obtenida al concatenar todas las etiquetas encontradas en el camino de la raíz a la hoja i deletrea el sufijo $S[i \dots n]$, para i desde 1 hasta n .

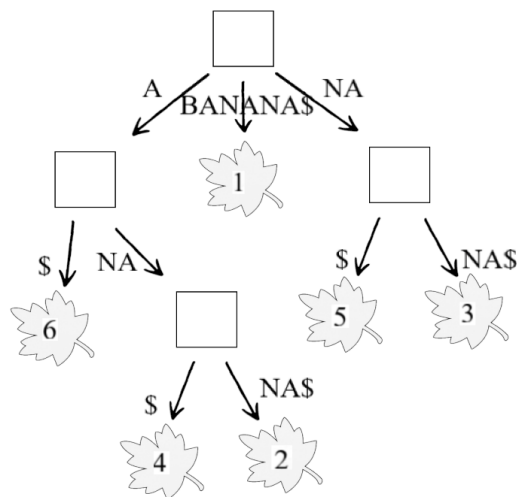


Figura 23: Árbol de sufijos para el texto BANANA. Cada subcadena termina con el carácter especial \$ que no aparece en la cadena (ver Fig. 23). Esto garantiza que ningún sufijo sea prefijo de otro, y que hayan n nodos de hoja, uno por cada uno de los n sufijos de S . Además, como los nodos internos -que no son raíz- son ramificados, pueden haber a lo más $n - 1$ nodos de este tipo, y $n + (n - 1) + 1 = 2n$ nodos en total, es decir, hay n hojas, a lo sumo $n - 1$ nodos internos y 1 raíz [100, 101].

Dado que no existe un árbol de este tipo para todas las cadenas, S se rellena con un símbolo terminal \$ que no aparece en la cadena (ver Fig. 23). Esto garantiza que ningún sufijo sea prefijo de otro, y que hayan n nodos de hoja, uno por cada uno de los n sufijos de S . Además, como los nodos internos -que no son raíz- son ramificados, pueden haber a lo más $n - 1$ nodos de este tipo, y $n + (n - 1) + 1 = 2n$ nodos en total, es decir, hay n hojas, a lo sumo $n - 1$ nodos internos y 1 raíz [100, 101].

Ahora, teniendo una idea de qué son los árboles de sufijos, ya se puede explicar el segundo modelo, el cual hace uso del paquete `suffix-trees` de Python. Este modelo utiliza

los árboles de sufijos para clasificar las secuencias de estructura secundaria en función de su relación con las familias introducidas en el capítulo III. Para esto, se le debe dar al modelo el archivo que contiene las estructuras secundarias junto con sus respectivas etiquetas, es decir, la familia a la que corresponde cada una de las estructuras. Luego, se crea un árbol de sufijos para cada familia -esto se hace para cada una de las tres diferentes representaciones- de los datos etiquetados. Debido a que un árbol de sufijos es una estructura que permite buscar de forma eficaz todos los sufijos de una cadena determinada, resulta útil aplicar esto para encontrar coincidencias o *patrones* de una estructura en los datos etiquetados y determinar a qué familia pertenece.

Después de construir los árboles de sufijos para cada familia teniendo en cuenta los datos etiquetados, se cargan los datos no etiquetados, es decir, las estructuras secundarias de los metagenomas. De esta forma, se va recorriendo cada estructura de los metagenomas y se buscan las coincidencias de la secuencia de la estructura en cada árbol de sufijos. El árbol de sufijos con más apariciones para dicha estructura se considera la familia a la que pertenece la secuencia, clasificando así cada una de las secuencias en una de las familias.

4.4.4. Resultados modelo II

Árbol ampliado y HIT: Al usar este modelo con estas dos representaciones, la clasificación que se obtuvo en ambos casos fue que ninguna de las secuencias de los metagenomas pertenecía a alguna de las familias. El archivo `csv` final mostraba que en la columna correspondiente a las predicciones no habían valores. Estos resultados se puede deber a múltiples factores:

1. Las secuencias del conjunto de datos metagenómicos o sin etiqueta no son representativas de las estructuras del conjunto de datos etiquetados. Esto significa que el conjunto de datos sin etiquetar puede contener estructuras secundarias que no coincidan con ninguna de las familias del conjunto de datos etiquetados.
2. Puede ocurrir que hayan limitaciones en la arquitectura del modelo.
3. Es posible que por el tipo de notación que estas dos estructuras tienen, no le sea posible al modelo encontrar semejanzas entre las estructuras de las secuencias metagenómicas y los árboles de cada familia, causando así la no clasificación.

De estos factores, el más plausible, considerando el modelo y los datos, resulta ser la opción 3. Al analizar el tipo de notaciones con las cuales se estaba haciendo la clasificación, la complejidad y variabilidad de los mismos dificulta la asociación entre las estructuras secundarias de los metagenomas y los árboles establecidos para cada familia, dando como consecuencia la no clasificación para los metagenomas. Debido a la forma en la que este tipo de algoritmos funcionan, resulta más complejo que se puedan encontrar coincidencias en las secuencias que correspondan o se puedan atribuir a alguno de los árboles generados para las familias *Arteriviridae*, *Orthomyxoviridae*, *Retroviridae*, *Sedoreoviridae* y *Spinareoviridae*.

Árbol de grano grueso: Para esta representación, el modelo clasificó 215988 secuencias de esta estructura secundaria en la familia *Retroviridae*, 187058 en la familia *Orthomyxoviridae*, 29788 se le atribuyeron a *Arteriviridae*, 49 quedaron con etiqueta de *Sedoreoviridae* y, finalmente, 22 se clasificaron como parte de la familia *Spinareoviridae* (ver Fig. 24). En este caso, quedaron sin clasificar 51837 estructuras de

los metagenomas o datos de prueba. Inicialmente los resultados de las métricas del modelo para los datos de prueba dieron que la exactitud, sensibilidad y puntuación F era de 0 y la precisión 1. Lo cual sugiere que en general no está haciendo un buen trabajo a la hora de clasificar los datos. Estos valores pueden deberse a que está ocurriendo algo similar a lo que pasó con las otras dos representaciones, y es que por la variabilidad de la notación en la estructura secundaria, no le es tan sencillo al modelo ajustar estas secuencias en un árbol asociado a una de las cinco familias.

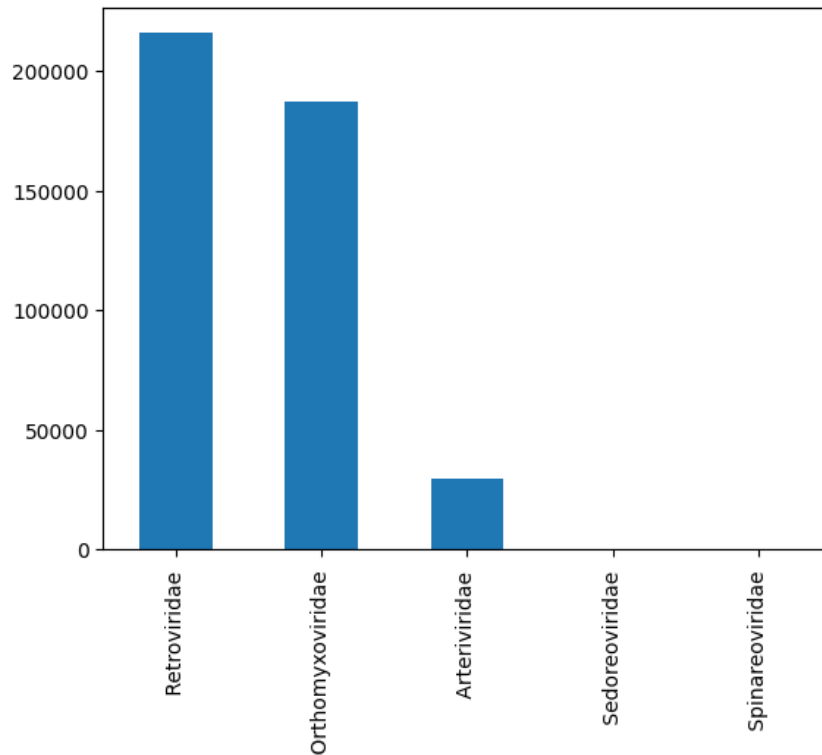


Figura 24: Histograma de los resultados obtenidos para la representación de árbol de grano grueso de los datos de prueba con el modelo II.

Se intentó mejorar el modelo haciendo algunos cambios como dividir los datos etiquetados en conjuntos de entrenamiento y de prueba. También se utilizó un parámetro de `ZeroDivisionError` que ayuda a establecer la precisión y la sensibilidad en cero para cualquier etiqueta que no tenga muestras predichas o verdaderas, respectivamente. Y por último se utilizó un método que convirtiera las etiquetas verdaderas y predichas en cadenas antes de pasarlas a las funciones de evaluación, para evitar problemas con las métricas. Al hacer estos cambios, sí hubo una mejora, ya que el valor para las métricas de exactitud, precisión, sensibilidad y puntuación F quedaron con un valor de 1, sin embargo, esto puede deberse a que los cambios ocasionaron un sobreajuste en el modelo.

Este segundo modelo por su naturaleza y el tipo de algoritmo, claramente presenta dificultades a la hora de extraer características significativas que ayuden a la clasificación de las representaciones de estructura secundaria. En este modelo, se hace evidente que el tipo de datos, es decir, su notación, complejizan el problema. En este caso, se podría reducir la

cantidad de árboles de sufijo para evitar el riesgo del sobreajuste, aunque esto ocasionaría otros problemas a la hora de la representación, lo cual no es ideal. Del mismo modo, otro inconveniente es que no se cuentan con suficientes datos para el entrenamiento, lo cual puede dificultar el proceso de aprendizaje para el modelo. Estos son aspectos útiles para tener en cuenta a la hora de trabajar con modelos basados en árboles de sufijos que además hacen uso de datos como los presentados en este trabajo.

4.4.5. Modelo III: Modelo oculto de Markov

Un modelo oculto de Markov (o HMM por sus siglas en inglés, Markov Hidden Models) es un modelo estadístico en el que se supone que el sistema a modelar es un proceso de Markov de parámetros desconocidos, esto es que un proceso X cuente con estados no observables u ocultos. Los modelos ocultos de Markov por definición requieren de que haya un proceso observable Y cuyos resultados estén influenciados por los resultados de X de forma conocida. Como no se puede observar X directamente, el objetivo es aprender sobre X examinando Y . Este modelo tiene un requisito adicional y es que el resultado de Y en $t = t_0$ debe estar influenciado exclusivamente por el resultados de X en $t = t_0$, y además los resultados de X e Y cuando $t < t_0$ deben ser condicionalmente independientes de Y en $t = t_0$ dado X en un tiempo $t = t_0$ [102, 103].

En otras palabras, un HMM consta de dos procesos estocásticos: un proceso invisible de estados ocultos y un proceso visible de símbolos observables. Los estados ocultos forman una cadena de Markov, y la distribución de probabilidad del símbolo observado depende del estado subyacente. Esto último es lo que hace al modelo tan útil, ya que se modelan observaciones en dos capas, una visible y otra invisible, y muchos problemas del mundo real tratan de clasificar las observaciones originales en una serie de categorías, o etiquetas de clase, que resultan más significativas dependiendo del problema [104].

Más formalmente, un HMM es una tupla $M = (Q, \Sigma, A, s, E)$ donde [102, 104, 105]:

- $Q = \{q_1, \dots, q_n\}$ corresponde a un conjunto finito de estados.
- $\Sigma = \{\sigma_1, \dots, \sigma_m\}$ es el alfabeto, es decir, el conjunto de símbolos (o palabras) de salida.
- $A = \{a_{ij}\}$ es una matriz $n \times n$ de probabilidades de transición, es decir, es el conjunto de probabilidades de transiciones entre estados. Así, $a_{ij} = P[q_{t+1} = j \mid q_t = i]$, o de otra forma, a_{ij} es la probabilidad de estar en el estado j en el instante $t + 1$ si en el instante anterior t se estaba en el estado i .
- $s = \{s_1, \dots, s_n\}$ es un vector de probabilidades de inicio, donde s_i es la probabilidad de que el primer estado sea el estado q_i . Eso es, $s_i = P[q_0 = i]$.
- $E = \{e_{ij}\}$ es una matriz $n \times m$ de probabilidades de emisión, es decir, es el conjunto de probabilidades de las observaciones. De otra manera, $e_{ij} = P[o_t = \sigma_j \mid q_t = i]$, así e_{ij} es la probabilidad de observar σ_j cuando se está en el estado j en el instante t .

Hay tres tipos de modelos ocultos de Markov: de izquierda a derecha, totalmente conectado y ergódico [106]. En un HMM de **izquierda a derecha**, las transiciones de estado

sólo van de izquierda a derecha, lo que significa que cada estado sólo puede transicionar hacia sí mismo o hacia los estados que le siguen, además, las probabilidades de transición del estado i al estado $i + 1$ suelen ser altas, mientras que las probabilidades de transición del estado i a los estados $i + 2$, $i + 3$, etc., suelen ser bajas o nulas. Este tipo de modelo es adecuado para modelar secuencias que tienen un orden temporal, como el habla o la música. En un HMM **totalmente conectado**, cada estado puede transicionar a cualquier otro estado, incluido el mismo, además en estos las probabilidades de transición entre dos estados cualesquiera pueden ser distintas de cero y suelen aprenderse a partir de los datos de entrenamiento. Este tipo de modelo es adecuado para modelar secuencias en las que cualquier símbolo puede aparecer en cualquier momento, como secuencias de texto o ADN/ARN. Y finalmente, un HMM **ergódico** es un tipo más general de HMM en el que cualquier estado puede transicionar a cualquier otro estado, incluido él mismo. Esto significa que el modelo puede tener ciclos, lo que le permite capturar patrones más complejos en los datos de secuencia, sin embargo, estimar las probabilidades de transición en un modelo de tipo ergódico es más difícil comparado a los otros dos, ya que hay más parámetros que estimar [99, 106].

El tipo de modelo que se implementó fue del tipo totalmente conectado. Este tercer modelo utiliza el modelo probabilístico para predecir la familia de cada una de las secuencias de los datos metagenómicos utilizando las diferentes representaciones de estructura secundaria. En este caso, el HMM es un MultinomialHMM que se utiliza para modelar secuencias de símbolos discretos, donde cada símbolo tiene una cierta probabilidad de ocurrir en cada estado, es decir, el modelo supone que la secuencia de estados subyacente es un proceso de Markov, lo que significa que la probabilidad de transición a un nuevo estado sólo depende del estado actual y no de la historia previa de estados. Por lo tanto, el MultinomialHMM supone que las observaciones resultantes por el modelo son generadas por un proceso de Markov oculto, en el que los estados ocultos corresponden a la secuencia subyacente de eventos que generaron las observaciones, y las probabilidades de emisión para cada estado vienen dadas por una distribución multinomial.

El modelo se entrena con los datos que tienen la etiqueta para cada secuencia -los de la base estándar que se elaboró- utilizando los datos de la secuencia convertidos en recuentos de símbolos mediante `CountVectorizer`. Durante el entrenamiento, el modelo aprende los parámetros del HMM, incluida la distribución inicial de estados, las probabilidades de transición de estados y las probabilidades de emisión, es decir, las probabilidades de observar cada símbolo en cada estado. Luego de esto, el modelo entrenado se usa para clasificar los datos metagenómicos mediante los datos de secuencia convertidos en recuentos de símbolos por medio del mismo `CountVectorizer`. El modelo aplica el *algoritmo de Viterbi*¹⁹ para encontrar la secuencia más probable de estados ocultos que generó la secuencia observada de símbolos en los datos de prueba. De esta forma, se realiza la clasificación y asignación de familia para cada una de las secuencias de estructura secundaria (esto se realiza para cada una de las representaciones) de los metagenomas.

4.4.6. Resultados modelo III

Árbol ampliado: Para esta representación de estructura secundaria, el modelo clasificó las secuencias de datos metagenómicos como sigue: 182765 secuencias corresponden

¹⁹Ver apéndice H.

a *Orthomyxoviridae*, 215677 se asignaron a *Arteriviridae*, 77138 a *Spinareoviridae*, 4595 a *Sedoreoviridae* y, finalmente, 4567 se clasificaron como *Retroviridae* (ver Fig. 25). En el reporte de clasificación, el modelo parece haber funcionado muy bien con los datos de prueba, se obtuvo una exactitud, sensibilidad y puntuación F de 1 para cada una de las cinco clases: *Arteriviridae*, *Orthomyxoviridae*, *Retroviridae*, *Sedoreoviridae* y *Spinareoviridae*. La precisión del modelo también es de 1, lo que indica que el modelo fue capaz de clasificar correctamente todas las muestras del conjunto de datos. Además, se revisó el `macro avg` y el `weighted avg`, estas son métricas adicionales que proporcionan un promedio de la precisión, la sensibilidad y la puntuación F en todas las clases del conjunto de datos. En este caso, ambas métricas obtienen puntuaciones de 1, lo que indica que el modelo ha funcionado bien para cada una de las clases. En general, este informe sugiere que el modelo fue capaz de clasificar con precisión las muestras del conjunto de datos y que su rendimiento es bueno.

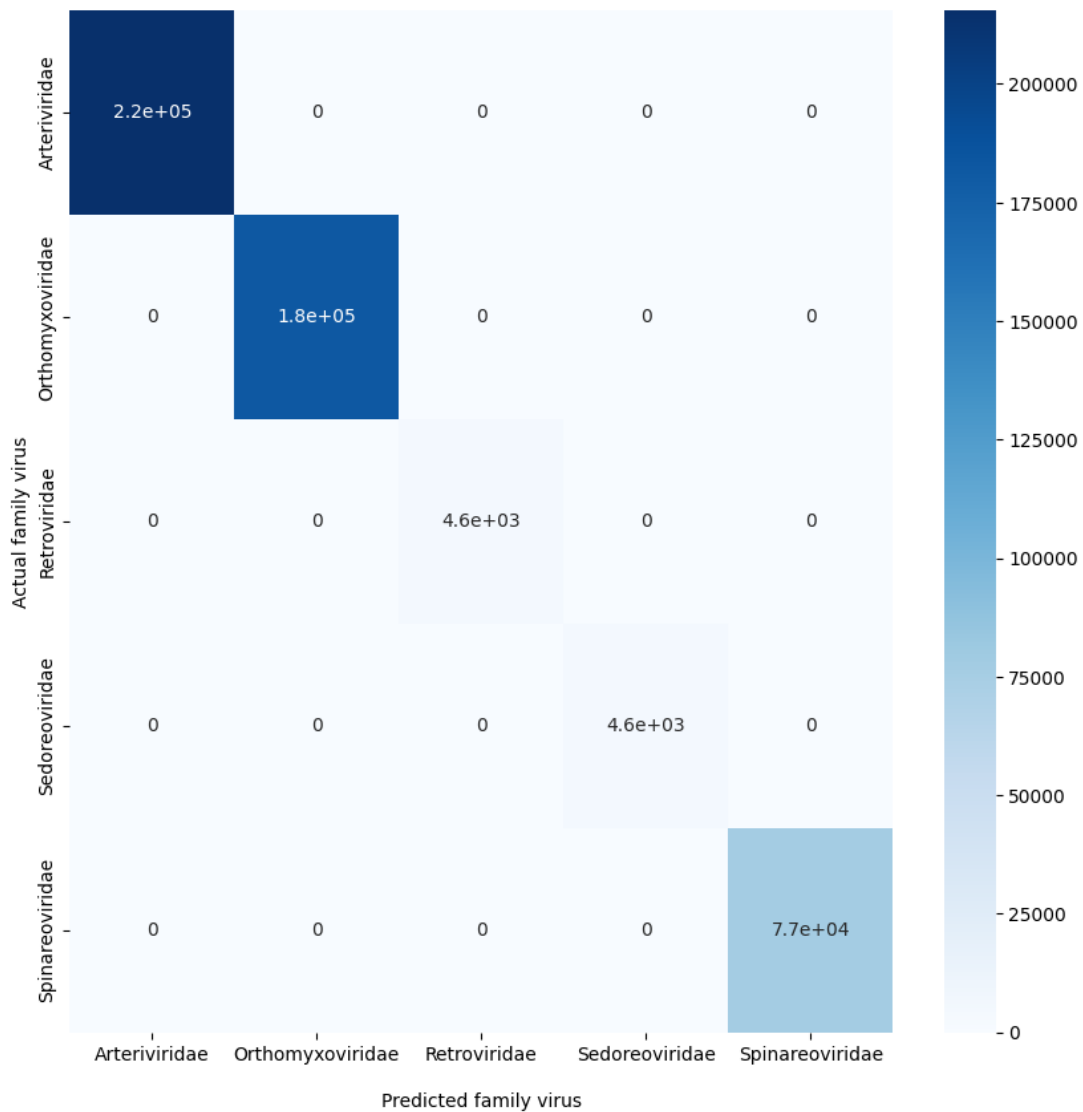


Figura 25: Matriz de confusión de los resultados obtenidos para la representación de árbol ampliado de los datos de prueba con el modelo III.

HIT: Para esta notación de estructura secundaria los resultados en la clasificación muestran una mejor distribución, es decir, se clasificaron 132773 secuencias de estructura como *Retroviridae*, 157221 secuencias como *Spinareoviridae*, a 117322 se les asignó la familia *Sedoreoviridae*, a 39859 la familia *Arteriviridae* y, finalmente, 37567 secuencias quedaron con la etiqueta de *Orthomyxoviridae* (ver Fig. 26).

Al probar el modelo con los datos de prueba, se obtuvo una exactitud, sensibilidad, puntuación F y precisión de 1 para cada una de las cinco clases: *Arteriviridae*, *Orthomyxoviridae*, *Retroviridae*, *Sedoreoviridae* y *Spinareoviridae*, indicando que el modelo fue capaz de clasificar correctamente todas las muestras del conjunto de datos. Además, las métricas de `macro avg` y el `weighted avg` tienen puntuaciones de 1, lo que indica que el modelo ha clasificado bien para cada una de las clases.

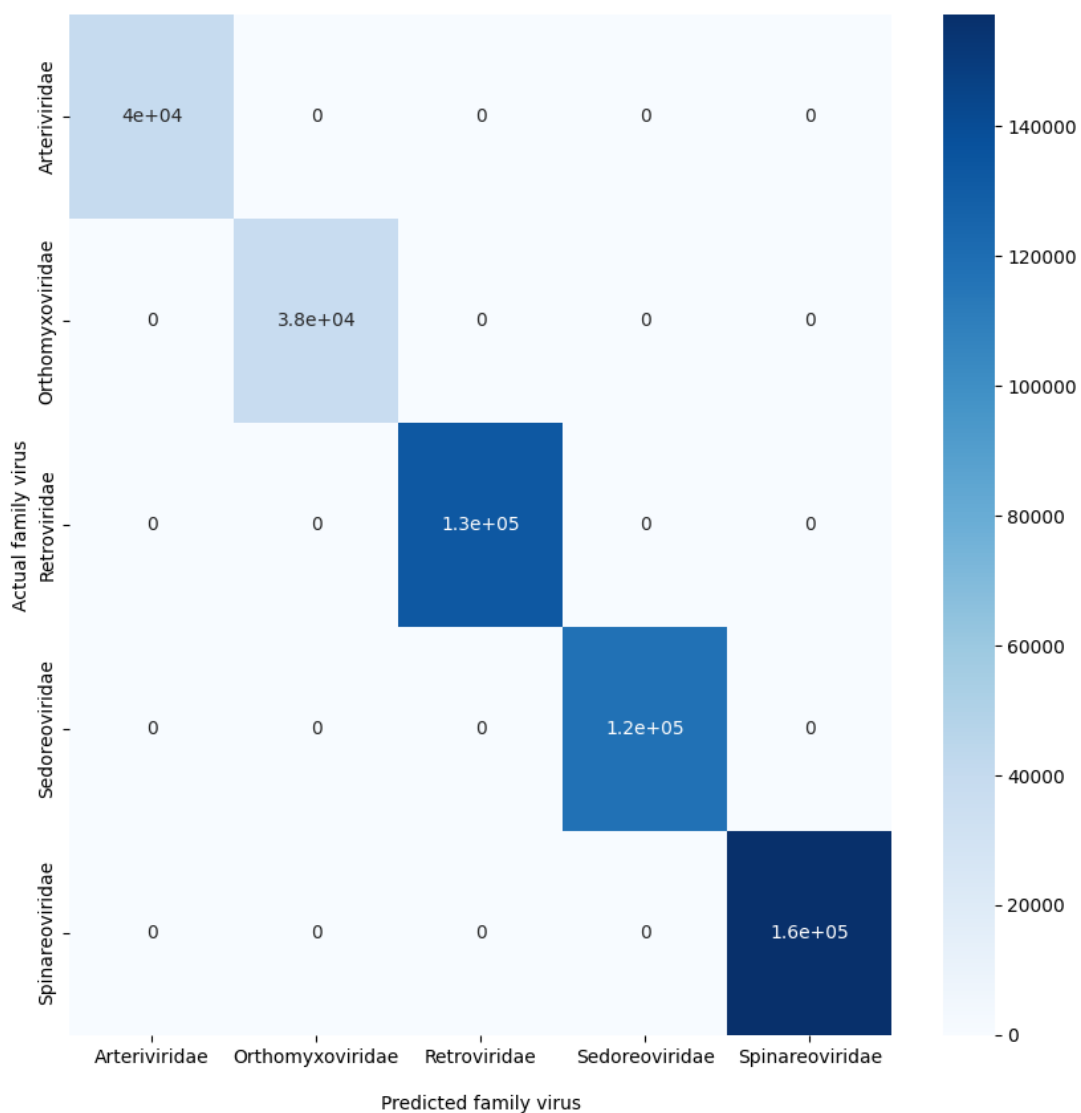


Figura 26: Matriz de confusión de los resultados obtenidos para la representación HIT de los datos de prueba con el modelo III.

Árbol de grano grueso: El modelo clasificó las secuencias metagenómicas de la siguien-

te manera: la familia *Sedoreoviridae* quedó con 398014 secuencias, 29705 se asignaron a *Spinareoviridae*, 31872 a *Retroviridae*, 12736 a *Arteriviridae* y, 12415 se clasificaron como *Orthomyxoviridae* (ver Fig. 27).

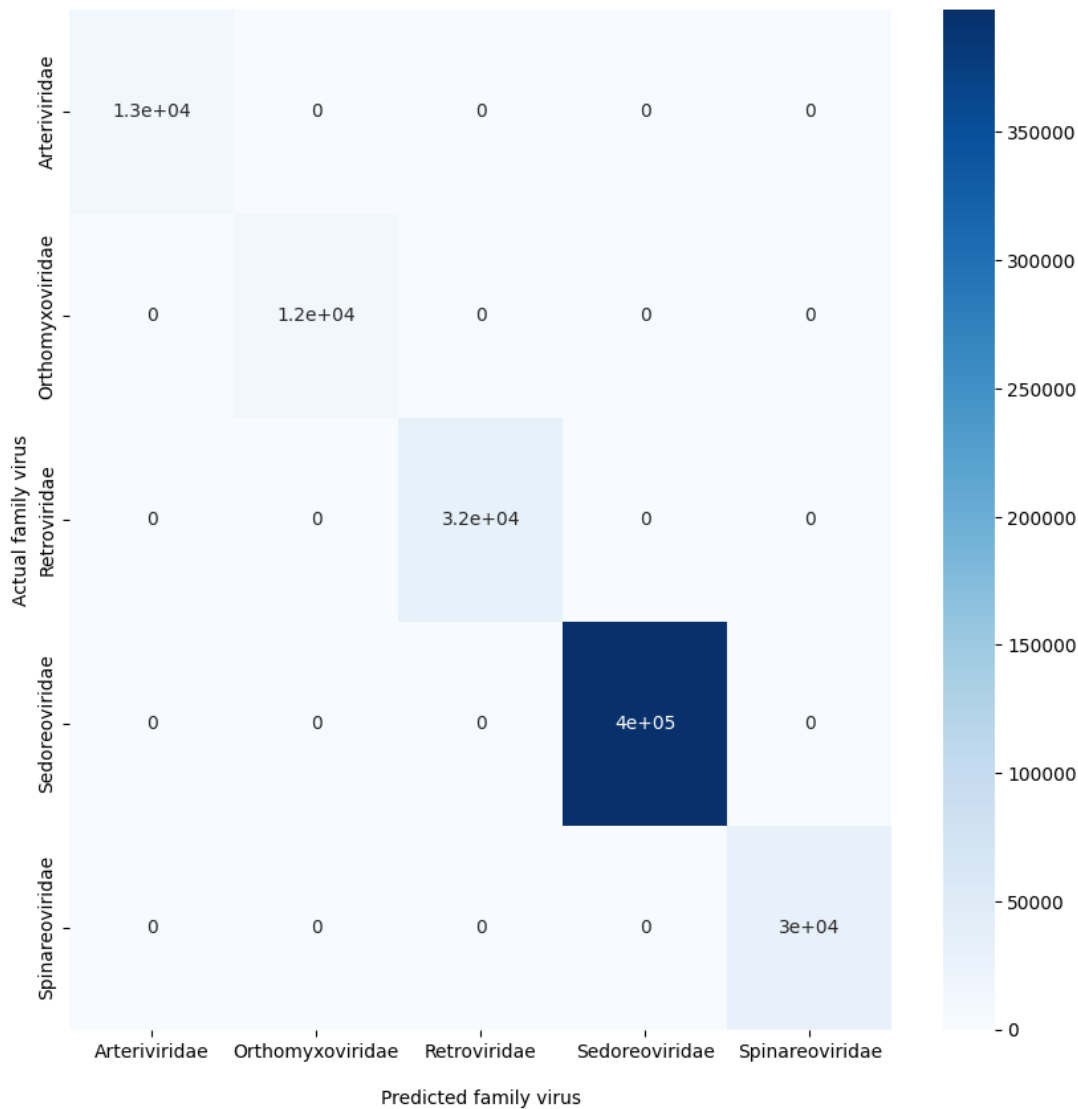


Figura 27: Matriz de confusión de los resultados obtenidos para la representación de árbol de *grano grueso* de los datos de prueba con el modelo III.

Esta representación dio como resultado una clasificación poco proporcional, la distribución que tienen estos datos es bastante asimétrica, además toca tener en cuenta que la familia que quedó con más del 80% de etiquetas, corresponde a una de las familias originales que tenía menos de 1% de secuencias en el conjunto de entrenamiento, es decir, era de las que menos representaciones de estructura tenía. Los resultados para las métricas de la prueba del modelo fueron iguales a los valores obtenidos para las dos representaciones previas. En teoría indicando que el modelo ha hecho un buen trabajo clasificando las clases.

Los resultados presentados para este tercer modelo se resumen en la Figura 28. A pesar de que los valores para las métricas de desempeño fueron buenas para todas las estructuras,

es necesario analizar estos resultados dentro de un esquema más amplio que considere factores como la notación de cada estructura, la cantidad de datos de entrenamiento y el modelo como tal. En el caso de las representaciones, es necesario considerar la estructura, es decir, del capítulo 2 se sabe que la estructura secundaria en el caso de la representación de árbol ampliado, incluye tres etiquetas adicionales a las usuales de punto y corchete, en la representación HIT se añaden las mismas etiquetas de árbol ampliado y además, se agregan los pesos, y en la representación de árbol de *grano grueso* hay siete etiquetas adicionales.

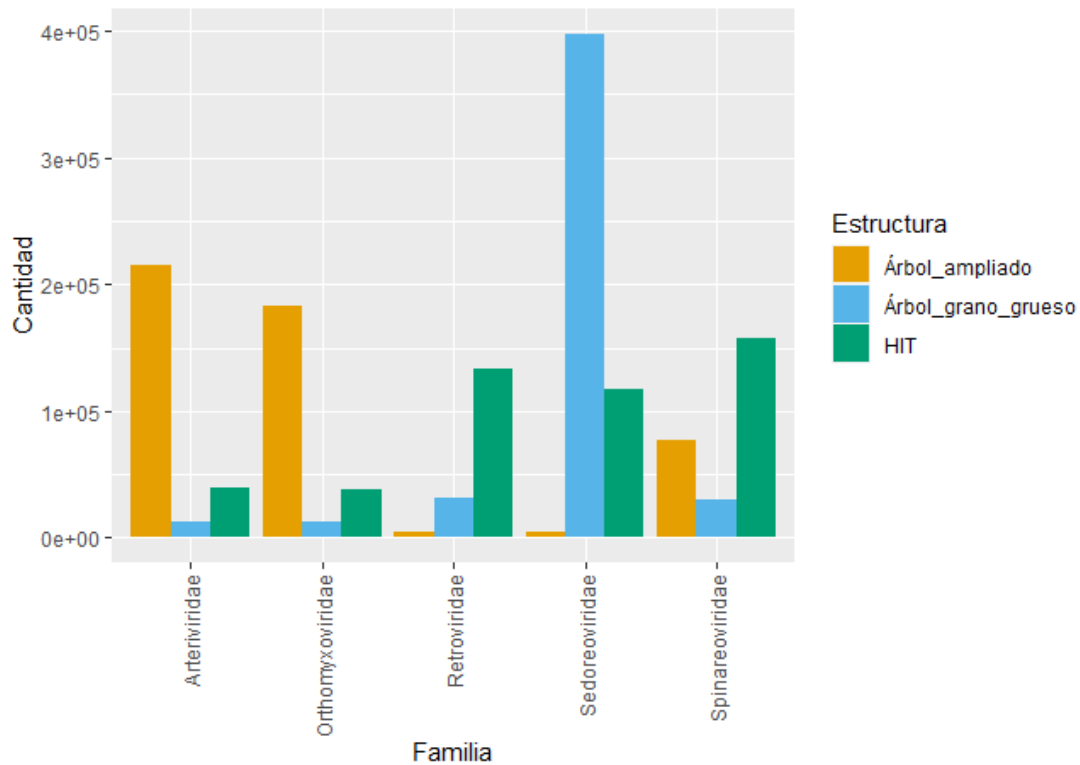


Figura 28: Histograma con los resultados de las tres estructuras para el modelo III.

La representación de árbol de *grano grueso* aparenta tener la mayor cantidad de etiquetas, sin embargo esto no es completamente cierto, por un lado esta representación no conserva toda la información de la estructura, ya que las nuevas etiquetas representan las sub-estructuras de la estructura original y, por otro lado, la representación HIT al tener como etiqueta adicional los pesos, cada peso sería una nueva etiqueta, que puede variar muchísimo más dependiendo de la cantidad de nucleótidos apareados y no apareados que hayan en la secuencia. Por ende, la especificidad de la estructura HIT permite encontrar mejores relaciones entre las secuencias metagenómicas y las etiquetas de las secuencias usadas en el entrenamiento del modelo.

Si bien los resultados fueron prometedores, es necesario tener en cuenta que es desequilibrio presente entre los datos de entrenamiento y prueba puede mejorar. Es posible que si se aumenta la cantidad de datos con los cuales se entrena el modelo, habrían mejores resultados a nivel general en la clasificación para las tres representaciones. Finalmente, la

construcción del modelo está basada en que cada símbolo o etiqueta nueva dentro de cada representación tiene una cierta probabilidad de ocurrir en cada estado, entonces, puede ser que al tener mayor variabilidad de las etiquetas, sea más precisa la forma en la que el modelo realiza las clasificaciones de las estructuras. De este modo, en este modelo la estructura HIT es la que mejores características aporta en el proceso de clasificación.

4.4.7. Modelo IV: CNN-LSTM

Este algoritmo particular se escogió porque se ha visto en trabajos como [131] y [132] el buen desempeño que tienen estos modelos con problemas similares al de este trabajo. Para entender qué es y cómo funciona de forma general una red neuronal convolucional con memoria a largo y corto plazo o CNN-LSTM, es necesario explicar que son las redes neuronales convoluciones y las LSTM. En principio una red neuronal convolucional (o CNN por sus siglas en inglés: convolutional neural network) es una clase de red neuronal que utiliza una operación matemática llamada *convolución*²⁰ en al menos una de sus capas [109]. Estas redes son un perceptrón multicapa diseñado específicamente para reconocer formas bidimensionales con un alto grado de invariabilidad a la traslación, la escala, la inclinación y otras formas de distorsión [97]. tienen la capacidad de reconocer patrones en los datos de entrada, independientemente de su ubicación, mediante la aplicación de filtros de convolución, además usan el mismo filtro para procesar diferentes partes de la entrada, lo cual sirve para extraer características importantes [110, 111].

Las CNN son versiones mejoradas de los perceptrones multicapa, es decir, en los perceptrones multicapa cada neurona de una capa está conectada a todas las neuronas de la siguiente capa, haciéndolas propensas a sobreajustar los datos. Mientras que en las CNN se aprovecha el patrón jerárquico de los datos para armar patrones cada vez más complejos utilizando patrones más pequeños y sencillos, esto permite a las CNN aprender de forma eficiente patrones complejos en los datos, mientras que minimiza el riesgo de sobreajuste [81, 110]. Una red neuronal convolucional consta de una capa de entrada, capas ocultas y una capa de salida. Las capas ocultas incluyen una o más capas que realizan convoluciones. A medida que el kernel de convolución²¹ se mueve a lo largo de la matriz de entrada de la capa, la operación de convolución genera un *mapa de características*, que a su vez contribuye a la entrada de la siguiente capa (ver Fig. 42). A esta le siguen otras capas, como las capas de agrupamiento o *pooling*, las capas totalmente conectadas y las capas de normalización.

- **Capa convolucional:** La capa de convolución es la base de las CNN, ya que conlleva la mayor parte de la carga computacional de la red. Esta capa realiza un producto punto entre dos matrices, una de las cuales es el conjunto de parámetros aprendibles, también conocido como kernel, y la otra es la parte restringida del campo receptivo [81, 110].
- **Capa de agrupamiento:** En esta capa se sustituye la salida de la red en determinados lugares obteniendo una estadística resumida de las salidas cercanas. Dicho de otra forma, se reducen las dimensiones de los datos combinando las salidas de los grupos de neuronas de una capa en una sola neurona en la siguiente capa, lo que disminuye la cantidad necesaria de cálculos y pesos [81, 110].

²⁰Ver apéndice I.

²¹El kernel de convolución es un filtro que se utiliza para realizar la operación de convolución.

- **Capa completamente conectada:** En estas capas las neuronas se conectan completamente con todas las neuronas de la capa anterior y posterior.
- **Capa de normalización:** Estas capas de normalización, también llamadas capas de no linealidad suelen colocarse directamente después de la capa convolucional para introducir la no linealidad en el mapa de activación. Para este propósito se suelen usar funciones como la sigmoide, que toma un número de valor real y lo ubica en un intervalo entre 0 y 1, la función tangente hiperbólica, que fuerza a un número de valor real en el intervalo $[-1, 1]$, o la función ReLU, que es computacionalmente más eficiente que la función sigmoide o tanh, que implican costosos cálculos exponenciales [109, 111].

Las redes neuronales convolucionales se han utilizado ampliamente en tareas de visión por ordenador como la clasificación de imágenes, y han mostrado su efectividad a la hora de extraer de características espaciales de imágenes utilizando filtros convolucionales y preservando la información importante mediante la agrupación de capas. Sin embargo, cuando se trata de tareas en las que intervienen datos secuenciales, las limitaciones de las CNN se hacen evidentes ya que este tipo de datos requieren modelos que puedan captar las dependencias a lo largo del tiempo. Los datos secuenciales se refieren a cualquier tipo de datos en los que el orden o la secuencia de los elementos son significativos y contienen información valiosa, por ejemplo: datos de series temporales, texto en lenguaje natural, secuencias de ADN y ARN. Por ende, es aquí cuando la memoria de largo y corto plazo (LSTM) juega un rol importante, ya que están diseñadas específicamente para manejar datos secuenciales. De esta forma, los modelos híbridos que combinan CNN y LSTM resultan ventajosos porque estos utilizan una CNN para extraer características locales de palabras individuales, mientras que la LSTM captura las dependencias secuenciales entre palabras, lo que da como resultado un sistema de análisis completo.

La memoria de largo y corto plazo (LSTM), es una red neuronal artificial utilizada en los campos de la inteligencia artificial y el aprendizaje profundo. A diferencia de otras redes neuronales, la LSTM tiene conexiones de retroalimentación. Este tipo de red neuronal recurrente puede procesar no solo puntos de datos individuales, sino también secuencias enteras de datos; esta característica hace que las redes LSTM sean ideales para procesar y predecir datos [112]. Una unidad LSTM común se compone de una celda, una puerta de entrada, una puerta de salida y una puerta de olvido [113, 114]. La celda recuerda valores a lo largo de intervalos de tiempo arbitrarios y las tres puertas regulan el flujo de información que entra y sale de la celda. Las puertas de olvido deciden qué información descartar de un estado anterior asignando a un estado previo un valor entre 0 y 1. Un valor de 1 significa conservar la información, y un valor de 0 significa descartarla. Las puertas de entrada deciden qué fragmentos de información nueva se almacenan en el estado actual, utilizando el mismo sistema de las puertas de olvido. Las puertas de salida controlan qué piezas de información del estado actual hay que emitir, asignando un valor de 0 a 1 a la información, teniendo en cuenta los estados anteriores y el actual [112, 113, 114].

Por lo tanto, en un modelo CNN-LSTM los datos de entrada pasan primero por una o varias capas convolucionales, que extraen características espaciales de los datos. Luego, la salida de estas capas convolucionales pasa a una capa LSTM, que procesa la infor-

mación temporal de la secuencia. La capa LSTM puede aprender a recordar u olvidar selectivamente información a lo largo del tiempo utilizando sus celdas de memoria, lo que le permite captar dependencias a largo plazo en la secuencia de entrada. Finalmente, la salida de la capa LSTM pasa por una o varias capas totalmente conectadas para producir la salida final [115].

En este código es necesario hacer algo similar a lo que se hizo en el modelo I, y es crear un objeto `LabelEncoder`, que sirve para codificar etiquetas categóricas con una representación numérica. Este paso se realiza para que a cada categoría o familia se le asigne un valor entero único, lo que permite utilizar esta información en los algoritmos posteriores que requieren una entrada numérica. Los datos de entrenamiento se separan en dos conjuntos, uno con el cual se va a entrar el modelo, que corresponde al 80% de los datos, y el otro 20% se utiliza para validar. Antes de continuar y definir la arquitectura del modelo, es necesario utilizar la clase `Tokenizer` para codificar como matrices de enteros las secuencias de estructura secundaria, esto se consigue haciendo que cada carácter en la secuencia de entrada sea tratado como un token separado.

Se ejecuta el método `fit_on_texts()` del objeto tokenizador en los datos de entrenamiento, que crea un vocabulario de todos los caracteres únicos de los datos de texto y asigna un número entero único a cada carácter. Después de ajustar el tokenizador a los datos de entrenamiento, se ejecuta el método `texts_to_sequences()` del objeto tokenizador en los datos de entrenamiento y de prueba, para convertir cada secuencia de estructura secundaria en una secuencia de enteros utilizando el vocabulario aprendido durante el ajuste. Todo este proceso previo es necesario porque ese tipo de modelos sólo pueden trabajar con datos numéricos. Al codificar las secuencias de estructura secundaria como matrices de enteros, el modelo puede aprender patrones y relaciones significativas entre las secuencias de entrada y sus etiquetas correspondientes.

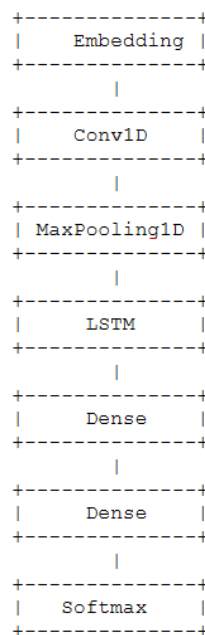


Figura 29: Estructura del modelo IV.

Debido a que la longitud de las secuencias varía, se rellenan las secuencias con una longitud fija, esta longitud se escoge teniendo en cuenta la longitud máxima de las secuencias que hay tanto en el conjunto de datos con etiquetas, como en el conjunto de datos metagenómicos. El modelo IV tiene la estructura mostrada en la Figura 29, cuya entrada es un arreglo, en la que cada fila representa una secuencia de ARN con una longitud fija y cada elemento de la fila representa la codificación entera de un carácter de ARN.

1. **Capa de *embedding***: En esta capa se convierten las secuencias de entrada (las entradas se representan como índices enteros) en vectores densos de tamaño fijo. Además, se fija un parámetro que corresponde al tamaño del vocabulario, es decir, al número de palabras únicas en el conjunto de entrenamiento más uno, esto se hace para tener en cuenta cualquier palabra fuera del vocabulario. Y se fija la longitud máxima de cualquier secuencia de entrada.
2. **Capa Conv1D**: En esta capa se aplica una operación de convolución 1D a la secuencia de entrada, esta es una operación matemática que consiste en pasar una ventana pequeña o kernel a lo largo de una señal de entrada unidimensional y calcular el producto punto entre el kernel y los valores de la ventana en cada posición. El resultado de hacer esto es una nueva señal unidimensional, con la misma longitud que la de entrada, que representa la presencia de determinadas características o patrones en la señal original. Además, se establecen otros parámetros como el de: **filters** en 32, **kernel_size** en 3, **padding** se establece en `same` (esto hace que se añada relleno a la secuencia de entrada para preservar su longitud), y se determina la función de activación ReLU.
3. **Capa MaxPooling1D**: Esta capa calcula una función **MaxPooling** en la salida de la capa convolucional, es decir, realiza un *down-sampling* o sub-muestreo en la salida de la capa convolucional. El parámetro **pool_size** se establece en 2, lo que significa que la capa tomará el valor máximo dentro de cada dos valores adyacentes en la salida de la capa convolucional. Este proceso ayuda a reducir el tamaño de la salida, conservando al mismo tiempo las características más importantes.
4. **Capa LSTM**: Esta capa añade una capa LSTM con 100 unidades de memoria. Cada unidad de memoria LSTM puede recordar u olvidar selectivamente datos de entrada anteriores en función de la entrada actual y la salida anterior.
5. **Capa densa**: Esta es una capa totalmente conectada con 50 unidades y activación ReLU.
6. **Capa densa**: Esta capa añade una última capa totalmente conectada con 5 unidades, que corresponde al número de clases únicas de la tarea de clasificación, es decir, las cinco familias. En esta la función de activación utilizada es softmax, que convierte los valores de salida en probabilidades.

Luego de tener la estructura del modelo, este se pasa a compilar con `sparse_categorical_crossentropy` como función de pérdida, esta es una función utilizada en problemas de clasificación multiclase; es similar a `categorical_crossentropy`, pero está diseñada para utilizarse cuando las clases son enteras en lugar de vectores codificados de un solo

punto. Como optimizador se utilizó adam y la exactitud como métrica de evaluación. El entrenamiento del modelo se llevó a cabo con los datos de entrenamiento y validación para, finalmente, usar este modelo entrenado en la clasificación de las etiquetas de las secuencias de estructura secundaria de ARN no etiquetadas. A continuación se detallan los resultados obtenidos para cada una de las diferentes representaciones.

4.4.8. Resultados modelo IV

Árbol ampliado: Para esta representación de estructura secundaria los resultados de clasificación se dieron entrenando el modelo por 100 épocas y un tamaño de lote de 32; estos valores se escogieron luego de cambiar dichos parámetros múltiples veces, ya que se notó que al usar menos tiempo de entrenamiento el modelo tenía dificultades aprendiendo características y, al aumentar empezaba a sobreajustarse a los datos.

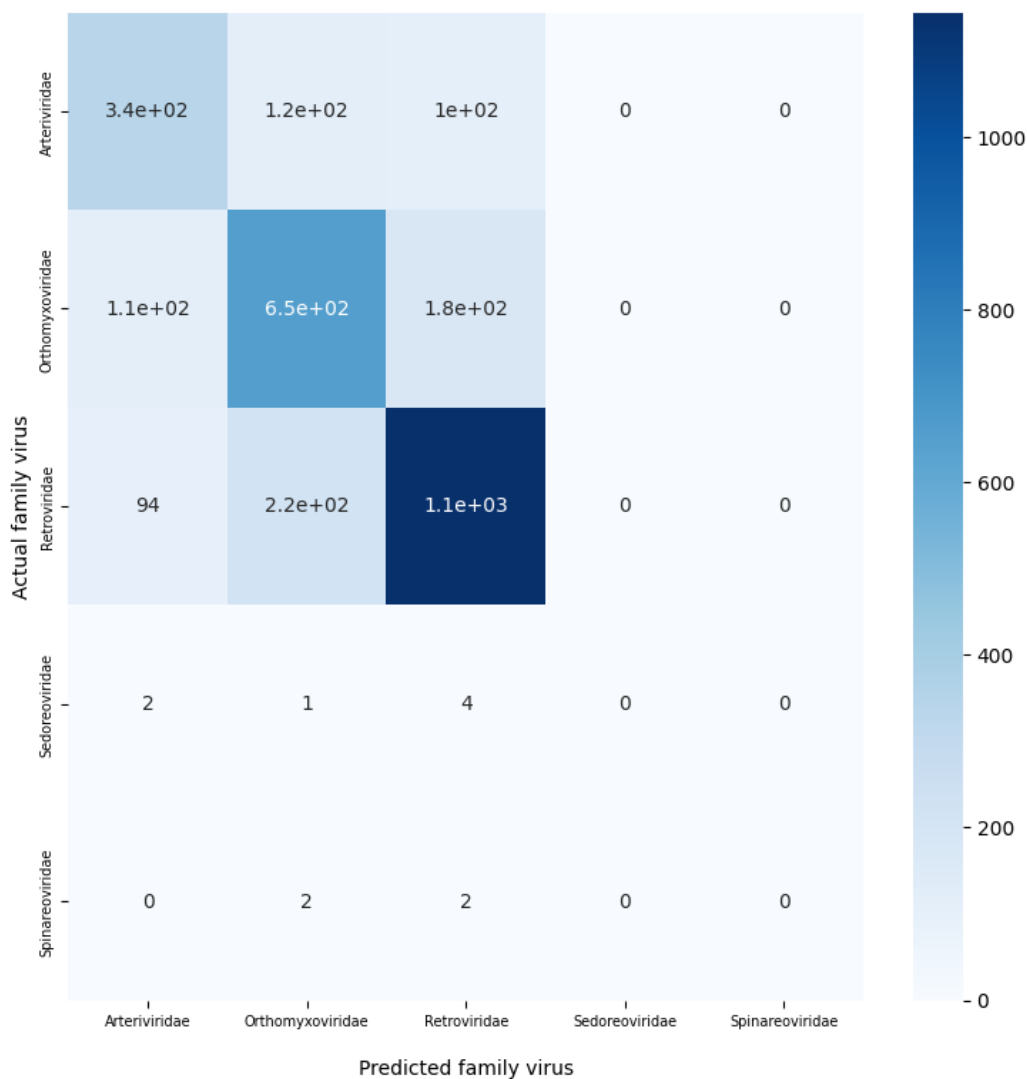


Figura 30: Matriz de confusión con los resultados para la estructura de árbol ampliado bajo el modelo IV con los datos de prueba.

En este caso la clasificación de los datos metagenómicos arrojó los siguientes re-

sultados: la familia *Orthomyxoviridae* quedó con 324876 secuencias, *Retroviridae* con 100461, *Arteriviridae* con 59404, *Spinareoviridae* con 1, y *Sedoreoviridae* con 0 (ver Fig. 30). El modelo alcanzó una exactitud del 84.46% en los datos de entrenamiento y del 72.09% en los datos de validación. Esto indica que el modelo clasifica correctamente las muestras. En cuanto a la prueba del modelo con los datos metagenómicos, se obtuvo una exactitud de 0.7208, una precisión del 0.7195, una sensibilidad de 0.7208 y una puntuación F de 0.72, de este modo, el modelo clasifica correctamente cerca del 72.1% de las muestras, en términos de precisión, cuando predice una clase positiva, acierta alrededor del 71.95% de las veces, e identifica alrededor del 72.1% de las muestras positivas. Los resultados indican que el modelo hace un buen trabajo, aunque se puede mejorar ya sea cambiando más los parámetros y aumentando la cantidad de datos del conjunto de entrenamiento.

HIT: El modelo se entrenó por 75 épocas y un tamaño de lote de 32, ya que al aumentar la cantidad de épocas disminuía drásticamente el desempeño de la prueba del modelo.

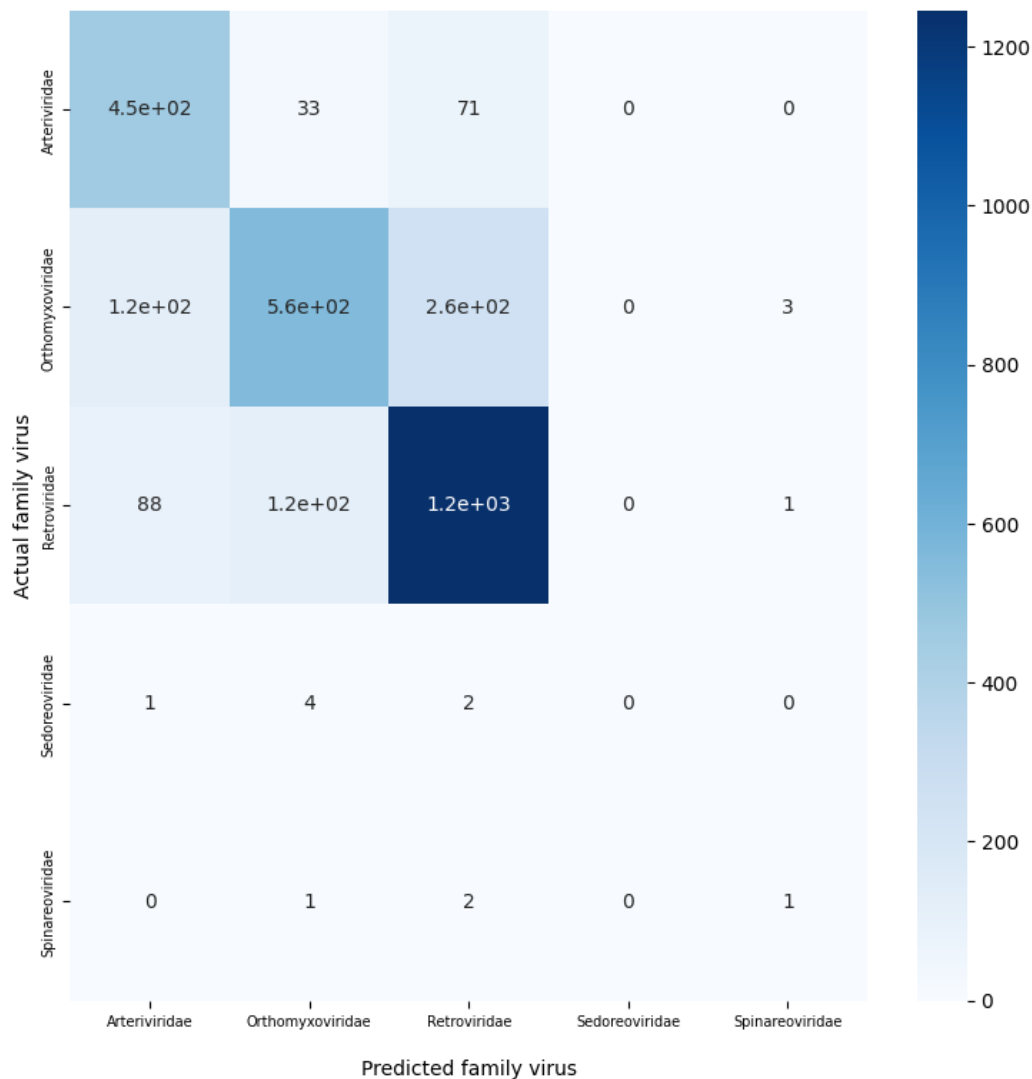


Figura 31: Matriz de confusión con los resultados para la estructura HIT de los datos de prueba bajo el modelo IV.

La clasificación de los datos metagenómicos usando esta estructura dio que 225195 secuencias pertenecen a la familia *Orthomyxoviridae*, 186913 a *Retroviridae*, 71029 a *Arteriviridae*, 1430 a *Spinareoviridae* y, 175 a *Sedoreoviridae* (ver Fig. 31). El modelo se entrenó durante 75 épocas y alcanzó una exactitud de 0.9722 y una pérdida de 0.0870 en el conjunto de entrenamiento, lo que da a entender que el modelo es capaz de aprender de forma correcta los patrones de los datos de entrenamiento. Al probar el modelo con los datos metagenómicos o de prueba, se consigue 0.7624 en exactitud, 0.7639 en precisión, 0.7624 en sensibilidad y 0.7573 en la puntuación F. Según estos resultados, el modelo parece funcionar bien, con una buena precisión y un equilibrio razonable entre precisión y sensibilidad.

Árbol de grano grueso: Para esta representación se entrenó el modelo por 75 épocas y con lotes de tamaño 32, al aumentar la cantidad de épocas de entrenamiento el desempeño del modelo no mejoraba más de un 1% respecto a los valores obtenidos cuando se entrenaba por 75 épocas, por ende se decidió dejar este valor para las épocas y así evitar problemas de sobreajuste.

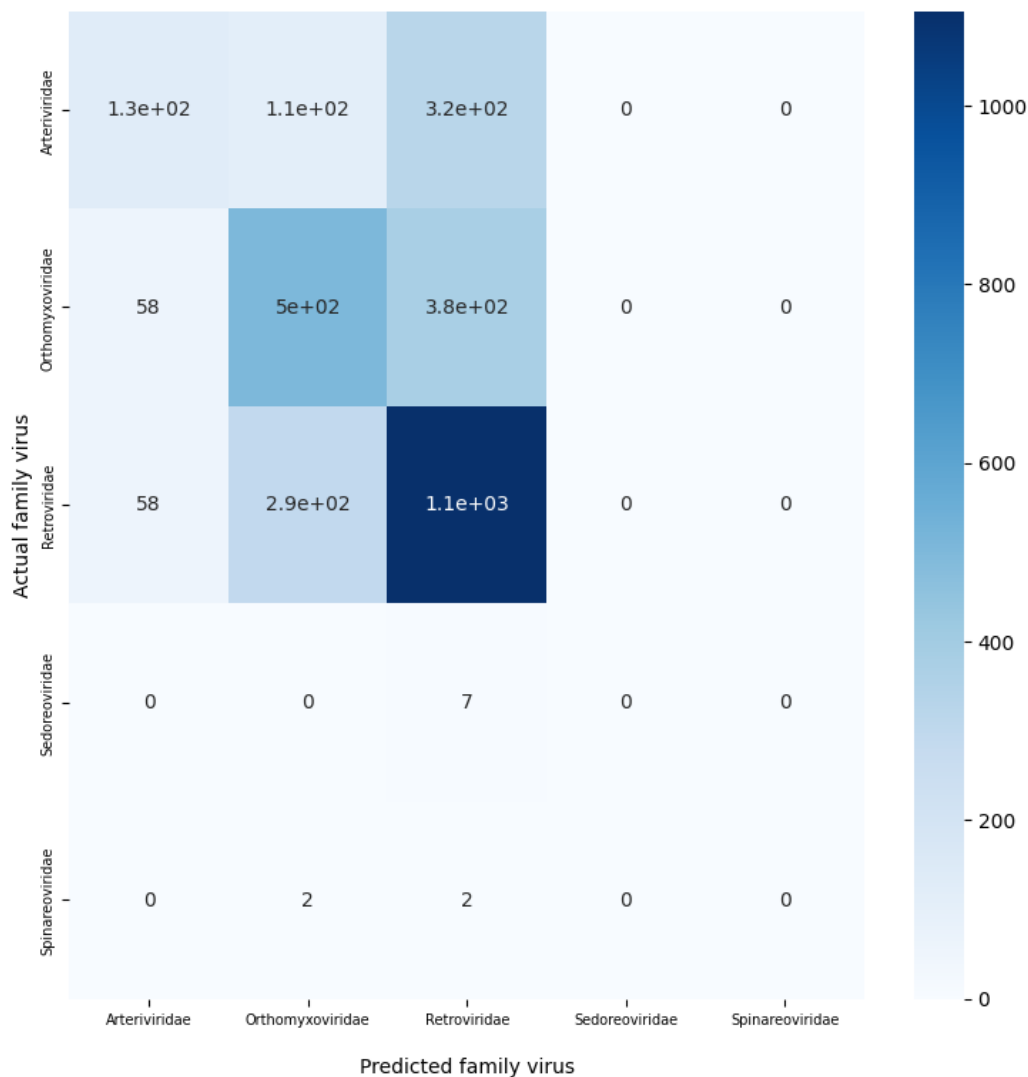


Figura 32: Matriz de confusión con los resultados para la estructura de árbol de *grano grueso* bajo el modelo IV.

Se obtuvieron 240021 estructuras clasificadas como *Retroviridae*, 220299 como *Orthomyxoviridae*, 24183 como *Arteriviridae*, 187 como *Spinareoviridae* y 57 como *Sedoreoviridae* (ver Fig. 32). Durante el entrenamiento se obtuvo una pérdida de 0.7149 y exactitud de 0.6617, lo cual indica que el modelo está cometiendo algunos errores en sus predicciones. Luego, al realizar la prueba del modelo con los datos metagenómicos o de prueba se obtiene que modelo alcanzó una exactitud de 0.5855, es decir, el modelo predice correctamente la etiqueta de clase en aproximadamente el 58.56 % de las muestras. La puntuación de precisión de 0.5734 indica que cuando el modelo predice una clase positiva, acierta alrededor del 57.34 % de las veces. Una sensibilidad de 0.5855 indica que el modelo es capaz de identificar correctamente aproximadamente el 58.56 % de las muestras positivas y, la puntuación F de 0.5644 muestra que hay un rendimiento en general bueno. Dichos resultados sugieren que se puede mejorar el desempeño; esto se puede conseguir si se aumenta la cantidad de datos de entrenamiento, ya que al aumentar el tiempo de entrenamiento se presentó una tendencia al sobreajuste.

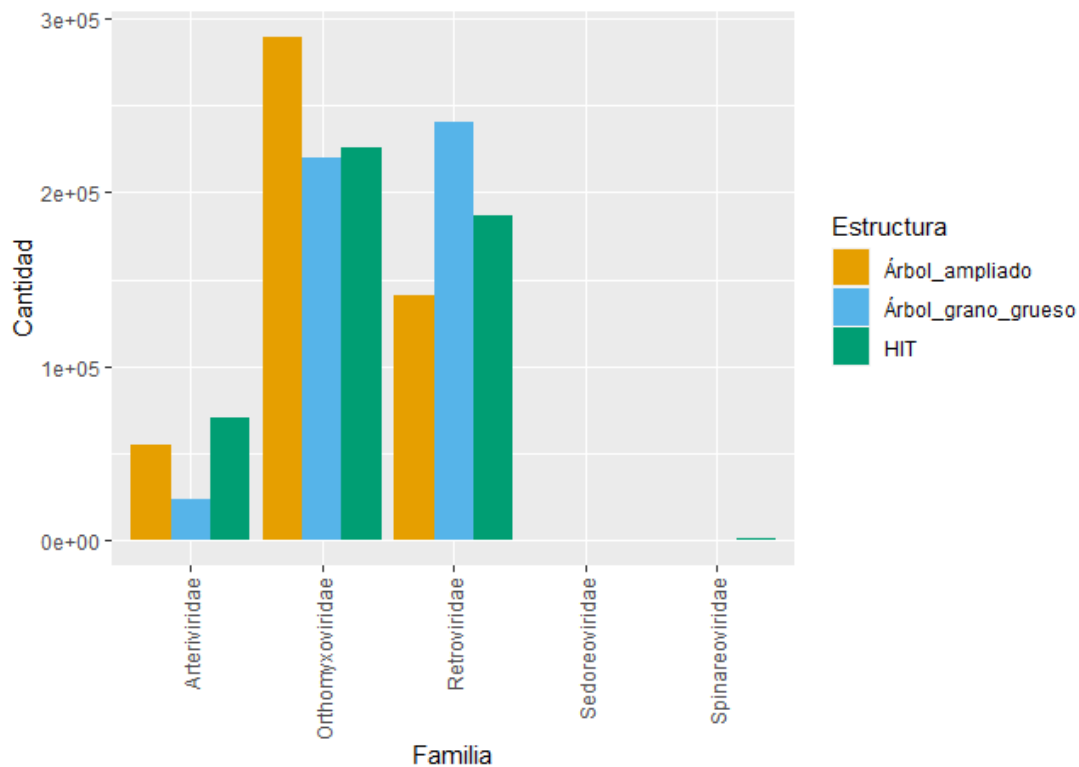


Figura 33: Histograma con los resultados de las tres estructuras secundarias para el modelo de clasificación IV.

El resumen de los resultados para este tercer modelo se exponen en la Figura 33. Durante el proceso de entrenamiento del modelo se presentaron múltiples retos en cuanto a las decisiones que debían tomarse a medida que se iba evaluando su comportamiento con los diferentes conjuntos de datos. Una de las primeras cosas que se evidenciaron, fue que al aumentar la cantidad de épocas o el tamaño del lote con el que se hacía el entrenamiento, se empezaban a presentar problemas de sobreajuste en el modelo. Para evitar problemas

de sobreajuste, se recurrió a añadir una técnica de regularización. Se intentó inicialmente con `dropout`, la cual es una técnica de regularización que elimina aleatoriamente algunas unidades de la red durante el entrenamiento para evitar que el modelo dependa demasiado de una sola característica; sin embargo, los resultados indican que esta técnica estaba haciendo que el modelo perdiera información importante. Luego, se cambió a la técnica del modelo I: la regularización L2, con la cual se obtuvieron mejores resultados en lo relacionado al sobreajuste. Al analizar de forma conjunta tanto los resultados, como las diferentes modificaciones hechas al modelo, es evidente la importancia de enriquecer el conjunto de datos con los cuales se está entrenando al modelo, ya que mejorar el desequilibrio existente entre las clases puede resultar en una extracción de características más útil para la clasificación.

4.5. Resultados

Teniendo los resultados para cada modelo, es posible agruparlos de acuerdo a las estructuras.

- **Árbol ampliado:** En la Figura 34 se encuentran los diferentes valores obtenidos para las métricas que ayudaron a evaluar el desempeño de cada uno de los modelos con la representación de árbol ampliado.

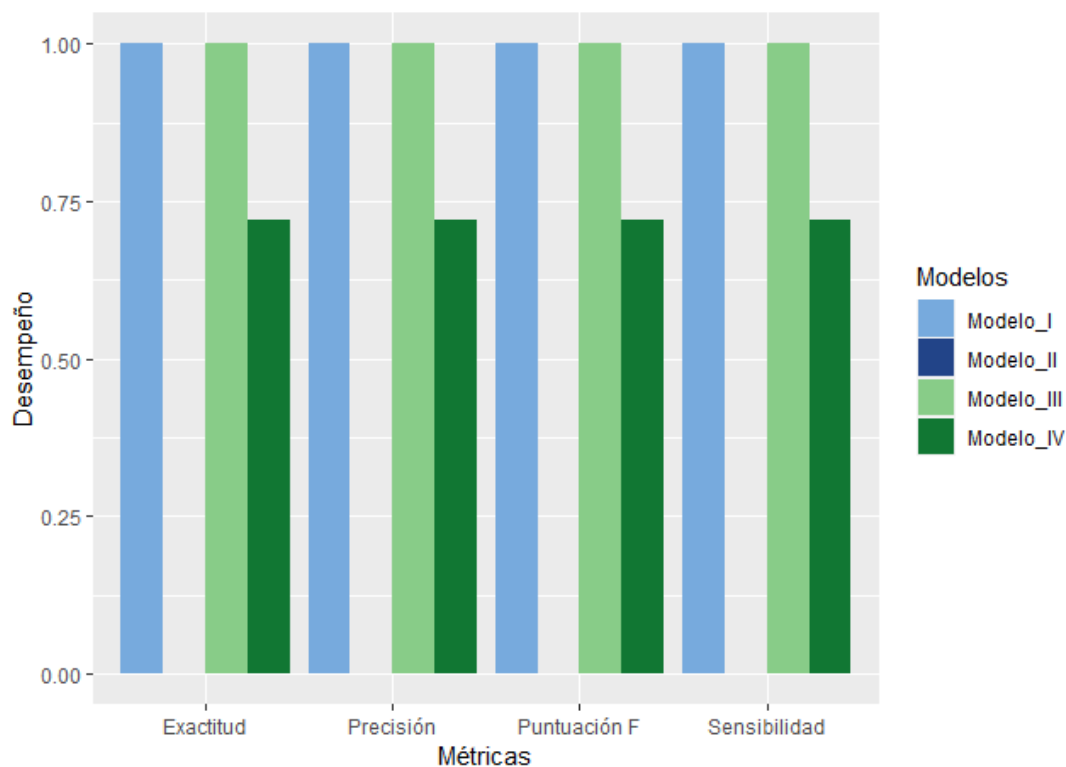


Figura 34: Diagrama de barras con los resultados de las métricas de cada modelo para la representación de árbol ampliado.

Es importante ser cuidadoso cuando se interpretan estos resultados de las pruebas de cada modelo, ya que se mencionó que a pesar de que en modelos como el prime-

ro, los valores de las métricas daban 1, lo que estaba ocurriendo de trasfondo era un sobreajuste que no fue posible mejorar tanto por la poca cantidad de datos de entrenamiento, como por el desequilibrio presente entre las clases o familias. Esta representación no parece funcionar bien con el modelo I ni con el II, ya que por la naturaleza de los modelos no les es posible extraer o capturar la información para hacer el proceso de clasificación. En el primer modelo parece que la notación de esta estructura ayudaría, ya que no es una notación compleja como con las otras estructuras, sin embargo, esto resulta ser un inconveniente a la hora de generalizar para datos que no ha visto. Mientras que en el segundo modelo, a pesar de que la notación es más sencilla, la estructura no lo es, complejizando el problema cuando se desea ubicar y clasificar las secuencias en una familia.

En cuanto a los modelos III y IV, el modelo III muestra ser mejor a la hora de generalizar con datos no conocidos. A pesar de que el modelo IV obtuvo resultados prometedores con esta representación, aún puede mejorarse la generalización de los datos, para esto sería necesario ampliar la cantidad de datos de entrenamiento y la cantidad de representantes que hay por cada familia, para tener así un conjunto de entrenamiento más balanceado.

- **HIT:** La representación HIT tuvo resultados equitativos en cuanto a la cantidad de modelos con los cuales se pudo hacer la clasificación (ver Fig. 35), es decir, funcionó con dos de los cuatro modelos.

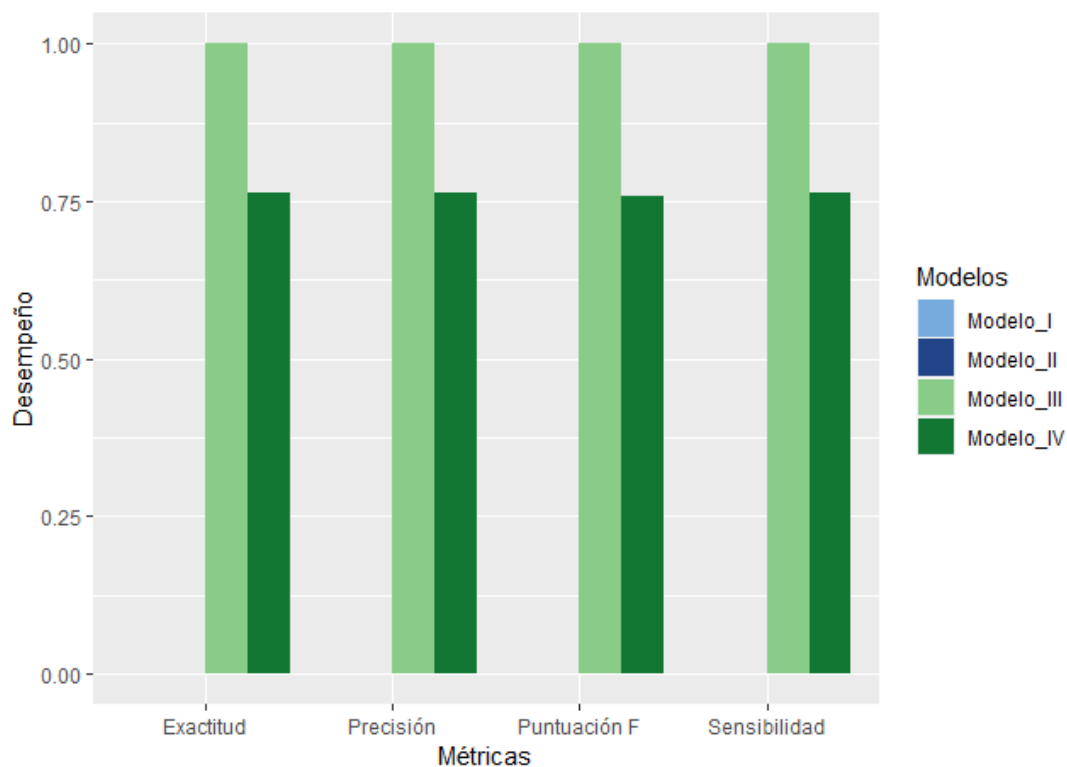


Figura 35: Diagrama de barras con los resultados de las métricas de cada modelo para la representación HIT.

Para los modelos I y II, la especificidad y nivel de complejidad de la notación de esta estructura secundaria, imposibilitó la clasificación de las secuencias. En ambos casos, la notación de la estructura dificultó que por un lado, se pudiera abstraer características específicas que ayudaran a representar dichas estructuras de una forma que el modelo pudiese interpretar y, por otro lado, tanta variabilidad en la notación causó que, en el caso del segundo modelo, fuera imposible clasificar las estructuras en alguno de los árboles asociados a las familias de interés. En el caso de los modelos III y IV, se obtuvo que esa misma complejidad en la notación resultó ser útil a la hora de probar el modelo. El modelo III mostró una ventaja sobre el modelo IV, y es que parece comportarse mejor frente a secuencias o estructuras nuevas, es decir, mostró potencial en la clasificación de las secuencias metagenómicas con las cuales se estaba haciendo la prueba de cada modelo. En el caso del modelo IV, a pesar de haber obtenido buenos resultados, es claro que el desequilibrio en las clases puede estar afectando el desempeño general del mismo, lo cual se vio reflejado al aumentar diferentes hiperparámetros, ya que causaba un sobreajuste en el modelo. De esta notación, se puede concluir que los modelos III y IV funcionan bien con estructuras complejas, ya que son buenos determinando las características de cada etiqueta. Al igual que con la representación anterior, estos modelos podrían verse beneficiados si se aumenta la cantidad de datos en el conjunto de entrenamiento.

- **Árbol de grano grueso:** La estructura de árbol de *grano grueso* mostró resultados que a primera vista parecen ser los más prometedores.

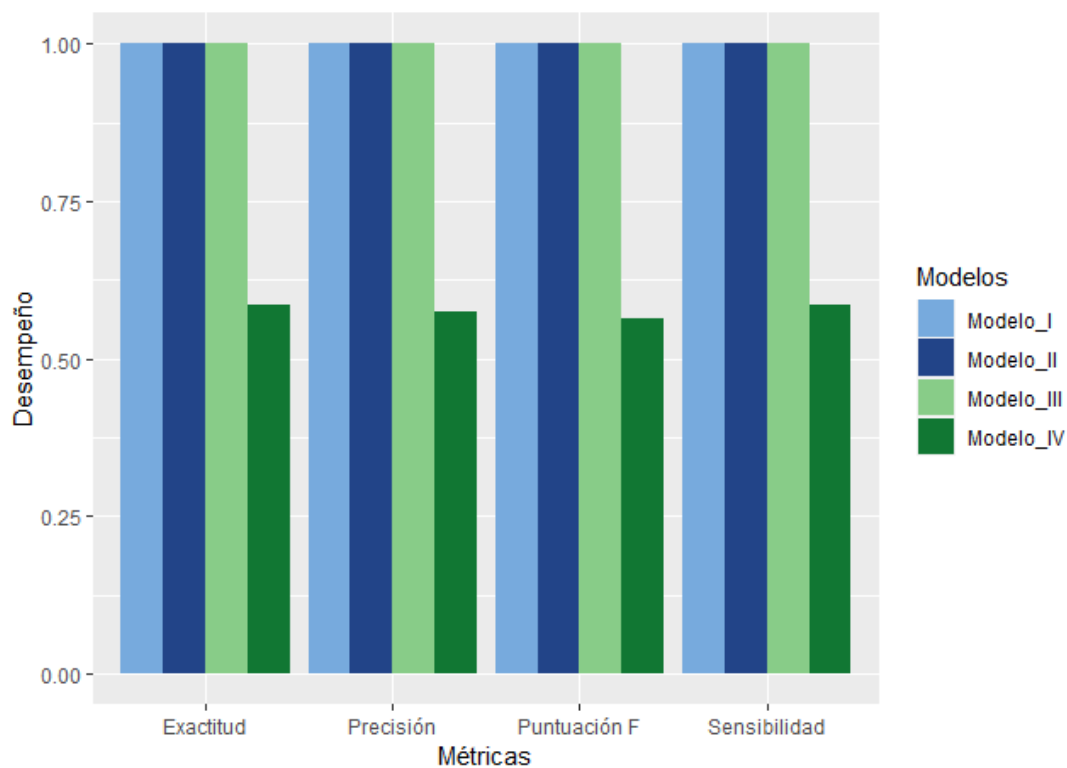


Figura 36: Diagrama de barras con los resultados de las métricas de cada modelo para la representación de árbol de *grano grueso*.

En el capítulo previo se mencionó que de las tres estructuras, la de *grano grueso* por definición, es la única que no preserva la información completa de las secuencias, y esto tiene que ver con el hecho de que esta estructura se representa o basa en otras subestructuras de la representación secundaria del ARN. Si se observa la figura 36, en términos de desempeño es a la que mejor le fue de las tres representaciones con los cuatro modelos. A pesar de que los resultados de las métricas, luego de hacer la prueba del modelo I, dieron 1 en todo, este se puede descartar ya que sin importar los ajustes que se le hicieron al modelo, sigue presentando problemas de sobreajuste y no es bueno generalizando. El modelo II solo funcionó para esta estructura, según los resultados este modelo es útil con representaciones de estructura cuyo nivel de complejidad es intermedio, es decir, en este caso la notación contaba con varios caracteres, pero al ser una notación basada en sub-estructuras, le fue más fácil capturar información y clasificar las secuencias en los diferentes árboles creados para cada familia.

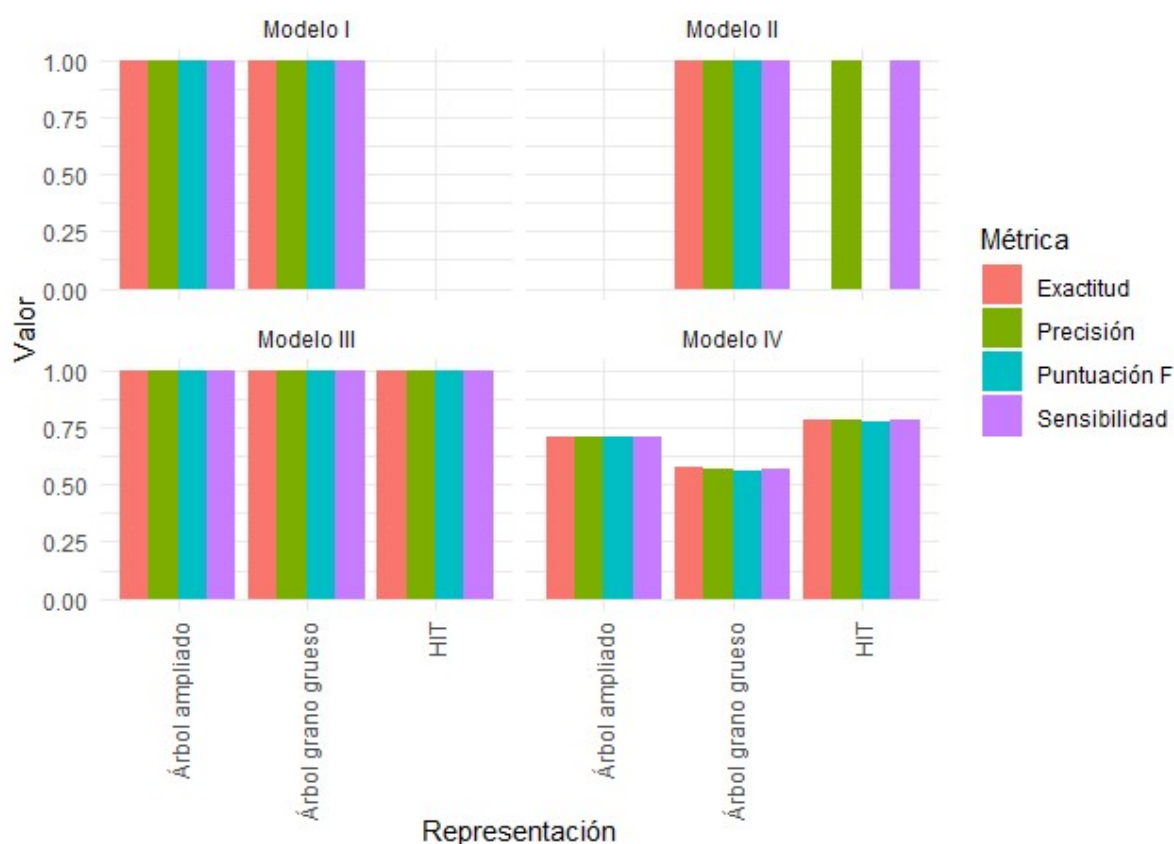


Figura 37: Resultados con las métricas de la prueba de cada modelo con las tres representaciones secundarias.

En el modelo III, a pesar de que por los valores de las métricas hay una inclinación en pensar que todo funciona muy bien, se hace evidente que tener pocos datos de entrenamiento y un desequilibrio entre las clases, puede conducir a resultados sesgados. Con este modelo, se clasificó un poco más del 80 % de las secuencias metagenómicas en una familia que tenía solo 102 representantes en el conjunto de entrenamiento, lo cual sugiere que tal vez el modelo no resulta ser tan bueno a la hora de generalizar con datos no vistos. En cuanto a los resultados del modelo IV, no son mucho mejores. No solo los valores

obtenidos para las métricas no fueron muy buenos, sino que además había una tendencia a sobreajustarse a los datos, y al aplicar técnicas que ayudaran a regular este sobreajuste, ocurrió que el desempeño disminuía drásticamente, de tal forma que tanto para los datos de entrenamiento como para los de validación, la clasificación correcta de los datos no llegaba ni a un 50%, indicando que tenía problemas tanto con los datos conocidos como con los no conocidos.

Aunque esta estructura mostraba tener el potencial de ser la más idónea por los resultados obtenidos, no resulta ser la mejor. Esto puede deberse precisamente a que la notación al no capturar toda la información de las secuencias, generaliza en formas que no son útiles a los modelos. Nuevamente, sería beneficioso considerar en trabajos futuros la posibilidad de aumentar la cantidad de datos en el conjunto de entrenamiento, de tal forma que se mejore el equilibrio entre las familias. En la figura 37 se encuentran unificados los resultados que se consiguieron para las métricas de desempeño durante la prueba de cada uno de los modelos con cada una de las representaciones secundarias.

4.6. Discusión

Este trabajo tenía no solo como objetivo proponer y evaluar un modelo basado en técnicas de machine learning que clasificara virus de ARN según sus niveles de representación teórica, de tal forma que con ayuda de este se pudiera estudiar y analizar cuál representación muestra el potencial de ser la más adecuada en el proceso de clasificación. Sino además, describir diferentes representaciones teóricas basadas en niveles de secuencia y estructura para los virus de ARN, para así determinar entradas compatibles para el modelo de acuerdo a dichos niveles de representación. Es importante tener en cuenta que este trabajo más allá de ser una continuación del trabajo realizado en [4], es una propuesta que abre las puertas a el uso de un enfoque diferente, donde el problema de clasificación se lleva a cabo por medio de otro tipo de notaciones y modelos.

Cuando se comparan los resultados obtenidos en este trabajo con otros métodos y modelos actuales como los presentados en [131] o [132], resulta evidente la utilidad y buen desempeño de los modelos basados en redes neuronales, particularmente, las CNN-LSTM en problemas de clasificación. Aunque las ventajas y capacidad que tienen estos modelos a la hora de extraer características de las secuencias los hace muy llamativos, es claro que para obtener modelos más robustos que puedan hacer uso de otras representaciones teóricas, como las propuesta acá, se necesita añadir otros algoritmos ya sea de aprendizaje supervisado o no supervisado, que mejoren y amplifiquen el nivel de clasificación al cual se puede llegar.

Si bien el problema de clasificar secuencias tanto de ADN como de ARN no es algo novedoso per se, lo interesante es poder hacer uso de otro tipo de representaciones teóricas que no se han considerado antes en la tarea clasificación. El proceso de implementación requirió de considerar diferentes estrategias que permitieran sacar provecho de las notaciones y características de las representaciones de árbol ampliado, HIT, y árbol de *grano grueso*. Los resultados que se obtuvieron ayudaron a visualizar que hay representaciones que no solo son más significativas y útiles biológicamente, sino también computacionalmente; a pesar de ser notaciones equivalentes, la complejidad o especificidad de estas representaciones parece ser esencial en proceso de clasificación, ya que esto ayuda a caracterizar

de forma más clara cada una de las clases sobre las cuales se tenía interés. Además, el haber obtenido que los modelos III y IV son los que mejor se acomodaron a la tarea de clasificación, abre las posibilidades sobre lo que los modelos basados en CNN-LSTM pueden hacer con diferentes representaciones teóricas de las secuencias de ARN; además de incrementarse el interés en modelos que usan HMM en este tipo de problemas.

Es necesario tener en cuenta dos limitantes que se evidenciaron en este trabajo: la cantidad de los datos de entrenamiento y el desbalance de las clases. Se tenía claro que ambos podrían incurrir en inconvenientes durante la clasificación, sin embargo, se procedió sin ningún tipo de herramienta que abordara el desequilibrio de clases en los modelos. Si bien se realizó una revisión de algoritmos y técnicas que se usan para tratar el desequilibrio, se tenía la preocupación de que si se utilizaban métodos de remuestreo se introdujeran muestras sintéticas que no fuesen representativas de la verdadera distribución presente en los datos, lo que podía dar lugar a un sobreajuste o a una mayor complejidad del modelo. Además, como estos métodos descartan una parte de los datos de la clase mayoritaria, se podría causar una pérdida de información y reducir la capacidad de generalización. Las otras técnicas que se analizaron estaban asociadas a la modificación del algoritmo de tal forma, que se trabajara o se pre-entrenara los modelos con sub-conjuntos de datos balanceados, sin embargo, por la distribución de estos (Ver Fig. 18 y Tabla 4) esta opción no es viable, ya que no hay suficientes secuencias que sirvan como ejemplo para crear datos sintéticos representativos.

Conclusiones y perspectivas futuras

5.1. Conclusiones

Por medio de este trabajo fue posible estudiar y analizar las representaciones secundarias de árbol ampliado, representación HIT, y árbol de *grano grueso*, para secuencias de ARN viral. Este proceso permitió determinar características útiles de cada una de las estructuras, de tal forma que por medio de estas se pudiera extraer información con el potencial de ser implementada en los modelos de clasificación. Considerando el tipo de información que cada una de las estructuras brindaba, se hizo evidente la necesidad de ampliar la búsqueda de algoritmos que se pudieran ajustar más a la tarea particular que se esperaba desarrollar.

En principio se evaluaron y examinaron algoritmos convencionales que se suelen implementar en tareas de clasificación, sin embargo, se optó por utilizar no solo éste tipo de modelos, sino otros que aportaran variedad respecto a la forma con la cual se usan las características de las distintas representaciones de estructura secundaria en el modelo. El primer modelo fue un perceptrón multicapa, cuyas principales ventajas fueron la habilidad que mostró a la hora de aprender relaciones complejas entre las entradas y salidas, además de la flexibilidad que tiene, es decir, fue capaz de manejar varios tipos de datos de entrada, sin embargo, una de sus limitaciones fue lo propenso que resultó al sobreajuste, especialmente cuando se trata de conjuntos de datos pequeños. El segundo modelo es un modelo probabilístico que opera sobre secuencias de símbolos o caracteres; con este modelo una ventaja fue la forma en la que se captaron patrones en las secuencias, no obstante presentó inconvenientes en ese mismo aspecto ya que no le fue posible captar patrones cuando las secuencias eran de alta dimensión. El tercer modelo basado en HMM funcionó bien para el tipo de datos de entrada que se tenían, aunque el nivel de variabilidad en las secuencias puede afectar la capacidad que el modelo tiene al momento de representar relaciones complejas. Finalmente, el último modelo combina redes neuronales convolucionales (CNN) y redes de largo y plazo (LSTM). Este modelo se destaca porque suelen tener la habilidad de aprender características y capturar patrones en los datos, sin embargo, requiere de grandes cantidades de datos etiquetados para generalizar bien y evitar el sobreajuste.

De acuerdo a las comparaciones y análisis de los resultados realizados para cada una de las estructuras secundarias respecto a cada uno de los modelos, se puede concluir que de las tres representaciones de estructura secundaria, la representación HIT tiene el potencial de ser la mejor para realizar la clasificación de virus de ARN, esto se debe a la variabilidad

que hay en la notación, lo cual ayuda a extraer patrones de forma más clara y útil. En cuanto al modelo, los modelos III y IV, es decir, el modelo oculto de Markov y el modelo de CNN-LSTM, mostraron los mejores resultados a la hora de capturar la información de las estructuras para la tarea de clasificación, no obstante, ambos presentan falencias que se podrían mejorar si se optimiza la cantidad y balance del conjunto de datos de entrenamiento.

5.2. Perspectivas futuras

Los resultados obtenidos abren las puertas para que a futuro se siga trabajando en mejorar modelos de clasificación, esto se puede hacer mediante la exploración de nuevas arquitecturas de aprendizaje profundo, que ayuden a capturar dependencias e interacciones más complejas en datos metagenómicos virales. También se puede continuar en la investigación de características y representaciones más informativas para metagenomas virales de ARN que mejoren el rendimiento de la clasificación, de tal forma que se identifiquen características que distingan a los distintos virus de ARN. Asimismo, a medida que el campo de la metagenómica y metavirómica continúa expandiéndose, es necesario abordar los retos asociados a la clasificación de virus ARN nuevos. Así, la investigación futura puede centrarse en el desarrollo de modelos de clasificación robustos que puedan detectar y clasificar eficazmente estos virus de ARN menos comunes, que pueden tener características en su representación de secuencia o estructura únicas.

Estas futuras líneas de investigación son algunas de las que pueden contribuir a avanzar en la comprensión de los virus de ARN, permitiendo una clasificación más precisa y eficiente, y facilitando su aplicación en diversos aspectos de la investigación, el diagnóstico y la vigilancia de enfermedades infecciosas. Los resultados de este trabajo permiten evidenciar el potencial presente en otras notaciones de las estructuras secundarias del ARN, y de otros algoritmos, que tal vez funcionan de forma promedio al implementarlos de manera individual, pero que si se juntan tal vez podrían conducir a modelos de clasificación para secuencias virales de ARN más robustos.

5.2.1. Limitaciones

Se evidenció que la poca cantidad en los datos de entrenamiento y el desequilibrio entre las clases, podría incurrir en resultados que indicaran un desempeño bajo en las pruebas del modelo. Estos dos limitantes son importantes considerarlos para que en trabajos futuros se puedan mejorar los resultados ya obtenidos. Por un lado, el problema asociado a la cantidad de datos de entrenamiento se puede mejorar ya sea tomando las secuencias de otra base de datos, o añadiendo a la base actual secuencias de otras bases de datos; de cualquier forma se requiere que haya una mayor cantidad de datos garantizando que no se presenten repeticiones de las secuencias, esto con el propósito disminuir el ruido y la redundancia de la información. Por otro lado, el desbalance entre las familias puede verse beneficiado si se arregla el problema anterior, de persistir este inconveniente, sería necesario re-evaluar y reconsiderar métodos y técnicas para balancear clases.

Grafo de De Bruijn

En teoría de grafos, un grafo de Bruijn n -dimensional de m símbolos es un grafo dirigido que representa solapamientos de longitud $n - 1$ entre secuencias de símbolos. Tiene m^n vértices, que consisten en todas las posibles secuencias de longitud n de los símbolos dados, donde el mismo símbolo puede aparecer varias veces en una secuencia. Además, cada vértice tiene exactamente m aristas entrantes y m aristas salientes. Para un conjunto de m símbolos $S = \{s_1, \dots, s_m\}$, el conjunto de vértices o nodos es:

$$V = \{(s_1, s_1, \dots, s_1), \dots, (s_1, s_1, \dots, s_m), \dots, (s_1, \dots, s_2, s_1), \dots, (s_m, \dots, s_m, s_m)\}$$

Si uno de los vértices puede expresarse como otro vértice desplazando todos sus símbolos un lugar a la izquierda y añadiendo un nuevo símbolo al final de este vértice, entonces este último tiene una arista dirigida al vértice anterior. Por lo tanto, el conjunto de arcos o aristas dirigidas es:

$$E = \{((t_1, t_2, \dots, t_n), (t_2, \dots, t_n, s_j)) \mid t_i \in S, 1 \leq i \leq n, 1 \leq j \leq m\}$$

Debido a que los grafos de De Bruijn han sido utilizados en bioinformática para el ensamblaje de genomas y transcriptomas a partir de fragmentos de reads provenientes de un secuenciamiento [137, 138], se puede interpretar estos grafos como que los nodos representan todos los k -meros posibles [137], y además simbolizan una serie de k -meros superpuestos, donde los k -meros adyacentes se solapan en $k - 1$ nucleótidos [136]. La forma de construir un grafo de estos es conectar un $(k - 1)$ -mero mediante una arista dirigida a un segundo $(k - 1)$ -mero si hay algún k -mero cuyo prefijo sea el primero y cuyo sufijo sea el segundo [137].

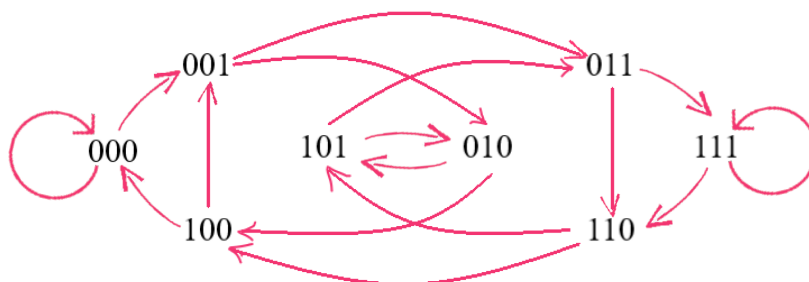


Figura 38: Grafo de De Bruijn construido con el alfabeto binario, donde $m = 2$ y $n = 3$.

Para ilustrar este concepto, considere el siguiente ejemplo (ver Fig. 38). Sea $A = \{0, 1\}$ el alfabeto binario, si se quiere construir un grafo con este alfabeto, de tal forma que las cadenas sean de longitud $n = 3$, se obtendría que hay en total $2^3 = 8$ cadenas:

000, 001, 010, 011, 111, 110, 101, 100

Note que hay un arco que va del nodo 111 al nodo 110, esto es porque el sufijo de longitud 2 de la cadena 111 es igual al prefijo de longitud 2 de la cadena 110. Lo mismo ocurre con el arco que va desde el nodo 010 hasta el nodo 100, en este caso, el sufijo de longitud 2 de 010 es igual al prefijo de longitud 2 del nodo 100. Esto mismo se extiende para cadenas de ADN o ARN, con la diferencia de que el alfabeto allí son las letras A, T, C, G o A, U, C, G, respectivamente. Y las cadenas que se forman son los k -meros de longitud k .

Matriz de Confusión

- Los verdaderos positivos (VP) o *true positive (TP)* son los casos en los que la clase real del punto de datos era Verdadero y la el resultado predicho también es verdadero [86].
- Los verdaderos negativos (VN) o *true negatives (TN)* son los casos en los que la clase real del punto de datos era falso y el resultado también es falso [86].
- Los falsos positivos (FP) o *false positive (FP)* son los casos en los que la clase real del punto de datos era falsa y la predicha es verdadera. Falso es porque el modelo ha predicho incorrectamente y positivo porque la clase predicha era positiva. A esta situación se le conoce como **error de tipo 1** [86].
- Los falsos negativos (FN) o *false negative (FN)* son los casos en los que la clase real del punto de datos era verdadera y la predicha es falsa. Falso es porque el modelo ha predicho incorrectamente y negativo porque la clase predicha era negativa. A esta situación se le conoce como **error de tipo 2** [86].

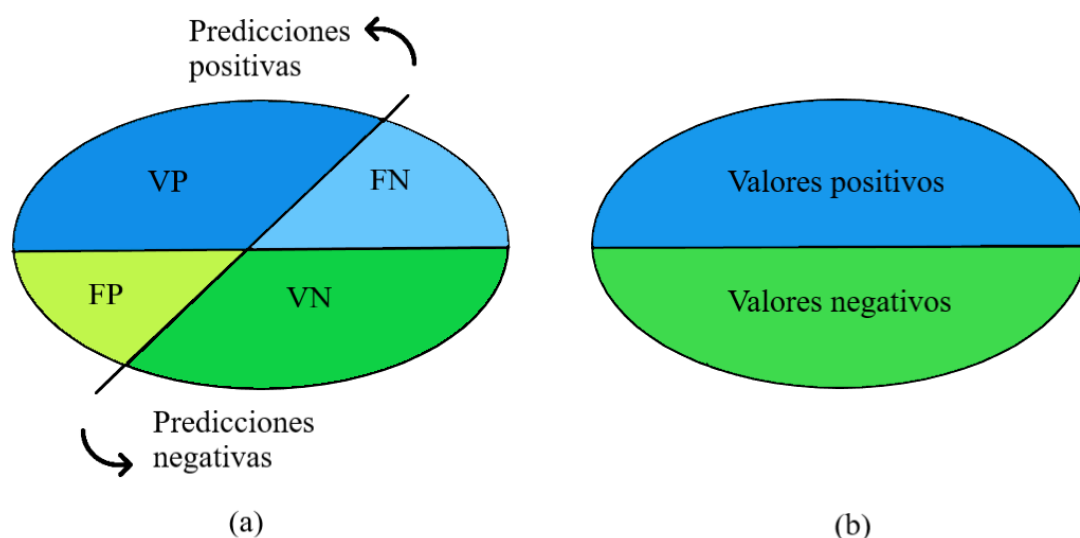


Figura 39: (a) Representación elíptica de los cuatro resultados posibles de la matriz de confusión. (b) Representación elíptica de los valores positivos y negativos del conjunto de datos. Imagen basada de [86].

Truco del Kernel

El truco del kernel evita tener una función explícita que es necesaria para que algoritmos de aprendizaje lineal aprendan un límite de decisión. Para cualquier \mathbf{x} y \mathbf{x}' en el espacio de entrada X , ciertas funciones $k(\mathbf{x}, \mathbf{x}')$ se pueden expresar como el producto interno en otro espacio V . Dada una función $k : X \times X \rightarrow \mathbb{R}$ (conocida como función kernel o *kernel*), el kernel se puede escribir como un *feature map* $\varphi : X \rightarrow V$ de tal forma que:

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_V$$

donde $\langle \cdot, \cdot \rangle_V$ es un producto interno. Una representación explícita de φ no se requiere, siempre y cuando V sea un espacio vectorial con producto escalar [81].

D

Función escalonada

En matemáticas una función escalonada es aquella función definida a trozos que en cualquier intervalo finito $[a, b]$ en el cual esté definida tiene un número finito de discontinuidades $c_1 < c_2 < \dots < c_n$, y en cada intervalo abierto (c_k, c_{k+1}) es constante, teniendo discontinuidades de salto en los puntos c_k . Ahora, en aprendizaje de máquina la función escalonada más utilizada en los perceptrones es la función escalonada de Heaviside (ver (16)), o a veces se utiliza la función de signo (ver (17)).

$$\text{heaviside}(z) = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z \geq 0 \end{cases} \quad (16)$$

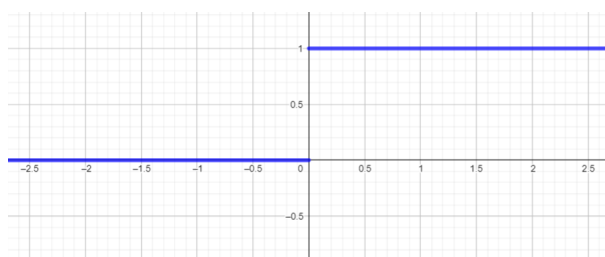


Figura 40: Función de Heaviside graficada en GeoGebra.

$$\text{sgn}(z) = \begin{cases} -1 & \text{si } z < 0 \\ 0 & \text{si } z = 0 \\ 1 & \text{si } z > 0 \end{cases} \quad (17)$$

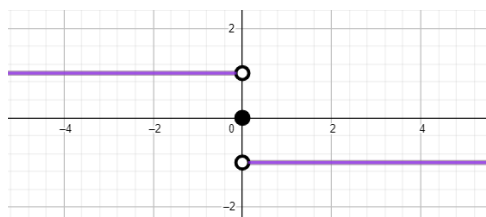


Figura 41: Función del signo graficada en GeoGebra.

Distribución de Boltzmann

La distribución de Boltzmann es una distribución de probabilidad que da la probabilidad de un determinado estado en función de la energía de ese estado y de la temperatura del sistema al que se aplica la distribución [88]:

$$p_i = \frac{1}{Q} e^{-\varepsilon_i/kT} = \frac{e^{-\varepsilon_i/(kT)}}{\sum_{j=1}^M e^{-\varepsilon_j/(kT)}}$$

donde p_i es la probabilidad del estado i , ε_i es la energía del estado i , k la constante de Boltzmann, T la temperatura absoluta del sistema y M el número de todos los estados accesibles al sistema de interés. El denominador de normalización Q es la función de partición canónica que resulta de la restricción de que las probabilidades de todos los estados accesibles deben sumar 1 [88].

$$Q = \sum_{i=1}^M e^{-\varepsilon_i/(kT)}$$

Puntuación de Información Mutua

Supongamos que se tiene un alineamiento múltiple \mathbb{A} de N secuencias. Por \mathbb{A}_i se denota la i -ésima columna del alineamiento, mientras que a_i^α es la entrada en la α -ésima fila de la i -ésima columna. La longitud de \mathbb{A} , es decir, el número de columnas, es n . Además, sea $f_i(X)$ la frecuencia de la base X en la posición i del alineamiento y sea $f_{ij}(XY)$ la frecuencia de encontrar simultáneamente a X en la posición i y a Y en j .

La *puntuación de información mutua* es la forma más común de cuantificar la covariación de secuencias [60, 73, 74, 75]:

$$MI_{ij} = \sum_{X,Y} f_{ij}(XY) \log \frac{f_{ij}(XY)}{f_i(X)f_j(Y)}$$

Esta puntuación no utiliza las reglas de emparejamiento de bases del ARN, lo cual es deseable para conjuntos de datos grandes, ya que permite identificar pares de bases no canónicos. Sin embargo, para los conjuntos de datos pequeños, no tener en cuenta las reglas de emparejamiento de bases resulta más perjudicial que beneficioso. En particular, la información mutua no tiene en cuenta en absoluto las mutaciones no compensatorias consistentes, es decir, si se tienen sólo pares GC y GU en las posiciones i y j , entonces $MI_{ij} = 0$. Así, sitios con dos tipos diferentes de pares de bases se tratan igual que un par de posiciones conservadas [60].

Valor p

Una prueba de hipótesis estadística es un método de inferencia estadística utilizado para decidir si los datos disponibles apoyan suficientemente una hipótesis concreta. Las pruebas de hipótesis permiten hacer afirmaciones probabilísticas. En este tipo de pruebas el valor p se define como la probabilidad de que un valor estadístico calculado sea posible dada una hipótesis nula cierta, es decir, este valor ayuda a diferenciar resultados que son producto del azar del muestreo, de resultados que son estadísticamente significativos. Un valor p muy pequeño significa que ese resultado extremo observado sería muy improbable bajo la hipótesis nula. A continuación se da una definición más formal.

Considere una prueba estadística observada t de distribución desconocida T . El valor p es la probabilidad a priori de observar un valor estadístico de prueba al menos tan *extremo* como t si la hipótesis nula H_0 fuera cierta, es decir:

- Para una prueba de cola derecha unilateral el valor p sería:

$$p = P(T \geq t \mid H_0)$$

- Para una prueba de cola izquierda unilateral el valor p sería:

$$p = P(T \leq t \mid H_0)$$

- Si la distribución de T es simétrica respecto a cero, entonces $p = P(|T| \leq |t| \mid H_0)$. Es decir, para una prueba de dos caras, el valor de p sería:

$$p = 2\text{mín}\{P(T \geq t \mid H_0), P(T \leq t \mid H_0)\}$$

Algoritmo de Viterbi

El algoritmo de Viterbi es un algoritmo de programación dinámica utilizado para encontrar la secuencia más probable de estados ocultos (o etiquetas) que produce una secuencia de sucesos observables. Se suele utilizar en el contexto de los modelos ocultos de Markov (HMM), en los que los estados ocultos representan la estructura subyacente no observada de la secuencia, y los sucesos observables son la secuencia de observaciones que se supone que dependen de los estados ocultos. Este algoritmo calcula, para cada estado posible en cada paso de tiempo, el camino más probable que termina en ese estado hasta ese momento, dada la secuencia de observaciones hasta ese paso de tiempo. Esto se hace de forma recursiva, calculando primero el camino más probable que termina en cada estado en el primer paso de tiempo, y luego utilizando estos valores para calcular el camino más probable que termina en cada estado en el segundo paso de tiempo, y así sucesivamente. En cada paso temporal, el algoritmo mantiene un registro del camino de máxima probabilidad que termina en cada estado, así como del estado que le precede en ese camino. Esto le permite retroceder por el camino más probable al final de la secuencia para obtener la secuencia de estados ocultos que produjo las observaciones [107, 108].

Convolución

La convolución es una operación matemática sobre dos funciones f y g que produce una tercera función $f * g$ que expresa cómo la forma de una es modificada por la otra. Esta operación se define como la integral del producto de las dos funciones después de reflejar una sobre el eje y , esta integral se evalúa para todos los valores de desplazamiento, produciendo la función de convolución, o de otra forma [116, 140]:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

Un tipo de convolución es la convolución discreta. Esta convolución se usa en el procesamiento de señales y es útil para entender la forma en la que las CNN funcionan (ver Fig. 42). Para funciones de valor complejo f y g definidas sobre el conjunto \mathbb{Z} , la convolución discreta de f y g viene dada por [139, 140]:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

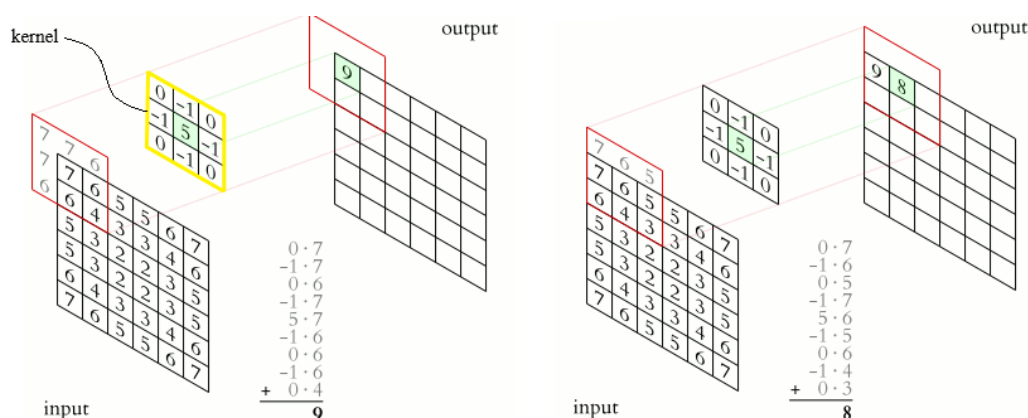


Figura 42: Imagen tomada y modificada de [140]. Plotke Michael (2013). 2D Image-Kernel Convolution Animation.

Distancia de Hamming

La distancia de Hamming entre dos cadenas de igual longitud es el número de posiciones en las que los símbolos correspondientes son diferentes. En otras palabras, mide el número mínimo de sustituciones necesarias para cambiar una cadena por la otra, o el número mínimo de errores que podrían haber transformado una cadena en la otra [77]. En la Figura 43 se encuentran dos ejemplos de la distancia de Hamming mínima entre dos vértices cualesquiera.

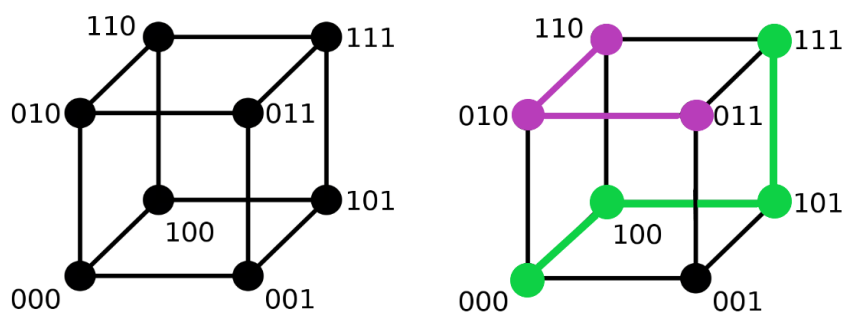


Figura 43: A la izquierda está un cubo binario de 3 bits para hallar la distancia de Hamming. A la derecha está la distancias $000 \rightarrow 111$, cuya distancia es 3 (verde), y de $110 \rightarrow 011$ la distancia es 2 (morado). Imagen modificada de Wikipedia contributors. (2023, Enero 22). Hamming distance. Wikipedia. https://en.wikipedia.org/wiki/Hamming_distance

Puntuación de Covarianza

El análisis de covariación es una forma de análisis comparativo, que identifica las posiciones en la molécula de ARN que tienen patrones similares de variación, o covariación, para todas o un subconjunto de las secuencias dentro de la misma familia de ARN [76].

Sea \mathbf{D} una matriz -adecuada- de 16×16 , $\mathcal{B} = (\text{GC}, \text{CG}, \text{AU}, \text{UA}, \text{GU}, \text{UG})$ y $d_H = (XY, X'Y')$ la distancia de Hamming de XY y $X'Y'$. Entonces, la puntuación de covarianza puede verse como el producto escalar $C_{ij} = \langle f_{ij}, \mathbf{D}f_{ij} \rangle$ [60]:

$$C_{ij} = \sum_{XY, X'Y'} f_{ij}(XY) \mathbf{D}_{XY, X'Y'} f_{ij}(X'Y')$$

La idea es que las mutaciones constantes como $\text{GC} \rightarrow \text{GU}$ cuenten menos que una mutación compensatoria como $\text{GC} \rightarrow \text{AU}$ [60].

Referencias

- [1] Reshu Chauhan, Surabhi Awasthi and Raghvendra Pratap Narayan (2021). Chapter 11 - Evolution and diversity of plant RNA viruses. *Plant Virus-Host Interaction* (Second Edition), Academic Press, Pages 303-318, ISBN 9780128216293, <https://doi.org/10.1016/B978-0-12-821629-3.00020-8>.
- [2] Marz M, Beerenwinkel N, Drosten C, et al. (2014) Challenges in RNA virus bioinformatics. *30*(13):1793-1799. doi:10.1093/bioinformatics/btu105
- [3] Villa, T.G., Abril, A.G., Sanchez, S. et al. Animal and human RNA viruses: genetic variability and ability to overcome vaccines. *Arch Microbiol* 203, 443–464 (2021). <https://doi.org/10.1007/s00203-020-02040-5>
- [4] Cobo Paz, V. (2020). Protocolo computacional para la asignación taxonómica de virus en metadatos genómicos. Universidad Nacional de Colombia
- [5] Mahmoudabadi, G., and Phillips, R. (2018). A comprehensive and quantitative exploration of thousands of viral genomes. *eLife*, 7, e31955. <https://doi.org/10.7554/eLife.31955>.
- [6] Struck D, Lawyer G, Ternes AM, Schmit JC, Bercoff DP (2014). Comet: adaptive context-based modeling for ultrafast hiv-1 subtype identification. *Nucleic Acids Res.*42(18):e144.
- [7] Wagner, Edward K.; Hewlett, Martinez J. (1999). *Basic virology*. Malden, MA: Blackwell Science, Inc. p. 249. ISBN 0-632-04299-0.
- [8] Patton JT (editor). (2008). *Segmented Double-stranded RNA Viruses: Structure and Molecular Biology*. Caister Academic Press. ISBN 978-1-904455-21-9.
- [9] Merriam-Webster. (n.d.). Orthomyxoviridae. In Merriam-Webster.com medical dictionary. Retrieved January 19, 2023, from <https://www.merriam-webster.com/medical/Orthomyxoviridae>.
- [10] Merriam-Webster. (n.d.). Retroviridae. In Merriam-Webster.com medical dictionary. Retrieved January 19, 2023, from <https://www.merriam-webster.com/medical/Retroviridae>.
- [11] Arteriviridae - ICTV. (s. f.). Tomado 19 de enero 2023, de https://ictv.global/report_9th/RNApos/Nidovirales/Arteriviridae.
- [12] Matthijnsens et al., (2022) ICTV Virus Taxonomy Profile: Sedoreoviridae, *Journal of General Virology* (2022) 103:001782.
- [13] Matthijnsens et al., (2022) ICTV Virus Taxonomy Profile: Spinareoviridae, *Journal of General Virology* (2022) 103:001781.

- [14] Jabeen A., Ahmad N., Raza K. (2018) Machine Learning-Based State-of-the-Art Methods for the Classification of RNA-Seq Data. In: Dey N., Ashour A., Borra S. (eds) Classification in BioApps. Lecture Notes in Computational Vision and Biomechanics, vol 26. Springer, Cham. <https://doi.org/10.1007/978-3-319-65981-76>.
- [15] Remita MA, Halioui A, Malick Diouara AA, Daigle B, Kiani G, Diallo AB. (2017 Apr 11) A machine learning approach for viral genome classification. BMC Bioinformatics. 18(1):208. doi:10.1186/s12859-017-1602-3
- [16] Fontana, W., Stadler, P. F., Bornberg-Bauer, E. G., Griesmacher, T., Hofacker, I. L., Tacker, M., Tarazona, P., Weinberger, E. D., & Schuster, P. (1993). RNA folding and combinatorial landscapes. Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics, 47(3), 2083–2099. <https://doi.org/10.1103/physreve.47.2083>.
- [17] Shapiro B. A. (1988). An algorithm for comparing multiple RNA secondary structures. Computer applications in the biosciences: CABIOS, 4(3), 387–393. <https://doi.org/10.1093/bioinformatics/4.3.387>.
- [18] Lorenz, Ronny and Bernhart, Stephan H. and Höner zu Siederdisen, Christian and Tafer, Hakim and Flamm, Christoph and Stadler, Peter F. and Hofacker, Ivo L. ViennaRNA Package 2.0. Algorithms for Molecular Biology, 6:1 26, 2011, doi:10.1186/1748-7188-6-26
- [19] Sikkema, R. S., y Koopmans, M. (2021). Preparing for Emerging Zoonotic Viruses. Encyclopedia of Virology, 256–266. <https://doi.org/10.1016/B978-0-12-814515-9.00150-8>.
- [20] Allen T. Global hotspots and correlates of emerging zoonotic diseases. Nature Communications. 2017;8(1)
- [21] Jones K.E. Global trends in emerging infectious diseases. Nature. 2008;451(7181):990–993.
- [22] S. Shadab, M. T. Alam Khan, N. A. Neezi, S. Adilina, and S. Shatabda, “DeepDBP: deep neural networks for identification of DNA-binding proteins,” Informatics in Medicine Unlocked, vol. 19, article 100318, 2020.
- [23] Gunasekaran, H., Ramalakshmi, K., Rex Macedo Arokiaraj, A., Deepa Kanmani, S., Venkatesan, C., y Suresh Gnana Dhas, C. (2021). Analysis of DNA Sequence Classification Using CNN and Hybrid Models. Computational and mathematical methods in medicine, 2021, 1835056. <https://doi.org/10.1155/2021/1835056>.
- [24] Fu, L., Niu, B., Zhu, Z., Wu, S., y Li, W. (2012). CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinformatics (Oxford, England), 28(23), 3150–3152. <https://doi.org/10.1093/bioinformatics/bts565>.
- [25] Li, W., y Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics (Oxford, England), 22(13), 1658–1659. <https://doi.org/10.1093/bioinformatics/bt1158>.

- [26] El Naqa, I., Murphy, M.J. (2015). What Is Machine Learning?. In: El Naqa, I., Li, R., Murphy, M. (eds) Machine Learning in Radiation Oncology. Springer, Cham. https://doi.org/10.1007/978-3-319-18305-3_1.
- [27] Larsson, A. (2014). AliView: a fast and lightweight alignment viewer and editor for large data sets. *Bioinformatics*30(22): 3276-3278. <http://dx.doi.org/10.1093/bioinformatics/bt>.
- [28] Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, Victor Robles, Machine learning in bioinformatics, Briefings in Bioinformatics, Volume 7, Issue 1, March 2006, Pages 86–112, <https://doi.org/10.1093/bib/bbk007>.
- [29] Shastry, K.A., Sanjay, H.A. (2020). Machine Learning for Bioinformatics. In: Srinivasa, K., Siddesh, G., Manisekhar, S. (eds) Statistical Modelling and Machine Learning Principles for Bioinformatics Techniques, Tools, and Applications. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-15-2445-5_3.
- [30] Samuel AL. Some studies in machine learning using the game of checkers. *IBM J Res Dev.* 1959;3(3):210–229.
- [31] Qifang Bi, Katherine E Goodman, Joshua Kaminsky, Justin Lessler, What is Machine Learning? A Primer for the Epidemiologist, *American Journal of Epidemiology*, Volume 188, Issue 12, December 2019, Pages 2222–2239, <https://doi.org/10.1093/aje/kwz189>.
- [32] Madhu Chetty, Jennifer Hallinan, Gonzalo A. Ruz, Anil Wipat, Computational intelligence and machine learning in bioinformatics and computational biology, *Biosystems*, Volume 222, 2022, 104792, ISSN 0303-2647, <https://doi.org/10.1016/j.biosystems.2022.104792>.
- [33] Hennig C, Meila M, Murtagh F, et al. *Handbook of Cluster Analysis*. 1st ed. Boca Raton, FL: CRC Press; 2015:34.
- [34] Bishop CM. *Pattern Recognition and Machine Learning*. 1st ed. New York, NY: Springer Publishing Compnay; 2006:424.
- [35] Bellett, A. J. D. (1967). Preliminary classification of viruses based on quantitative comparisons of viral nucleic acids. *Journal of Virology*, 1(2), 245-259.
- [36] LWOFF, A., & TOURNIER, P. (1971). Remarks on the Classification of Viruses. *Comparative Virology*, 1–42. <https://doi.org/10.1016/B978-0-12-470260-8.50006-3>.
- [37] Libretexts. (2021, 3 enero). 9.8A: Positive-Strand RNA Viruses of Animals. *Biology LibreTexts*. [https://bio.libretexts.org/Bookshelves/Microbiology/Microbiology_\(Boundless\)/09:_Viruses](https://bio.libretexts.org/Bookshelves/Microbiology/Microbiology_(Boundless)/09:_Viruses).
- [38] Patton JT, ed. (2008). *Segmented Double-stranded RNA Viruses: Structure and Molecular Biology*. Caister Academic Press. ISBN 978-1-904455-21-9.

- [39] Sanjuán, R., Nebot, M. R., Chirico, N., Mansky, L. M., & Belshaw, R. (2010). Viral mutation rates. *Journal of virology*, 84(19), 9733-9748.
- [40] Klein DW, Prescott LM, Harley J (1993). *Microbiology*. Dubuque, Iowa: Wm. C. Brown. ISBN 978-0-697-01372-9.
- [41] Domingo E. (1997). Rapid evolution of viral RNA genomes. *The Journal of nutrition*, 127(5 Suppl), 958S-961S. <https://doi.org/10.1093/jn/127.5.958S>.
- [42] RNA: The Versatile Molecule. (s. f.). <https://learn.genetics.utah.edu/content/basics/rna/>
- [43] Molnar, C. (2015, 14 mayo). 9.1 The Structure of DNA – Concepts of Biology – 1st Canadian Edition. Pressbooks. <https://opentextbc.ca/biology/chapter/9-1-the-structure-of-dna/>
- [44] Berg, J. M., Tymoczko, J. L., Stryer, L., & National Center for Biotechnology Information (U.S.). (2002). *Biochemistry, Fifth Edition*. W. H. Freeman.
- [45] Daròs, J. A., Elena, S. F., & Flores, R. (2006). Viroids: an Ariadne's thread into the RNA labyrinth. *EMBO reports*, 7(6), 593-598. <https://doi.org/10.1038/sj.embor.7400706>.
- [46] Payne S. (2017). Introduction to RNA Viruses. *Viruses*, 97-105. <https://doi.org/10.1016/B978-0-12-803109-4.00010-6>.
- [47] Wang D, Farhana A. Biochemistry, RNA Structure. [Updated 2022 May 8]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2022 Jan-. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK558999/>.
- [48] Wooley, J. C., Godzik, A., & Friedberg, I. (2010). A primer on metagenomics. *PLoS computational biology*, 6(2), e1000667. <https://doi.org/10.1371/journal.pcbi.1000667>.
- [49] Paez-Espino, D., Eloë-Fadrosh, E. A., Pavlopoulos, G. A., Thomas, A. D., Huntemann, M., Mikhailova, N., Rubin, E., Ivanova, N. N., & Kyrpides, N. C. (2016). Uncovering Earth's virome. *Nature*, 536(7617), 425-430. <https://doi.org/10.1038/nature19094>.
- [50] Metagenomics. (s. f.). En *Metagenomics*. Recuperado 24 de febrero de 2023, de <https://en.wikipedia.org/wiki/Metagenomics#Viruses>.
- [51] Staden R. (1979). A strategy of DNA sequencing employing computer programs. *Nucleic acids research*, 6(7), 2601-2610. <https://doi.org/10.1093/nar/6.7.2601>.
- [52] Edwards, R., Rohwer, F. Viral metagenomics. *Nat Rev Microbiol* 3, 504-510 (2005). <https://doi.org/10.1038/nrmicro1163>.
- [53] Kristensen DM, Mushegian AR, Dolja VV, Koonin EV. New dimensions of the virus world discovered through metagenomics. *Trends Microbiol*. 2010 Jan;18(1):11-9. doi: 10.1016/j.tim.2009.11.003. Epub 2009 Nov 26. PMID: 19942437; PMCID: PMC3293453.

- [54] Delwart, E. L. (2007). Viral metagenomics. *Reviews in medical virology*, 17(2), 115-131.
- [55] Sommers, P., Chatterjee, A., Varsani, A., & Trubl, G. (2021). Integrating Viral Metagenomics into an Ecological Framework. *Annual review of virology*, 8(1), 133–158. <https://doi.org/10.1146/annurev-virology-010421-053015>.
- [56] Grasis J. A. (2018). Host-Associated Bacteriophage Isolation and Preparation for Viral Metagenomics. *Methods in molecular biology* (Clifton, N.J.), 1746, 1–25. https://doi.org/10.1007/978-1-4939-7683-6_1.
- [57] Alavandi SV, Poornima M. Viral metagenomics: a tool for virus discovery and diversity in aquaculture. *Indian J Virol*. 2012 Sep;23(2):88-98. doi: 10.1007/s13337-012-0075-2. Epub 2012 Aug 14. PMID: 23997432; PMCID: PMC3550753.
- [58] Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins DG. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Mol Syst Biol*. 2011 Oct 11;7:539. doi: 10.1038/msb.2011.75. PMID: 21988835.
- [59] Sievers, F. and Higgins, D.G. (2018), Clustal Omega for making accurate alignments of many protein sequences. *Protein Science*, 27: 135-145. <https://doi.org/10.1002/pro.3290>.
- [60] Hofacker, Ivo & Stadler, Peter. (2006). *RNA Secondary Structures*. 10.1002/3527600906.mcb.200500009.
- [61] IUPAC. *Compendium of Chemical Terminology*, 2nd ed. (the Gold Book). Compiled by A. D. McNaught and A. Wilkinson. Blackwell Scientific Publications, Oxford (1997). Online version (2019-) created by S. J. Chalk. ISBN 0-9678550-9-8. <https://doi.org/10.1351/goldbook>.
- [62] Jones, C. P., & Ferré-D’Amaré, A. R. (2015). RNA quaternary structure and global symmetry. *Trends in biochemical sciences*, 40(4), 211–220. <https://doi.org/10.1016/j.tibs.2015.02.004>.
- [63] Xia, T., SantaLucia, J., Jr, Burkard, M. E., Kierzek, R., Schroeder, S. J., Jiao, X., Cox, C., & Turner, D. H. (1998). Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37(42), 14719–14735. <https://doi.org/10.1021/bi9809425>.
- [64] Mathews, D. H., Disney, M. D., Childs, J. L., Schroeder, S. J., Zuker, M., & Turner, D. H. (2004). Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences*, 101(19), 7287-7292.
- [65] Zuker M. (1989). On finding all suboptimal foldings of an RNA molecule. *Science* (New York, N.Y.), 244(4900), 48–52. <https://doi.org/10.1126/science.2468181>.
- [66] Sato, K., Akiyama, M. & Sakakibara, Y. RNA secondary structure prediction using deep learning with thermodynamic integration. *Nat Commun* 12, 941 (2021). <https://doi.org/10.1038/s41467-021-21194-4>.

- [67] Gruber, A. R., Findeiß, S., Washietl, S., Hofacker, I. L., & Stadler, P. F. (2010). RNAz 2.0: improved noncoding RNA detection. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 69–79.
- [68] Washietl, S., Hofacker, I. L., & Stadler, P. F. (2005). Fast and reliable prediction of noncoding RNAs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(7), 2454–2459. <https://doi.org/10.1073/pnas.0409169102>.
- [69] Roux, S., Enault, F., Hurwitz, B. L., & Sullivan, M. B. (2015). VirSorter: mining viral signal from microbial genomic data. *PeerJ*, 3, e985. <https://doi.org/10.7717/peerj.985>.
- [70] Thomas, T., Gilbert, J., & Meyer, F. (2012). Metagenomics - a guide from sampling to data analysis. *Microbial informatics and experimentation*, 2(1), 3. <https://doi.org/10.1186/2042-5783-2-3>.
- [71] Hofacker, I. L., Fekete, M., & Stadler, P. F. (2002). Secondary structure prediction for aligned RNA sequences. *Journal of molecular biology*, 319(5), 1059–1066. [https://doi.org/10.1016/S0022-2836\(02\)00308-X](https://doi.org/10.1016/S0022-2836(02)00308-X).
- [72] Washietl, S., & Hofacker, I. L. (2004). Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics. *Journal of molecular biology*, 342(1), 19–30. <https://doi.org/10.1016/j.jmb.2004.07.018>.
- [73] Chiu, D. K., & Kolodziejczak, T. (1991). Inferring consensus structure from nucleic acid sequences. *Computer applications in the biosciences : CABIOS*, 7(3), 347–352. <https://doi.org/10.1093/bioinformatics/7.3.347>.
- [74] Gutell, R. R., & Woese, C. R. (1990). Higher order structural elements in ribosomal RNAs: pseudo-knots and the use of noncanonical pairs. *Proceedings of the National Academy of Sciences of the United States of America*, 87(2), 663–667. <https://doi.org/10.1073/pnas.87.2.663>.
- [75] Gutell, R. R., Power, A., Hertz, G. Z., Putz, E. J., & Stormo, G. D. (1992). Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. *Nucleic acids research*, 20(21), 5785–5795. <https://doi.org/10.1093/nar/20.21.5785>.
- [76] Shang, L., Xu, W., Ozer, S., & Gutell, R. R. (2012). Structural constraints identified with covariation analysis in ribosomal RNA. *PLoS One*, 7(6), e39383.
- [77] Waggener, Bill (1995). *Pulse Code Modulation Techniques*. Springer. p. 206. ISBN 9780442014360.
- [78] I.L. Hofacker, W. Fontana, P.F. Stadler, S. Bonhoeffer, M. Tacker, P. Schuster (1994), "Fast Folding and Comparison of RNA Secondary Structures", *Monatshefte f. Chemie*: 125, pp 167-188
- [79] Zuker, M., & Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1), 133–148. <https://doi.org/10.1093/nar/9.1.133>.

- [80] Hofacker I. L. (2003). Vienna RNA secondary structure server. *Nucleic acids research*, 31(13), 3429–3431. <https://doi.org/10.1093/nar/gkg599>.
- [81] Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, Inc.
- [82] Sen, P.C., Hajra, M., Ghosh, M. (2020). Supervised Classification Algorithms in Machine Learning: A Survey and Review. In: Mandal, J., Bhattacharya, D. (eds) *Emerging Technology in Modelling and Graphics. Advances in Intelligent Systems and Computing*, vol 937. Springer, Singapore. https://doi.org/10.1007/978-981-13-7403-6_11.
- [83] Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1), 3-24.
- [84] Kotsiantis, S. (2011). Feature selection for machine learning classification problems: a recent overview. *Artificial Intelligence Review*, 42(1), 157-176.
- [85] Viral Genomes in Nature. (2021, January 3). Boundless. <https://bio.libretexts.org/@go/page/9330>.
- [86] Vujović, Ž. (2021). Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, 12(6), 599-606.
- [87] A short Tutorial on RNA Bioinformatics. The ViennaRNA Package and related Programs. (s. f.). Recuperado 10 de abril de 2023, de <https://algosb2019.sciencesconf.org/data/RNAtutorial.pdf>.
- [88] McQuarrie, A. (2000). *Statistical Mechanics*. Sausalito, CA: University Science Books.
- [89] Raschka, S. (2017). *Machine Learning*. University of Wisconsin–Madison. Department of Statistics. Recuperado 11 de abril de 2023, de https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf.
- [90] Wikipedia contributors. (2023, March 31). K-nearest neighbors algorithm. In Wikipedia, *The Free Encyclopedia*. Retrieved 16:44, April 11, 2023, from https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=1147498657.
- [91] Landau, S., Leese, M., Stahl, D., & Everitt, B. S. (2011). *Cluster analysis*. John Wiley & Sons.
- [92] 1.4. Support Vector Machines. (s. f.). scikit-learn. <https://scikit-learn.org/stable/modules/svm.html>
- [93] Wikipedia contributors. (2023, March 12). Support vector machine. In Wikipedia, *The Free Encyclopedia*. Retrieved 22:16, April 11, 2023, from https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=1144271534.
- [94] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).

- [95] Song, Y. Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2), 130–135. <https://doi.org/10.11919/j.issn.1002-0829.215044>.
- [96] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). *The Elements of Statistical Learning* (2nd ed.). Springer. ISBN 0-387-95284-5.
- [97] Haykin, S. S. (2009). *Neural networks and learning machines*. Upper Saddle River, NJ: Pearson Education.
- [98] Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60, 97-116.
- [99] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [100] Wikipedia contributors. (2023, March 2). Suffix tree. In Wikipedia, The Free Encyclopedia. Retrieved April 27 2023, from https://en.wikipedia.org/w/index.php?title=Suffix_tree&oldid=1142499280.
- [101] Ukkonen, E. (1995). On-line construction of suffix trees. *Algorithmica*, 14(3), 249-260.
- [102] Wikipedia contributors. (2023, March 22). Hidden Markov model. In Wikipedia, The Free Encyclopedia. Retrieved April 28 2023, from https://en.wikipedia.org/w/index.php?title=Hidden_Markov_model&oldid=1146111455.
- [103] Blunsom, P. (2004). Hidden markov models. *Lecture notes*, August, 15(18-19), 48.
- [104] Yoon B. J. (2009). Hidden Markov Models and their Applications in Biological Sequence Analysis. *Current genomics*, 10(6), 402–415. <https://doi.org/10.2174/138920209789177575>.
- [105] M. Przytycka, & Zheng, J. (2003). *Encyclopedia of Life Sciences: Hidden Markov Models (TM in Nature Encyclopedia of the Human Genome Nature Publishing Group, Ed.)*. NCBI. Recuperado 28 de abril de 2023, de <https://www.ncbi.nlm.nih.gov/CBBresearch/Przytycka/index.cgi#publications>.
- [106] Nelwamondo, F. V., Marwala, T., & Mahola, U. (2006). Early classifications of bearing faults using hidden Markov models, Gaussian mixture models, mel-frequency cepstral coefficients and fractals. *International Journal of Innovative Computing, Information and Control*, 2(6), 1281-1299.
- [107] Ryan, M. S., & Nudd, G. R. (1993). *The viterbi algorithm*.
- [108] Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications* (Vol. 5, Pages 237-301). Cham: Springer.
- [109] Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. MIT Press. p. 326.

- [110] Wikipedia contributors. (2023, April 30). Convolutional neural network. In Wikipedia, The Free Encyclopedia. Retrieved April 30, 2023, from https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1152491486.
- [111] Mishra, M. (2021, 15 diciembre). Convolutional Neural Networks, Explained - Towards Data Science. Medium. <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>.
- [112] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [113] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: continual prediction with LSTM. *Neural computation*, 12(10), 2451–2471. <https://doi.org/10.1162/089976600300015015>.
- [114] Hochreiter, S., & Schmidhuber, J. (1996). LSTM can solve hard long time lag problems. *Advances in neural information processing systems*, 9.
- [115] Brownlee, J. (2019). CNN Long Short-Term Memory Networks. <https://machinelearningmastery.com/cnn-long-short-term-memory-networks/>.
- [116] Zill, D., & Shanahan, P. (2009). *A First Course in Complex Analysis with Applications*. Jones & Bartlett Learning.
- [117] Lefkowitz, E. J., Dempsey, D. M., Hendrickson, R. C., Orton, R. J., Siddell, S. G., & Smith, D. B. (2018). Virus taxonomy: the database of the International Committee on Taxonomy of Viruses (ICTV). *Nucleic acids research*, 46(D1), D708-D717.
- [118] King, A. M., Adams, M. J., Carstens, E. B., & Lefkowitz, E. J. (2012). Virus taxonomy. Ninth report of the International Committee on Taxonomy of Viruses, 9.
- [119] Simmonds, P. (2015). Methods for virus classification and the challenge of incorporating metagenomic sequence data. *Journal of General Virology*, 96(6), 1193-1206.
- [120] Forterre, P. (2010). Giant viruses: conflicts in revisiting the virus concept. *Intervirology*, 53(5), 362-378.
- [121] Lwoff, A. (1959). Factors influencing the evolution of viral diseases at the cellular level and in the organism. *Bacteriological reviews*, 23(3), 109-124.
- [122] Yamada, T. (2011). Giant viruses in the environment: their origins and evolution. *Current opinion in virology*, 1(1), 58-62.
- [123] Doolittle, R. F., & Feng, D. F. (1992). Tracing the origin of retroviruses. *Genetic Diversity of RNA Viruses*, 195-211.
- [124] Temin, H. M. (1970). Malignant transformation of cells by viruses. *Perspectives in biology and medicine*, 14(1), 11-26.
- [125] Illangasekare, M., Sanchez, G., Nickles, T., & Yarus, M. (1995). Aminoacyl-RNA synthesis catalyzed by an RNA. *Science*, 267(5198), 643-647.
- [126] Gilbert, W. (1986). Origin of life: The RNA world. *nature*, 319(6055), 618-618.

- [127] Li, D., Liu, C. M., Luo, R., Sadakane, K., & Lam, T. W. (2015). MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics* (Oxford, England), 31(10), 1674–1676. <https://doi.org/10.1093/bioinformatics/btv033>.
- [128] Li, D., Luo, R., Liu, C. M., Leung, C. M., Ting, H. F., Sadakane, K., Yamashita, H., & Lam, T. W. (2016). MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices. *Methods* (San Diego, Calif.), 102, 3–11. <https://doi.org/10.1016/j.ymeth.2016.02.020>.
- [129] Xiong, J. (2006). Protein Motifs and Domain Prediction. In *Essential Bioinformatics* (pp. 85-94). Cambridge: Cambridge University Press. doi:10.1017/CB09780511806087.008.
- [130] Iqbal, T., Elahi, A., Wijns, W., & Shahzad, A. (2022). Exploring Unsupervised Machine Learning Classification Methods for Physiological Stress Detection. *Frontiers in medical technology*, 4, 782756. <https://doi.org/10.3389/fmedt.2022.782756>.
- [131] Mock, F., Kretschmer, F., Kriese, A., Böcker, S., & Marz, M. (2022). Taxonomic classification of DNA sequences beyond sequence similarity using deep neural networks. *Proceedings of the National Academy of Sciences*, 119(35), e2122636119.
- [132] Shang, J., & Sun, Y. (2021). CHEER: HierarCHical taxonomic classification for viral mEtagEnomic data via deep leaRning. *Methods* (San Diego, Calif.), 189, 95–103. <https://doi.org/10.1016/j.ymeth.2020.05.018>.
- [133] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- [134] Huson, D. H., Auch, A. F., Qi, J., & Schuster, S. C. (2007). MEGAN analysis of metagenomic data. *Genome research*, 17(3), 377-386.
- [135] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.
- [136] Zerbino, D. R., & Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome research*, 18(5), 821–829. <https://doi.org/10.1101/gr.074492.107>.
- [137] Compeau, P. E., Pevzner, P. A., & Tesler, G. (2011). How to apply de Bruijn graphs to genome assembly. *Nature biotechnology*, 29(11), 987–991. <https://doi.org/10.1038/nbt.2023>.
- [138] Martin, J., Wang, Z. (2011) Next-generation transcriptome assembly. *Nat Rev Genet* 12, 671–682. <https://doi.org/10.1038/nrg3068>.
- [139] Damelin, S. B., & Miller Jr, W. (2012). *The mathematics of signal processing* (No. 48). Cambridge University Press.

- [140] Wikipedia contributors (2023) Convolution. In Wikipedia, The Free Encyclopedia. Retrieved May 25, 2023, from <https://en.wikipedia.org/w/index.php?title=Convolution&oldid=1155936911>.
- [141] Budach, S., & Marsico, A. (2018). pysster: classification of biological sequences by learning sequence and structure motifs with convolutional neural networks. *Bioinformatics* (Oxford, England), 34(17), 3035–3037. <https://doi.org/10.1093/bioinformatics/bty222>.
- [142] Gelderblom, H. R. (1996). Structure and Classification of Viruses. In S. Baron (Ed.), *Medical Microbiology*. (4th ed.). University of Texas Medical Branch at Galveston.
- [143] Louten J. (2016). Virus Structure and Classification. *Essential Human Virology*, 19–29. <https://doi.org/10.1016/B978-0-12-800947-5.00002-8>.
- [144] Ajami, N. J., Wong, M. C., Ross, M. C., Lloyd, R. E., & Petrosino, J. F. (2018). Maximal viral information recovery from sequence data using VirMAP. *Nature communications*, 9(1), 3205. <https://doi.org/10.1038/s41467-018-05658-8>.
- [145] Lin, J., Kramna, L., Autio, R., Hyöty, H., Nykter, M., & Cinek, O. (2017). Vipie: web pipeline for parallel characterization of viral populations from multiple NGS samples. *BMC genomics*, 18(1), 378. <https://doi.org/10.1186/s12864-017-3721-7>.
- [146] Lin, H. H., & Liao, Y. C. (2017). drVM: a new tool for efficient genome assembly of known eukaryotic viruses from metagenomes. *GigaScience*, 6(2), 1–10. <https://doi.org/10.1093/gigascience/gix003>.
- [147] Rampelli, S., Soverini, M., Turrone, S., Quercia, S., Biagi, E., Brigidi, P., & Candela, M. (2016). ViromeScan: a new tool for metagenomic viral community profiling. *BMC genomics*, 17, 165. <https://doi.org/10.1186/s12864-016-2446-3>.
- [148] Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., & Huttenhower, C. (2012). Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature methods*, 9(8), 811–814. <https://doi.org/10.1038/nmeth.2066>.
- [149] Tithi, S. S., Aylward, F. O., Jensen, R. V., & Zhang, L. (2018). FastViromeExplorer: a pipeline for virus and phage identification and abundance profiling in metagenomics data. *PeerJ*, 6, e4227. <https://doi.org/10.7717/peerj.4227>.
- [150] Yamashita, A., Sekizuka, T., & Kuroda, M. (2016). VirusTAP: Viral Genome-Targeted Assembly Pipeline. *Frontiers in microbiology*, 7, 32. <https://doi.org/10.3389/fmicb.2016.00032>.
- [151] Zhao, G., Wu, G., Lim, E. S., Droit, L., Krishnamurthy, S., Barouch, D. H., Virgin, H. W., & Wang, D. (2017). VirusSeeker, a computational pipeline for virus discovery and virome composition analysis. *Virology*, 503, 21–30. <https://doi.org/10.1016/j.virol.2017.01.005>.
- [152] Menzel, P., Ng, K. L., & Krogh, A. (2016). Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nature communications*, 7, 11257. <https://doi.org/10.1038/ncomms11257>.

- [153] Wood, D. E., & Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3), R46. <https://doi.org/10.1186/gb-2014-15-3-r46>.
- [154] Fiers, Walter & Contreras, Roland & Duerinck, Fred & Haegeman, Guy & Iserentant, Dirk & Merregaert, Joseph & Jou, Willy & Molemans, Francis & Raeymaekers, Alex & Berghe, A & Volckaert, Guido & Ysebaert, Marc. (1976). Complete nucleotide sequence of bacteriophage MS2 RNA: primary and secondary structure of the replicase gene. *Nature*. 260. 500-7. [10.1038/260500a0](https://doi.org/10.1038/260500a0).
- [155] Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, C. A., Hutchison, C. A., Slocombe, P. M., & Smith, M. (1977). Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, 265(5596), 687–695. <https://doi.org/10.1038/265687a0>.
- [156] Cobbin, J. C., Charon, J., Harvey, E., Holmes, E. C., & Mahar, J. E. (2021). Current challenges to virus discovery by meta-transcriptomics. *Current Opinion in Virology*, 51, 48-55.
- [157] Bashiardes, S., Zilberman-Schapira, G., & Elinav, E. (2016). Use of Metatranscriptomics in Microbiome Research. *Bioinformatics and biology insights*, 10, 19–25. <https://doi.org/10.4137/BBI.S34610>.
- [158] Aguiar-Pulido, V., Huang, W., Suarez-Ulloa, V., Cickovski, T., Mathee, K., & Narasimhan, G. (2016). Metagenomics, Metatranscriptomics, and Metabolomics Approaches for Microbiome Analysis. *Evolutionary bioinformatics online*, 12(Suppl 1), 5–16. <https://doi.org/10.4137/EB0.S36436>.
- [159] Kelly, D., Yang, L., & Pei, Z. (2017). A review of the oesophageal microbiome in health and disease. *Methods in microbiology*, 44, 19-35.