UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño e implementación de herramientas para la caracterización de información genómica asociada a enfermedades raras mediante redes neuronales artificiales

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Escribano de la Torre, Yaiza

Tutor/a: Sempere Luna, José María

CURSO ACADÉMICO: 2023/2024

Resum

La retinosi pigmentària és una malaltia genètica rara que afecta la retina, provocant una pèrdua progressiva de la visió. A causa de la raresa d'aquesta condició, un dels principals desafiaments en la investigació és l'escassetat de dades disponibles per a entrenar models de predicció de mutacions patogèniques. Els conjunts de dades no són només petits sinó també desbalancejats, la qual cosa significa que hi ha una preponderància d'exemples d'una classe (per exemple, mutacions no patogèniques) sobre l'altra (mutacions patogèniques). Aquest desequilibri pot portar al fet que els models d'aprenentatge automàtic desenvolupen un biaix cap a la classe majoritària, reduint així la seua capacitat per a identificar correctament les mutacions patogèniques, que són precisament les de major interés clínic. A més, treballar amb dades limitades augmenta el risc de sobreajustament, on el model aprén a memoritzar els detalls de les dades d'entrenament, perdent capacitat de generalització a noves dades. Per a mitigar aquests problemes, es van implementar diverses tècniques de preprocessament i modelatge, com l'ús de la tècnica de sobremostratge combinada amb neteja de soroll (`SMOTETomek`) per a equilibrar les classes, l'addició de soroll a les dades per a evitar el sobreajustament, i l'aplicació de regularització L2 en les capes denses del model. Aquestes tècniques estaven dissenyades per a millorar la capacitat del model d'aprendre patrons rellevants sense dependre excessivament de les peculiaritats del conjunt d'entrenament.

Després de nombrosos intents i ajustos en l'arquitectura del model i la selecció d'hiperparàmetres, es va aconseguir desenvolupar un model de xarxa neuronal convolucional (CNN) que va demostrar un rendiment superior al de configuracions anteriors. El model final va aconseguir resultats notables, amb una precisió en la validació que va aconseguir un 0,799, la qual cosa indica que el model no sols s'adapta bé a les dades d'entrenament, sinó que també és capaç de realitzar prediccions precises en dades no vistes.

Aquests resultats són particularment significatius, donat que en experiments anteriors el model patia de sobreajustament o no aconseguia millorar la precisió en el conjunt de validació. La combinació de tècniques de preprocessament, regularització, i l'ús de la funció de pèrdua `BinaryFocalCrossentropy` van resultar crucials per a aconseguir aquests resultats. L'èxit del model suggereix que és possible desenvolupar solucions efectives per a la identificació de mutacions patogèniques en contextos de dades limitades i desbalancejades, aportant una contribució valuosa a la investigació genètica en malalties rares com la retinosi pigmentària.

Paraules clau: aprenentatge automàtic, malalties rares, xarxes neuronals, xarxes neuronals convolucionals, genètica computacional

Resumen

La retinosis pigmentaria es una enfermedad genética rara que afecta a la retina, provocando una pérdida progresiva de la visión. Debido a la rareza de esta condición, uno de los principales desafíos en la investigación es la escasez de datos disponibles para entrenar modelos de predicción de mutaciones patogénicas. Los conjuntos de datos son no solo pequeños sino también desbalanceados, lo que significa que hay una preponderancia de ejemplos de una clase (por ejemplo, mutaciones no patogénicas) sobre la otra (mutaciones patogénicas). Este desequilibrio puede llevar a que los modelos de aprendizaje automático desarrollen un sesgo hacia la clase mayoritaria, reduciendo así su capacidad para identificar correctamente las mutaciones patogénicas, que son precisamente las de mayor interés clínico. Además, trabajar con datos limitados aumenta el riesgo de sobreajuste (*overfitting*), donde el modelo aprende a memorizar los detalles de los datos de entrenamiento, perdiendo capacidad de generalización a nuevos datos. Para mitigar estos problemas, se implementaron diversas técnicas de preprocesamiento y modelado, tales como el uso de la técnica de sobremuestreo combinada con limpieza de ruido (`SMOTETomek`) para equilibrar las clases, la adición de ruido a los datos para evitar el sobreajuste, y la aplicación de regularización L2 en las capas densas del modelo. Estas técnicas estaban diseñadas para mejorar la capacidad del modelo de aprender patrones relevantes sin depender excesivamente de las peculiaridades del conjunto de entrenamiento.

Después de numerosos intentos y ajustes en la arquitectura del modelo y la selección de hiperparámetros, se logró desarrollar un modelo de red neuronal convolucional (CNN) que demostró un rendimiento superior al de configuraciones anteriores. El modelo final tuvo notables resultados, con una precisión en la validación que alcanzó un 0.799, lo que indica que el modelo no solo se adapta bien a los datos de entrenamiento, sino que también es capaz de realizar predicciones precisas en datos no vistos.

Estos resultados son particularmente significativos, dado que en experimentos anteriores el modelo sufría de sobreajuste o no lograba mejorar la precisión en el conjunto de validación. La combinación de técnicas de preprocesamiento, regularización, y el uso de la función de pérdida `BinaryFocalCrossentropy` resultaron cruciales para alcanzar estos resultados. El éxito del modelo sugiere que es posible desarrollar soluciones efectivas para la identificación de mutaciones patogénicas en contextos de datos limitados y desbalanceados, aportando una contribución valiosa a la investigación genética en enfermedades raras como la retinosis pigmentaria.

Palabras clave: aprendizaje automático, enfermedades raras, redes neuronales, redes neuronales convolucionales, genética computacional

Abstract

Retinitis pigmentosa is a rare genetic disease that affects the retina, leading to a progressive loss of vision. Due to the rarity of this condition, one of the main challenges in research is the scarcity of data available to train models for predicting pathogenic mutations. The datasets are not only small but also imbalanced, meaning there is a predominance of examples from one class (e.g., non-pathogenic mutations) over the other (pathogenic mutations). This imbalance can cause Machine Learning models to develop a bias towards the majority class, thereby reducing their ability to correctly identify pathogenic mutations, which are of the most clinical interest. Furthermore, working with limited data increases the risk of overfitting, where the model learns to memorize the details of the training data, losing its ability to generalize to new data. To mitigate these issues, various preprocessing and modeling techniques were implemented, such as the use of the `SMOTETomek` technique to balance the classes, adding noise to the data to prevent overfitting, and applying L2 regularization in the model's dense layers. These techniques were designed to improve the model's ability to learn relevant patterns without excessively relying on the peculiarities of the training set.

After numerous attempts and adjustments to the model's architecture and hyperparameter selection, a convolutional neural network (CNN) model was developed that demonstrated superior performance compared to previous configurations. The final model achieved notable results, with a validation accuracy reaching 0.799, indicating that the model not only adapts well to the training data but is also capable of making accurate predictions on unseen data.

These results are particularly significant, given that in previous experiments, the model suffered from overfitting or failed to improve accuracy on the validation set. The combination of preprocessing techniques, regularization, and the use of the BinaryFocalCrossentropy loss function proved crucial in achieving these results. The success of the model suggests that it is possible to develop effective solutions for identifying pathogenic mutations in contexts with limited and imbalanced data, contributing valuable insights to genetic research in rare diseases such as retinitis pigmentosa.

Key words: machine learning, rare diseases, neural networks, convolutional neural networks, computational genetics

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

1.1 Planteamiento del problema

La retinosis pigmentaria (RP) es una enfermedad genética rara que afecta a la retina y puede provocar una pérdida progresiva de la visión. Dado que se trata de una enfermedad poco común, el número de pacientes diagnosticados es limitado, al igual que la cantidad de datos genéticos disponibles. Esta limitación de datos presenta un desafío considerable para el análisis y la predicción de mutaciones patogénicas, que son cruciales para el diagnóstico, la prevención y el tratamiento de la enfermedad.

En el presente estudio, se han reunido exomas — la parte del ADN que codifica proteínas y alberga la mayoría de las variantes genéticas relevantes para enfermedades — de 55 pacientes diagnosticados con retinosis pigmentaria. Además, se cuenta con un documento detallado que especifica el diagnóstico genético de cada paciente, incluyendo la mutación causante cuando está disponible. Estos datos han sido preprocesados para ser utilizados en un modelo de red neuronal, cuyo objetivo es identificar mutaciones potencialmente patogénicas. No obstante, el principal desafío radica en la limitación de datos y en el desequilibrio entre la representación de mutaciones patogénicas y no patogénicas, lo que complica el entrenamiento eficaz del modelo.

La escasez de datos en una clase específica en tareas de clasificación binaria, como la identificación de mutaciones patogénicas, presenta un desafío significativo. En este caso, la clase minoritaria, es decir, las mutaciones patogénicas, está subrepresentada, lo que puede conducir a un sesgo en el modelo y a una baja capacidad de generalización. Esta dificultad se agrava en el contexto de enfermedades raras, donde la falta de datos puede limitar la eficacia de los modelos predictivos y su aplicabilidad en la práctica clínica.

Para abordar estos problemas, es esencial desarrollar estrategias que optimicen el rendimiento del modelo a pesar de la escasez de datos. Esto puede incluir técnicas avanzadas de preprocesamiento, la utilización de algoritmos de aprendizaje automático que manejen desequilibrios en los datos y enfoques que incorporen conocimientos biológicos adicionales para mejorar la capacidad predictiva del modelo.

1.2 Motivación

El estudio de enfermedades raras, como la retinosis pigmentaria, es crucial para avanzar en la comprensión de los mecanismos genéticos subyacentes y para desarrollar estrategias más efectivas de diagnóstico y tratamiento. No obstante, debido a su baja prevalencia, estas enfermedades suelen recibir menos atención y recursos en comparación con

condiciones más comunes. La limitada cantidad de pacientes y la escasez de datos restringen significativamente la investigación y el desarrollo de nuevas terapias.

En el caso específico de la retinosis pigmentaria, la escasez de datos representa un desafío adicional al trabajar con modelos de aprendizaje automático, como las redes neuronales. La dificultad para obtener una representación adecuada de la clase minoritaria, en este caso, las mutaciones patogénicas, complica el entrenamiento del modelo y puede llevar a una menor capacidad para identificar correctamente nuevas mutaciones patogénicas.

Este estudio tiene como objetivo abordar el reto de trabajar con conjuntos de datos pequeños y desbalanceados en el contexto de enfermedades raras. La investigación se centra en mejorar la identificación de mutaciones patogénicas mediante el desarrollo y la implementación de metodologías avanzadas que puedan superar las limitaciones de datos y desequilibrio en las clases. Además de contribuir al conocimiento científico sobre la retinosis pigmentaria, el estudio busca establecer enfoques que puedan ser aplicables a otras enfermedades raras con desafíos similares.

La relevancia social de esta investigación radica en su potencial para mejorar el diagnóstico temprano y la identificación de nuevas mutaciones, lo cual podría facilitar el desarrollo de tratamientos más efectivos y personalizados para los pacientes afectados, mejorando así su calidad de vida y proporcionando avances significativos en el campo de la genómica computacional.

1.3 Objetivos

El objetivo principal de este trabajo es desarrollar un modelo de red neuronal capaz de identificar mutaciones patogénicas en pacientes con retinosis pigmentaria, utilizando un conjunto de datos limitado y desbalanceado. Para alcanzar este objetivo, se han establecido los siguientes objetivos específicos:

- **Preprocesar los datos genéticos:** convertir los datos genómicos y de diagnóstico en un formato adecuado para el entrenamiento de redes neuronales, asegurando la calidad y consistencia de los datos. Este proceso incluye la limpieza de datos, la normalización y la preparación de características relevantes para el modelo.
- **Aplicar técnicas de sobremuestreo y ajuste de datos:** utilizar métodos como `SMOTETomek` para abordar el problema del desequilibrioen los datos, mejorando la representatividad de las mutaciones patogénicas en el conjunto de datos. Estas técnicas buscan equilibrar las clases y reducir el sesgo en el entrenamiento del modelo.
- **Desarrollar y entrenar una red neuronal convolucional:** crear un modelo de red neuronal que pueda aprender a partir de los datos preprocesados y ajustados para identificar mutaciones potencialmente patogénicas. Esto incluye la selección de la arquitectura adecuada, la configuración de hiperparámetros y el proceso de entrenamiento y validación del modelo.
- **Evaluar el rendimiento del modelo:** medir la eficacia del modelo entrenado en la identificación de mutaciones patogénicas utilizando métricas de evaluación como la precisión del entrenamiento, la precisión de validación, la pérdida de entrenamiento y la pérdida de validación.
- **Proponer mejoras y recomendaciones:** basado en el rendimiento del modelo, sugerir posibles mejoras en el enfoque de entrenamiento y preprocesamiento, así como en la interpretación de resultados. Estas recomendaciones estarán orientadas

a optimizar el modelo y su aplicabilidad en futuras investigaciones y aplicaciones clínicas.

La consecución de estos objetivos permitirá una mejor comprensión de las mutaciones asociadas con la retinosis pigmentaria y contribuirá al desarrollo de herramientas de diagnóstico más precisas y efectivas para enfermedades raras en general. Además, se espera que los enfoques y técnicas desarrollados en este trabajo puedan ser adaptados y aplicados a otros contextos y enfermedades raras, ampliando su impacto en la investigación y la práctica clínica.

CAPÍTULO 2
Estado del arte

En la última década, el campo de la biología computacional ha experimentado un crecimiento exponencial, impulsado por el avance de técnicas de inteligencia artificial, particularmente el aprendizaje profundo (*Deep Learning*). Este enfoque ha transformado la forma en que se analizan y comprenden los datos biológicos, facilitando la extracción de patrones complejos y la mejora en la precisión de las predicciones biológicas. Las redes neuronales profundas (DNNs) han sido fundamentales en este avance, proporcionando mejoras significativas en áreas como la predicción de la estructura tridimensional de proteínas, la identificación de regiones funcionales en el genoma y la interpretación de secuencias genéticas.

Uno de los avances más importantes en el aprendizaje profundo aplicado a la biología computacional es la predicción de la estructura de proteínas. La estructura tridimensional de una proteína es crucial para comprender su función biológica, ya que la actividad de una proteína está íntimamente relacionada con su conformación espacial. La predicción precisa de la estructura de proteínas puede acelerar significativamente el descubrimiento de fármacos y la comprensión de enfermedades, facilitando el diseño de terapias más efectivas y específicas.

Un avance significativo en este campo es el desarrollo de *AlphaFold*, una IA creada por *DeepMind* que ha revolucionado la predicción de la estructura de proteínas. *AlphaFold* ha logrado una precisión sin precedentes en la predicción de estructuras proteicas, superando los métodos tradicionales y proporcionando *insights* cruciales para la biología estructural [38].

En el ámbito de las enfermedades raras, la IA ha demostrado ser una herramienta valiosa para el diagnóstico temprano y el desarrollo de tratamientos personalizados. La integración de datos genómicos y clínicos mediante algoritmos de aprendizaje profundo ha facilitado la identificación de biomarcadores específicos para estas enfermedades, permitiendo diagnósticos más rápidos y precisos. El estudio de Vallejo *et al.* (2019) destaca cómo la IA puede analizar grandes volúmenes de datos para descubrir patrones que indican la presencia de enfermedades raras [64].

Además, la IA ha sido crucial en el desarrollo de tratamientos personalizados para enfermedades raras. La capacidad de analizar datos a gran escala permite a los investigadores identificar posibles dianas terapéuticas y desarrollar enfoques de tratamiento más específicos. La aplicación de modelos de IA en el análisis de datos de ensayos clínicos ha permitido encontrar patrones en la respuesta de los pacientes a diferentes tratamientos, mejorando la eficacia y reduciendo efectos secundarios [64].

El aprendizaje profundo ha demostrado ser particularmente efectivo en la biología computacional gracias a su capacidad para analizar datos biológicos complejos mediante redes neuronales convolucionales (CNNs) y redes neuronales recurrentes (RNNs). Estas

técnicas han logrado un rendimiento superior en tareas cruciales como la predicción de la expresión génica y la identificación de variantes genéticas funcionales [3]. Las CNNs, conocidas por su capacidad para detectar patrones espaciales en datos, se han utilizado para analizar secuencias de ADN y detectar motivos genéticos asociados con la regulación génica. Por otro lado, las RNNs han demostrado ser eficaces en el modelado de datos secuenciales, como la expresión génica bajo diversas condiciones celulares, capturando la dinámica temporal y espacial de los procesos biológicos.

La capacidad de integrar datos multimodales sigue siendo una de las contribuciones más destacadas del aprendizaje profundo en la biología computacional. Esto implica combinar diferentes tipos de datos biológicos, como secuencias de ADN, perfiles de expresión génica, datos epigenéticos y datos de proteómica, para construir modelos más precisos y completos de los sistemas biológicos. La integración de datos multimodales a través del aprendizaje profundo ha permitido avances en áreas como la medicina de precisión y la genómica funcional, facilitando, por ejemplo, la extracción de características esenciales mediante *autoencoders* para grandes conjuntos de datos genómicos.

A pesar de estos avances, la aplicación del aprendizaje profundo en biología computacional enfrenta desafíos significativos, como la necesidad de grandes volúmenes de datos etiquetados y la dificultad de interpretar modelos complejos. Los datos biológicos son frecuentemente escasos y costosos de obtener, limitando la aplicabilidad de estos métodos. Además, los modelos de aprendizaje profundo son a menudo considerados "cajas negras", lo que restringe su interpretabilidad y plantea retos en un campo donde entender los mecanismos subyacentes es crucial. Sin embargo, estos desafíos también presentan oportunidades para el desarrollo de nuevas metodologías, como el aprendizaje semi-supervisado y no supervisado, y la creación de modelos más interpretables. Estas áreas de investigación podrían ampliar significativamente el impacto del aprendizaje profundo en la biología computacional en los próximos años [3].

En resumen, el aprendizaje profundo ha revolucionado la biología computacional, facilitando avances significativos en el análisis e interpretación de datos biológicos. A medida que se desarrollen nuevas técnicas y se aborden los desafíos existentes, es probable que veamos una integración aún mayor de estas tecnologías en la investigación biomédica y la medicina de precisión. La contribución de *AlphaFold* y el impacto de la IA en el diagnóstico y tratamiento de enfermedades raras subrayan el potencial transformador de estas herramientas en la comprensión de sistemas biológicos complejos y en la mejora de la atención médica [38, 64].

CAPÍTULO 3
Conceptos básicos

3.1 Las enfermedades raras y la Inteligencia Artificial

Las enfermedades raras (ER) son aquellas enfermedades que tienen una prevalencia menor a 1:2000 individuos. Se estima que afecta al 6-8 % de la población y que existen entre 7000 y 8000 enfermedades raras diferentes. Suelen ser enfermedades crónicas, invalidantes y de origen genético en más del 80 % de los casos. La escasa incidencia conlleva un limitado entendimiento por parte del personal médico, resultando en diagnósticos tardíos y poco específicos. Diagnosticar correctamente a los pacientes es un desafío considerable: según una encuesta realizada en 2013 [23], en promedio, toma más de cinco años, la participación de ocho médicos y entre dos a tres diagnósticos incorrectos antes de que un paciente con una enfermedad rara obtenga un diagnóstico preciso. Incluso después de un diagnóstico correcto, los problemas continúan, ya que el bajo número de pacientes frecuentemente reduce los incentivos comerciales para desarrollar tratamientos específicos [59]. Este inconveniente, además, conlleva costos adicionales y afecta negativamente la calidad de vida de los pacientes y sus familias [14]. La inteligencia artificial (IA) y el aprendizaje automático tienen un gran potencial para mejorar el diagnóstico y tratamiento de las enfermedades raras. Pueden ayudar a abordar este desafío al procesar grandes cantidades de datos y reconocer patrones complejos que pueden no ser evidentes para los médicos humanos [59]. Las tecnologías de IA también tienen la capacidad de integrar y analizar datos de diversas fuentes. Esta capacidad resulta fundamental para afrontar problemáticas como las bajas tasas de diagnóstico, el limitado número de pacientes y su dispersión geográfica. En consecuencia, el conocimiento facilitado por la IA sobre las ER podría impulsar de manera significativa el desarrollo de nuevas terapias [10].

Actualmente, el *Machine Learning* ya está transformando la biomedicina mediante su aplicación en el diagnóstico, como la evaluación y clasificación de variantes patogénicas en secuencias genéticas, y en la investigación preclínica y el desarrollo de fármacos. Esto incluye la creación de nuevas moléculas, la reducción de pruebas en animales y el análisis de interacciones medicamentosas. Estos avances representan solo algunos de los beneficios que la IA ha aportado a la biomedicina. Muchas de estas herramientas son útiles para una amplia gama de enfermedades, incluidas las enfermedades raras [10].

3.2 Distrofias retinianas hereditarias

Las distrofias hereditarias de retina (DHR) son un grupo diverso de enfermedades raras que se caracterizan por la degeneración de las células sensibles a la luz de la retina y/o del epitelio pigmentario, lo que afecta la visión de diversas maneras. Estas enferme-

dades presentan una gran variabilidad clínica y genética, lo que complica su diagnóstico y tratamiento. Se consideran raras debido a su baja prevalencia, afectando a 1 de cada 3000 a 4000 personas. Las DHR son crónicas y progresivas, llevando a una pérdida gradual de la visión que puede culminar en ceguera legal, lo que afecta significativamente la calidad de vida y el bienestar emocional de los pacientes y sus familias [54].

Las DHR pueden dividirse en dos grandes categorías según las áreas afectadas de la retina: periféricas y centrales. Las retinopatías periféricas, como la retinosis pigmentaria (RP) y la coroideremia, afectan principalmente a los bastones, provocando síntomas como ceguera nocturna y reducción del campo visual periférico. Una forma temprana de retinopatía periférica es la Amaurosis Congénita de Leber (LCA), que se manifiesta desde el nacimiento o en la infancia temprana con una rápida pérdida de agudeza visual. Por otro lado, las retinopatías centrales, como la distrofia de conos, la acromatopsia y el monocromatismo de conos azules, afectan principalmente a los conos, resultando en fotofobia, alteraciones en la percepción del color y baja agudeza visual. Las distrofias maculares hereditarias, como la enfermedad de Stargardt y la distrofia macular viteliforme de Best, causan pérdida de visión central, a menudo con atrofia macular [54].

Aproximadamente entre el 15 % y el 20 % de las DHR están asociadas con otras afecciones sistémicas, conocidas como DHR sindrómicas. Estas incluyen síntomas adicionales como hipoacusia, discapacidad intelectual, malformaciones físicas y trastornos endocrinológicos, lo que puede aumentar la gravedad y la complejidad del cuadro clínico. El síndrome de Usher, que combina retinitis pigmentaria con pérdida auditiva, es la forma sindrómica más común de DHR, seguido por el síndrome de Bardet-Biedl y el síndrome de Alström, que también incluyen una variedad de síntomas multisistémicos [54].

3.3 Retinosis pigmentaria

La retinosis pigmentaria (RP) destaca como la forma más frecuente de DRH, se caracteriza por una disfunción primaria de los bastones seguida de la pérdida de los fotorreceptores cónicos. Inicialmente, esto se manifiesta con nictalopía y una reducción del campo visual, mientras que en etapas posteriores, la degeneración de los conos conduce a una disminución en la agudeza visual y la pérdida del campo visual central [58].

La RP afecta a aproximadamente 1 de cada 3,700 personas, aunque esta prevalencia puede variar según el país y la región. Se considera la principal causa de ceguera de origen genético en la población adulta. Esta condición tiene una mayor incidencia en los hombres, con alrededor del 60 % de los afectados siendo hombres y el 40 % mujeres.

El diagnóstico de retinosis pigmentaria se fundamenta en una historia clínica detallada y un examen exhaustivo del ojo. Se confirma la presencia de esta afección cuando se detecta afectación en ambos ojos, pérdida de la visión periférica y disfunción de los fotorreceptores, que se evidencia mediante alteraciones en el electrorretinograma [2].

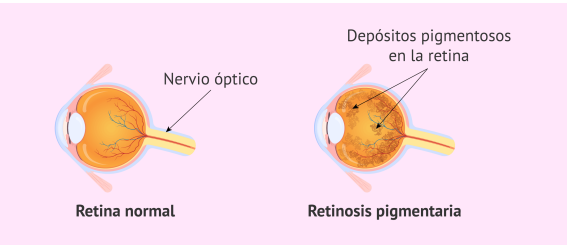

Figura 3.1: Ilustración de las diferencias entre una retina sin retinosis pigmentaria y una con [33].

CAPÍTULO 4

Inteligencia Artificial: *Machine Learning* y *Deep Learning*

La inteligencia artificial (IA) es un conjunto de técnicas que permiten a los ordenadores emular el comportamiento humano y realizar o incluso mejorar la toma de decisiones humanas en la resolución de tareas complejas, ya sea de manera autónoma o con una intervención mínima por parte de las personas. La IA aborda una variedad de problemas fundamentales como la representación del conocimiento, el razonamiento, el aprendizaje, la planificación, la percepción y la comunicación. Para ello, utiliza una serie de herramientas y métodos, como el razonamiento basado en casos, los sistemas basados en reglas, los algoritmos genéticos, los modelos difusos y los sistemas multiagente. En sus inicios, la investigación en IA se enfocó en codificar declaraciones en lenguajes formales, permitiendo que los ordenadores razonaran automáticamente mediante reglas de inferencia lógica, lo cual se conoce como el enfoque basado en conocimientos. Sin embargo, este método tiene sus limitaciones, ya que a menudo es difícil para los humanos expresar de forma explícita todo el conocimiento tácito necesario para realizar tareas complejas [35].

El aprendizaje automático (ML) resuelve algunas de las limitaciones mencionadas anteriormente. En esencia, el ML consiste en que un programa de ordenador mejora su rendimiento en una tarea específica a medida que adquiere experiencia, basándose en ciertos tipos de tareas y métricas de rendimiento. El objetivo del ML es automatizar la creación de modelos analíticos para realizar tareas cognitivas, como la detección de objetos o la traducción de lenguaje natural. Esto se logra mediante el uso de algoritmos que aprenden de forma iterativa a partir de datos de entrenamiento, lo que permite a los ordenadores descubrir patrones complejos y conocimientos ocultos sin necesidad de una programación explícita [35].

El ML es particularmente eficaz para manejar datos de alta dimensión, en tareas como la clasificación, la regresión y el agrupamiento. Esto se debe a su capacidad para identificar patrones y regularidades en grandes volúmenes de datos, lo que permite generar decisiones precisas y consistentes. Gracias a su habilidad para aprender de datos previos y detectar patrones, los algoritmos de ML se han utilizado con éxito en una amplia gama de aplicaciones, incluyendo la detección de fraudes, la evaluación crediticia, el análisis de ofertas, el reconocimiento de voz e imagen, y el procesamiento de lenguaje natural (NLP) [35].

El aprendizaje automático (ML) se clasifica en tres tipos principales: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Estos tipos de ML emplean varios algoritmos, como modelos de regresión, árboles de decisión y **redes neuronales**, cada uno adaptado a diferentes tareas de aprendizaje.

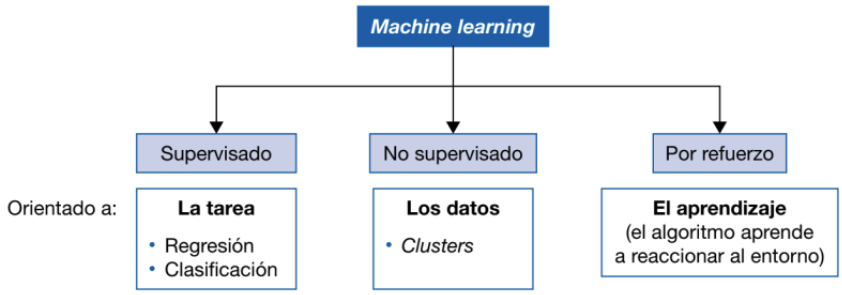

Figura 4.1: Clasificación de los grandes tipos de aprendizaje de algoritmos de *Machine Learning* [6].

Deep Learning, o aprendizaje profundo, es una técnica avanzada de aprendizaje automático que emplea redes neuronales artificiales con múltiples capas ocultas. Estas redes están organizadas en estructuras profundamente anidadas, permitiéndoles procesar datos de manera jerárquica. A medida que los datos pasan a través de las diferentes capas, la red aprende a identificar patrones complejos y características abstractas sin necesidad de intervención humana para definir qué buscar. Esta capacidad para descubrir automáticamente representaciones útiles hace que el aprendizaje profundo sea especialmente efectivo en tareas que implican grandes volúmenes de datos complejos, como imágenes, texto y audio. A diferencia de otros métodos de aprendizaje automático que requieren una definición previa de las características por parte de los humanos, el *Deep Learning* puede manejar datos en bruto y extraer la información relevante por sí mismo, logrando resultados más precisos en aplicaciones que demandan comprensión detallada y procesamiento de datos multidimensionales [35].

4.1 Aprendizaje supervisado

El aprendizaje automático es una disciplina que permite a las máquinas aprender de datos y mejorar su rendimiento en tareas específicas sin ser explícitamente programadas para cada caso [7]. Dentro del aprendizaje automático, el aprendizaje supervisado es una subdisciplina fundamental que se centra en la construcción de modelos predictivos a partir de datos etiquetados [49].

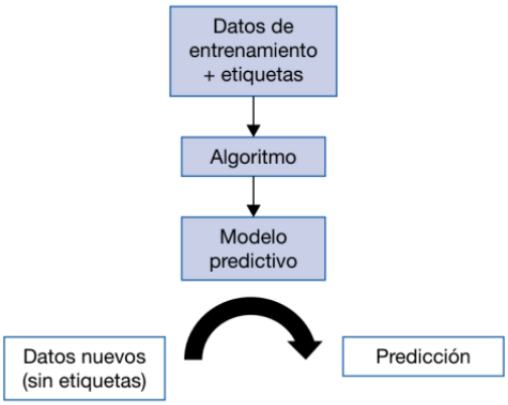

Figura 4.2: Descripción genérica del flujo de un algoritmo de aprendizaje supervisado [6].

En el **aprendizaje supervisado**, el proceso se estructura en dos fases principales: entrenamiento y prueba [7]. Durante la fase de **entrenamiento**, el modelo se alimenta con un conjunto de datos que incluye ejemplos con características y etiquetas conocidas. El objetivo en esta etapa es que el modelo aprenda a identificar patrones y relaciones entre las características de los datos y sus etiquetas correspondientes [7]. El modelo ajusta sus parámetros para minimizar la diferencia entre sus predicciones y las etiquetas reales, mejorando así su capacidad predictiva [24]. Una vez completado el entrenamiento, el modelo se evalúa en la fase de **prueba**. En esta etapa, se utilizan datos de prueba, que no están etiquetados, para verificar el rendimiento del modelo [7]. El modelo aplica lo aprendido durante el entrenamiento para hacer predicciones o clasificaciones sobre estos datos nuevos.

La efectividad del modelo se mide en función de su capacidad para generar predicciones precisas y fiables en comparación con los resultados esperados [26].

El aprendizaje supervisado se divide principalmente en dos categorías: **clasificación** y **regresión** [49].

- La **clasificación** se ocupa de predecir etiquetas discretas o categóricas, como identificar si una mutación es patogénica (etiqueta 1, por ejemplo) o no lo es (etiqueta 0) [26].
- La **regresión** se enfoca en predecir valores continuos, como estimar el precio de una vivienda en función de sus características [49].

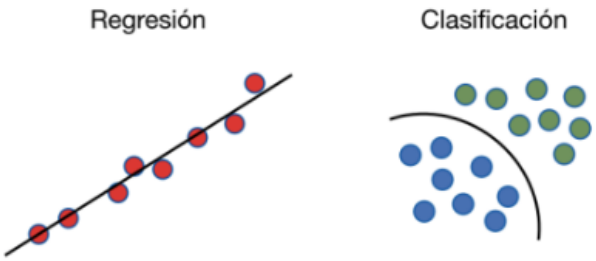

Figura 4.3: Representación gráfica del aprendizaje supervisado de un algoritmo de regresión en comparación con uno de clasificación [6].

En ambos casos, el aprendizaje supervisado busca construir un modelo predictivo que pueda generalizar y hacer predicciones precisas sobre datos no vistos [24].

4.1.1. Tipos de aprendizaje supervisado

Estos son algunos de los algoritmos de aprendizaje supervisado más populares:

- **Árboles de decisión:** es un modelo de clasificación que organiza el espacio de instancias de manera recursiva. Se compone de nodos que forman un árbol jerárquico, comenzando con un nodo raíz sin aristas entrantes. Los nodos internos, o nodos de prueba, dividen el espacio de instancias en subespacios basados en un atributo específico. Las hojas, que son los nodos terminales, representan las clases finales y pueden contener vectores de probabilidad que indican la probabilidad de que un atributo objetivo tome un valor determinado. La clasificación se realiza navegando desde la raíz hasta una hoja en función de los resultados de las pruebas [52].

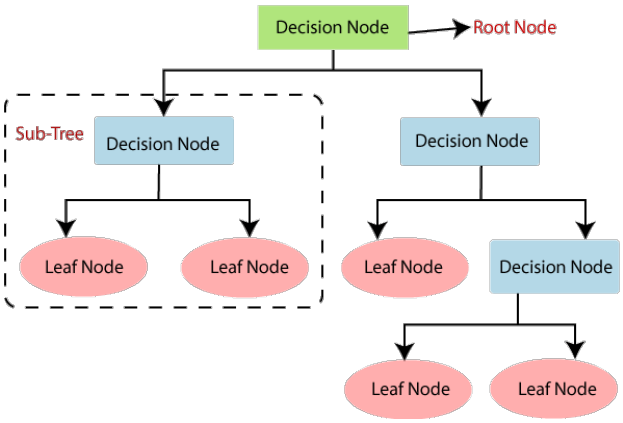

Figura 4.4: Esquema explicativo de un árbol de decisión [36].

- **Regresión lineal:** es un método estadístico dentro de la familia de algoritmos de regresión que modela la relación entre una variable dependiente continua (y) y una o más variables independientes (X). Utiliza una función lineal para ajustar los datos y minimizar una función de pérdida, con el objetivo de predecir valores de y para nuevas observaciones [52].

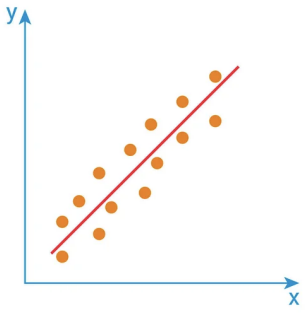

Figura 4.5: Ejemplo de Regresión lineal.

- ***Naive Bayes*:** es un método de aprendizaje supervisado basado en la teoría de probabilidad de Bayes. Asume una independencia condicional entre las características dadas una clase, lo que simplifica el cálculo de probabilidades. A pesar de esta suposición, que rara vez se cumple en la práctica, el clasificador Naive Bayes es efectivo y robusto frente al ruido en los datos, permitiendo resolver problemas de clasificación prediciendo la probabilidad de pertenencia a una clase específica [52].

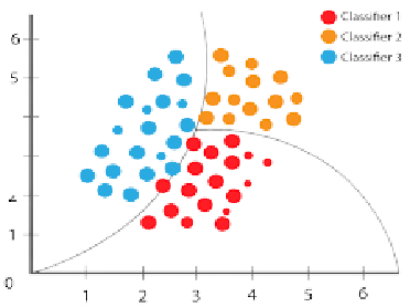

Figura 4.6: Ejemplo del algoritmo Naive Bayes [67].

La fórmula del clasificador *Naive Bayes* es la siguiente:

$$P(c \mid x) = \frac{P(x \mid c) \cdot P(c)}{P(x)}$$

- **Regresión logística:** es un modelo de regresión utilizado para predecir la probabilidad de ocurrencia de un evento, categorizando los datos en dos clases distintas. Este modelo utiliza la función sigmoide para transformar una combinación lineal de las variables independientes en una probabilidad en el rango de 0 a 1. Se entrena minimizando una función de costo mediante optimización, y es particularmente útil en problemas de clasificación binaria, donde se desea estimar la probabilidad de que una observación pertenezca a una clase específica [52].

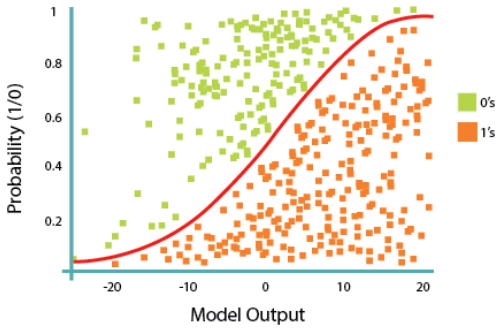

Figura 4.7: Ejemplo de Regresión logística [31].

4.2 Aprendizaje no supervisado

El **aprendizaje no supervisado** es un tipo de aprendizaje automático en el que los algoritmos trabajan con datos no etiquetados. A diferencia del aprendizaje supervisado, donde los ejemplos de entrenamiento vienen con etiquetas o salidas deseadas, en el aprendizaje no supervisado, los datos no tienen etiquetas y el objetivo es descubrir patrones o estructuras ocultas en los datos. Esto se logra a través de la agrupación o *clustering*, donde los datos se organizan en grupos basados en sus similitudes. Los prototipos de estos grupos se inicializan al azar y se ajustan iterativamente para mejorar su semejanza con los datos agrupados. El proceso continúa hasta que los prototipos convergen, maximizando las similitudes dentro de cada grupo. El aprendizaje no supervisado es especialmente útil para tareas como la agrupación de datos, donde el objetivo es identificar relaciones naturales entre los elementos sin necesidad de etiquetas previas [37].

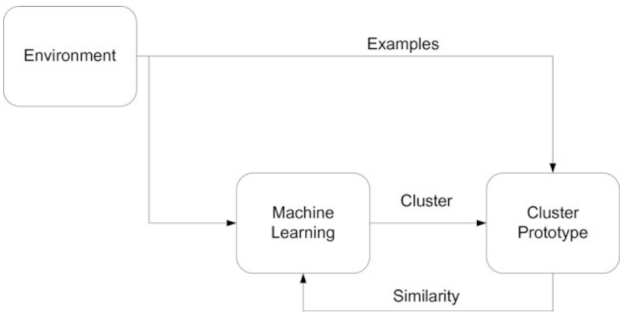

Figura 4.8: Aprendizaje no supervisado [37].

4.2.1. Tipos de aprendizaje no supervisado

Clustering

El ***clustering***, o análisis de agrupamiento, es una técnica en el aprendizaje no supervisado utilizada para clasificar elementos en grupos basados en sus características similares [50].

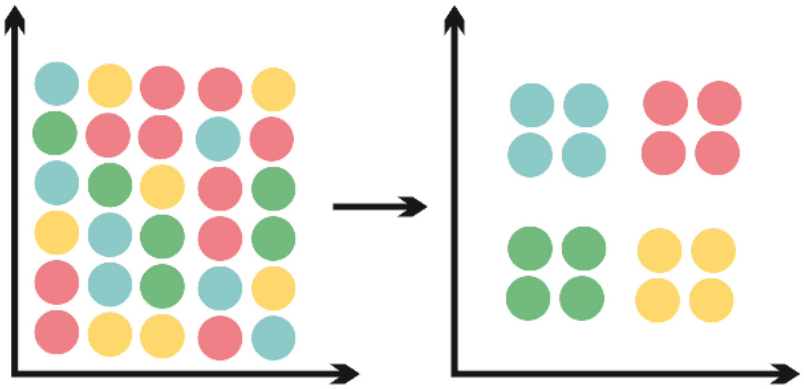

Figura 4.9: Ejemplo de *clustering* [50].

Existen varios tipos de *clustering*, que se describen a continuación:

- ***Clustering* exclusivo**: en este método, cada dato pertenece a un único grupo. Asegura que cada elemento esté asignado a un solo clúster sin superposiciones. Un ejemplo común es el algoritmo ***K-means***, que divide los datos en clústeres claramente definidos [50].
- ***Clustering* jerárquico**: este enfoque organiza los datos en una estructura jerárquica. Se puede dividir en dos tipos:
  - **Aglomerativo**: comienza con cada punto de datos como un grupo independiente y combina iterativamente los dos grupos más cercanos para formar una jerarquía de clústeres [50].
  - **Divisivo**: comienza con todos los datos en un solo grupo y divide progresivamente el grupo en subgrupos más pequeños [50].
- ***Clustering* superpuesto**: permite que un punto de datos pertenezca a dos o más grupos simultáneamente, con diferentes grados de pertenencia. Este método es útil para reflejar relaciones más complejas entre los datos [50].
- ***Clustering* probabilístico**: utiliza distribuciones de probabilidad para asignar datos a diferentes grupos. En lugar de asignar un único clúster a cada punto de datos, este enfoque asigna probabilidades de pertenencia a varios clústeres, proporcionando una representación más flexible [50].

Asociación

El Aprendizaje de Reglas de Asociación (ARL) es una técnica no supervisada para descubrir asociaciones entre variables en grandes conjuntos de datos. ARL puede manejar datos no numéricos y se utiliza para encontrar vínculos entre variables, como la

tendencia de los compradores de motocicletas a adquirir cascos. Estas asociaciones pueden monetizarse recomendando productos relacionados en tiendas en línea. ARL usa medidas como soporte y confianza para identificar patrones, donde el soporte mide la frecuencia de una asociación y la confianza evalúa la validez de la relación [50].

Detección de anomalías

La detección de anomalías busca identificar valores atípicos en un conjunto de datos, los cuales pueden señalar actividades inusuales, fallos en sensores o datos que necesitan limpieza. Por ejemplo, un patrón de tráfico de red irregular puede indicar que un sistema comprometido está enviando datos a un servidor no autorizado. Esta técnica es útil en áreas como la detección de intrusiones, seguros, fraude y vigilancia [50].

Autoencoders

Los ***autoencoders*** son redes neuronales diseñadas para aprender representaciones comprimidas de los datos. Utilizan una arquitectura con un "cuello de botella" que obliga a la red a reducir la dimensionalidad de la entrada y luego reconstruirla. Esta estructura ayuda a aprender patrones en los datos si existe alguna correlación entre los atributos. Sin el cuello de botella, la red simplemente memorizaría los datos de entrada [50].

4.3 Aprendizaje por refuerzo

El **aprendizaje por refuerzo** es un tipo de aprendizaje automático en el que un agente aprende a tomar decisiones optimizando sus acciones para maximizar recompensas y minimizar penalizaciones, basándose en la interacción con su entorno. En este proceso, el agente recibe entradas del entorno, genera una acción y, según la respuesta del entorno (ya sea una recompensa o una penalización), ajusta sus parámetros para mejorar su desempeño futuro. A diferencia de otros tipos de aprendizaje, como el supervisado o no supervisado, en el aprendizaje por refuerzo las experiencias de entrenamiento son variables y evolucionan con el tiempo, lo que permite al agente adaptarse dinámicamente a las condiciones cambiantes del entorno [37].

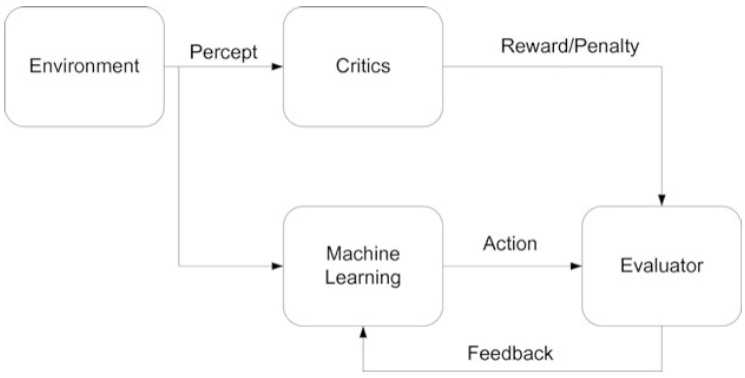

Figura 4.10: Aprendizaje por refuerzo [37].

4.4 *Deep Learning*

El ***Deep Learning*** (aprendizaje profundo) es una subárea del aprendizaje automático que se enfoca en el uso de redes neuronales artificiales con múltiples capas (llamadas redes neuronales profundas) para modelar patrones y representaciones complejas en los datos [41]. Este enfoque ha ganado popularidad en las últimas décadas debido a su capacidad para superar a otros métodos en una amplia variedad de tareas, desde el reconocimiento de imágenes hasta la comprensión del lenguaje natural [24].

4.4.1. Aplicaciones del *Deep Learning*

El *Deep Learning* ha demostrado ser particularmente eficaz en tareas como la visión por ordenador, el procesamiento de lenguaje natural y el reconocimiento de voz. En visión por ordenador, las redes neuronales convolucionales (CNN) se utilizan para la clasificación de imágenes, detección de objetos y segmentación de imágenes [40]. En el procesamiento del lenguaje natural, los modelos de redes neuronales recurrentes (RNN) y sus variantes, como LSTM y GRU, se han aplicado con éxito en tareas como la traducción automática, la generación de texto y el análisis de sentimientos [68].

CAPÍTULO 5

Marco teórico: redes neuronales artificiales y redes convolucionales

Dentro del marco del aprendizaje supervisado, las **redes neuronales** artificiales y las **redes convolucionales** son técnicas avanzadas que han demostrado ser altamente efectivas en la resolución de problemas complejos. Las redes neuronales artificiales, inspiradas en el funcionamiento del cerebro humano, utilizan una estructura de capas interconectadas de neuronas artificiales para modelar relaciones complejas en los datos. Por su parte, las redes convolucionales son una variante de las redes neuronales diseñadas específicamente para el procesamiento de datos con estructura espacial, como las imágenes. Estas redes aplican filtros convolucionales para extraer características importantes, mejorando la capacidad del modelo para reconocer patrones en datos visuales [52].

5.1 Redes neuronales artificiales

Los ordenadores modernos son capaces de realizar cálculos complejos de manera muy rápida, pero aún no igualan al cerebro humano en tareas perceptivas como el reconocimiento de imágenes y lenguaje. A diferencia de los ordenadores, que requieren entradas precisas y siguen instrucciones de forma secuencial, los cerebros humanos abordan las tareas de manera distribuida y en paralelo. Inspiradas en la biología, las redes neuronales artificiales (RNA) surgen como un diseño de inteligencia artificial que imita estas características [69].

5.1.1. Redes neuronales biológicas

Las redes neuronales artificiales (RNA) se inspiran en la estructura de las neuronas biológicas del cerebro humano. Este cerebro, compuesto por una gran red de neuronas interconectadas, realiza tareas complejas como el reconocimiento de imágenes y lenguaje con notable velocidad y precisión. Mientras que una sola neurona procesa información en milisegundos, el cerebro completa tareas como reconocer un rostro en apenas unos cientos de milisegundos, lo que indica que lo hace en aproximadamente cien pasos computacionales, en comparación con los millones que requiere un ordenador [69].

Esta eficiencia radica en que la información no se transmite de forma completa entre neuronas; en su lugar, está codificada en la red de conexiones neuronales, un proceso conocido como conexionismo. Una neurona se compone de dendritas, un cuerpo celular (soma) y un axón. Las dendritas reciben señales, el soma procesa la información, y el axón transmite las señales a través de sinapsis. Cuando la señal recibida supera un umbral, la

neurona se activa y envía una señal electroquímica, facilitando la comunicación en la red [69].

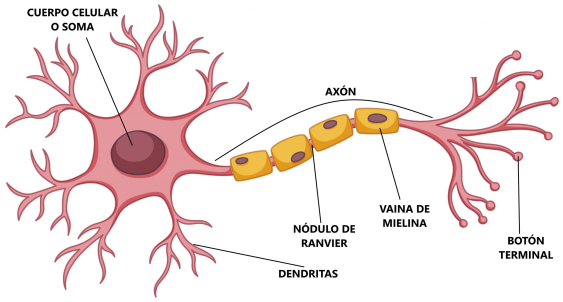

Figura 5.1: Ilustración de una neurona biológica [1].

5.1.2. Visión general de una red neuronal artificial

Una red neuronal artificial está modelada según la estructura de las neuronas biológicas. Similar a estas, una RNA es una red de nodos interconectados, que funcionan como las neuronas. Una RNA tiene tres componentes fundamentales: el carácter del nodo, la topología de la red y las reglas de aprendizaje.

- **Carácter del nodo:** define cómo se procesan las señales dentro de cada nodo, incluyendo el número de entradas y salidas, los pesos asociados a estas, y la función de activación. Cuando la suma ponderada de las entradas supera un umbral, el nodo se activa y transmite la señal a través de una función de transferencia [69].

- **Topología de la red:** describe la organización y conexión de los nodos, que se estructuran en capas: de entrada, ocultas y de salida. El diseño de la topología implica determinar el número de nodos por capa, la cantidad de capas y cómo se conectan entre sí [69].

- **Aprendizaje:** consiste en el ajuste los pesos de las conexiones para optimizar el rendimiento de la red. En el aprendizaje supervisado, se utilizan ejemplos de entrada y salida para entrenar la red y minimizar errores. En el aprendizaje no supervisado, la red intenta descubrir patrones en los datos de entrada sin tener salidas objetivo predefinidas [69].

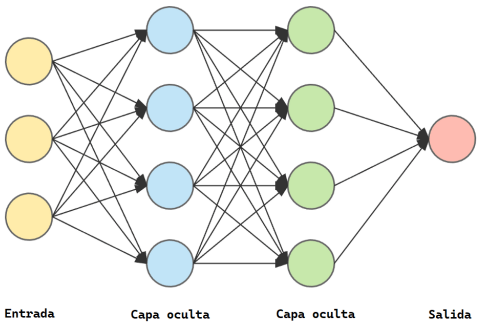

Figura 5.2: Esquema explicativo de una red Neuronal con tres nodos de entrada, dos capas ocultas de 4 nodos cada una y un nodo de salida [56].

Tipos de redes neuronales

A lo largo de los años, se han desarrollado diferentes tipos de redes neuronales para abordar diversos problemas, cada una con características y aplicaciones específicas. A continuación, se describen algunos de los tipos más importantes:

- **Perceptrón multicapa (MLP):** es una de las formas más básicas de red neuronal. Está compuesto por una capa de entrada, una o más capas ocultas y una capa de salida. Cada neurona en una capa está conectada a las neuronas de la siguiente capa a través de pesos sinápticos. El MLP es capaz de resolver problemas que no son linealmente separables mediante la función de activación no lineal, como la función sigmoide o la ReLU (*Rectified Linear Unit*). Este tipo de red se utiliza principalmente para tareas de clasificación y regresión simple [27].
- **Redes neuronales convolucionales (CNN):** son una clase especial de redes neuronales diseñadas para procesar datos que tienen una estructura de cuadrícula, como imágenes. Estas redes se componen de capas convolucionales, que aplican filtros sobre la entrada, detectando características como bordes, texturas y patrones más complejos a medida que se avanza en las capas. Las CNN han sido fundamentales en el desarrollo de aplicaciones de visión por ordenador, como el reconocimiento de objetos y la segmentación de imágenes [42].
- **Redes neuronales recurrentes (RNN):** están diseñadas para manejar datos secuenciales, como series temporales o texto. A diferencia de las redes *feedforward* como las MLP, las RNN tienen conexiones cíclicas que les permiten mantener un "estado" o memoria de las entradas anteriores, lo cual es crucial para tareas donde el contexto temporal es importante. Sin embargo, las RNN tradicionales tienen problemas con las dependencias a largo plazo debido al problema del desvanecimiento del gradiente, por lo que se han desarrollado variantes como LSTM (*Long Short-Term Memory*) y GRU (*Gated Recurrent Unit*) para mitigar estos problemas [30].
- **Redes generativas antagónicas (GAN):** consisten en dos redes neuronales que compiten entre sí: una red generadora, que crea datos falsos, y una red discriminadora, que intenta distinguir entre los datos reales y los generados. El objetivo es que la red generadora mejore hasta que el discriminador no pueda diferenciar entre datos reales y generados. Las GANs han sido muy exitosas en la **generación de imágenes realistas, videos, y otros** tipos de datos sintéticos [25].
- **Redes neuronales de transformadores:** son un tipo de red neuronal introducido en 2017 que ha revolucionado el procesamiento del lenguaje natural (NLP). A diferencia de las RNN, los Transformadores no procesan los datos secuencialmente, sino que utilizan mecanismos de atención para capturar las relaciones entre las palabras en un texto, independientemente de su posición en la secuencia. Esto ha permitido mejorar significativamente las tareas de **traducción automática**, **resumen de texto**, y **modelado de lenguaje** [66].

5.2 Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son una clase especializada de redes neuronales diseñadas para procesar datos que tienen una estructura de cuadrícula, como imágenes. Introducidas inicialmente por Yann LeCun en la década de 1990 [42], las CNN han demostrado ser extremadamente efectivas en tareas relacionadas con la visión por ordenador, y se han convertido en una piedra angular de los avances en IA.

Estructura básica de una CNN

Una CNN está compuesta principalmente por tres tipos de capas: las capas convolucionales, las capas de *pooling*, y las capas completamente conectadas.

- **Capas convolucionales:** estas capas aplican un conjunto de filtros (***kernels***) sobre la entrada para producir mapas de características (*feature maps*). La operación de convolución permite que la red sea capaz de aprender representaciones jerárquicas, donde capas más profundas detectan características más complejas [42].

- **Capas de *pooling*:** después de cada capa convolucional, es común incluir una capa de *pooling*, que reduce las dimensiones espaciales del mapa de características. Esto se realiza mediante operaciones como el ***max-pooling***, que selecciona el valor máximo de una ventana de la matriz de entrada. El *pooling* permite que la red sea invariante a pequeñas traslaciones en la entrada y reduce la carga computacional [9].

- **Capas completamente conectadas:** estas capas, ubicadas típicamente al final de la red, funcionan de manera similar a las capas en un Perceptrón Multicapa (MLP). Convierten los mapas de características en una salida de tamaño fijo, como una distribución de probabilidad sobre diferentes clases en un problema de clasificación. Cada neurona en estas capas está conectada a todas las neuronas en la capa anterior [42].

Funciones de activación

Las funciones de activación introducen no linealidades en la red, lo que permite que las CNN modelen relaciones complejas en los datos. La función de activación más utilizada en CNN es la ReLU (*Rectified Linear Unit*), definida como $f(x) = \max(0, x)$. La ReLU ayuda a prevenir el problema del desvanecimiento del gradiente, acelerando el proceso de entrenamiento al mantener un flujo de gradientes robusto [51].

Entrenamiento de las CNN

El entrenamiento de una CNN se realiza a través del algoritmo de retropropagación o *backprop*, donde los errores se propagan desde la capa de salida hacia las capas anteriores, ajustando los pesos de los filtros y las conexiones completamente conectadas. El objetivo es minimizar una función de pérdida, como la entropía cruzada en problemas de clasificación.La **optimización estocástica del gradiente** (SGD) es una técnica comúnmente utilizada para este proceso [8].

Aplicaciones de las CNN

Las CNN han sido fundamentales en el avance de diversas aplicaciones, entre las cuales destacan:

- **Reconocimiento de imágenes:** una de las aplicaciones más destacadas de las CNN es el reconocimiento de imágenes, como el caso del conjunto de datos *ImageNet*, donde modelos de CNN como *AlexNet* lograron un rendimiento sin precedentes [40].

- **Detección de objetos:** las CNN también se utilizan en la detección de objetos, donde no solo se reconoce qué objeto está presente en una imagen, sino también su ubicación exacta. Modelos como YOLO (*You Only Look Once*) han sido desarrollados específicamente para esta tarea [57].
- **Segmentación semántica:** en segmentación semántica, las CNN se utilizan para asignar una etiqueta a cada píxel en una imagen, lo cual es crucial en aplicaciones como la conducción autónoma y la medicina [45].

Avances recientes y futuro de las CNN

Con el avance de la tecnología, las CNN han evolucionado significativamente. Innovaciones como las redes neuronales profundas (DNN), las redes neuronales residuales (*ResNets*) [29], y las arquitecturas convolucionales eficientes como *MobileNet* [32], han permitido que las CNN sean más profundas, precisas y eficientes. Se espera que en el futuro, las CNN sigan siendo una herramienta clave en el desarrollo de nuevas aplicaciones de inteligencia artificial.

5.3 Descripción del modelo utilizado

En este trabajo, se ha implementado un modelo de red neuronal convolucional (CNN) utilizando las librerías `TensorFlow` y `Keras` para abordar el problema de clasificación. A continuación, se describen los pasos seguidos en la construcción y entrenamiento del modelo.

5.3.1. Elección del modelo

Las CNN son comúnmente utilizadas en el procesamiento de imágenes, como se ha mencionado previamente, debido a su capacidad para identificar patrones espaciales y locales. Sin embargo, en este estudio, una CNN demostró ser efectiva para analizar datos tabulares que, aunque no son imágenes, contienen una estructura que puede beneficiarse de la arquitectura convolucional.

El *dataset* utilizado presenta atributos que incluyen la posición de variaciones genéticas en los cromosomas, alelos de referencia y alternativos, tipo de variación genética, y consecuencias codificantes, entre otros. Estos datos, aunque tabulares, presentan relaciones espaciales y locales que las CNNs pueden capturar eficientemente. Por ejemplo:

- **Patrones espaciales en los datos:** los atributos como `POS` y `CHROM` pueden ser interpretados como si tuvieran una disposición espacial. Las redes convolucionales son capaces de identificar patrones locales entre estas características, de manera similar a cómo identifican patrones en imágenes.
- **Transformación de los datos:** los datos fueron transformados en representaciones numéricas, como *one-hot encodings* o *embeddings*. Esto permitió que las CNNs interpretaran estos datos como una especie de "pseudoimagen" donde cada atributo puede ser tratado como un canal o dimensión espacial.
- **Captura de interacciones locales:** las CNNs son eficaces en la captura de interacciones entre características vecinas. En el contexto genómico, esto significa que la red puede identificar relaciones significativas entre variaciones genéticas que están próximas entre sí.

- **Reducción de la dimensionalidad y regularización:** las capas convolucionales y de *pooling* ayudan a reducir la dimensionalidad del espacio de características, extrayendo las más relevantes y reduciendo el riesgo de *overfitting*, lo que es especialmente útil en datasets con alta dimensionalidad como los de genómica.

Este enfoque ha demostrado ser eficaz para la clasificación de variaciones genéticas en este estudio, superando los resultados de las redes neuronales convencionales.

5.3.2. Preprocesamiento de datos

Para mejorar la calidad y diversidad de los datos, se aplicaron varias técnicas de preprocesamiento:

- **Sobremuestreo y limpieza de ruido:** se utilizó el método `SMOTETomek` para combinar el sobremuestreo de las clases minoritarias con la limpieza de ruido [11]. Este método permite balancear las clases en el conjunto de datos al generar nuevos ejemplos sintéticos y eliminar ejemplos ruidosos. Esto mejora la calidad de los datos.
- **Adición de ruido:** se añadió ruido a los datos aumentados para hacer el modelo más robusto. La función `add_noise` genera ruido *gaussiano* (distribuido normalmente) y lo suma a los datos, luego recorta los valores para que estén en el rango [0,1]. Este ruido se caracteriza por ser añadido intencionalmente bajo condiciones específicas, de manera controlada y manejable; a diferencia del ruido anterior que se quiere eliminar, que se refiere a datos potencialmente erróneos o irrelevantes [43].
- **Concatenación y división de datos:** los datos originales y los datos aumentados se concatenaron para formar el conjunto final. Posteriormente, se dividió el conjunto de datos en entrenamiento y validación utilizando un 20 % de los datos para validación [20].
- **Redimensionamiento de datos:** los datos fueron redimensionados para que pudieran ser procesados por la red neuronal convolucional. En este caso, se añadió una dimensión adicional a los datos de entrada para adaptarlos a la estructura requerida por la CNN.

5.3.3. Construcción del modelo

El modelo de CNN se construyó utilizando la biblioteca *Keras* de *TensorFlow* y se compone de las siguientes capas:

- **Capas convolucionales y de normalización:** se añadieron capas convolucionales con funciones de activación ReLU, seguidas de capas de normalización por lotes y de *max pooling* para reducir la dimensionalidad y evitar el sobreajuste. Se emplearon tres bloques de capas convolucionales con diferentes tamaños de filtros y funciones de activación [42].
- **Capas densas y regularización:** después de aplanar las características extraídas por las capas convolucionales, se añadieron capas densas con funciones de activación **ReLU** y **regularización L2** para controlar el sobreajuste. Se incluyó una capa de salida con una activación **sigmoide** para la clasificación binaria [27].
- **Compilación del modelo:** el modelo se compiló utilizando el optimizador `Adam` con una tasa de aprendizaje de 0.0005 y la pérdida `Binary Focal Crossentropy` con un

parámetro gamma de 2.0 para manejar el desequilibrio de clases [44]. Además, se implementó el `Early Stopping` para evitar el sobreentrenamiento [55].

- **Entrenamiento del modelo:** el modelo fue entrenado durante 50 épocas con un tamaño de lote de 64, usando el conjunto de datos de entrenamiento y validación. Se asignaron pesos de clase balanceados para tratar el desequilibrio en el conjunto de datos [18].

5.3.4. Evaluación y visualización

Tras el entrenamiento, se evaluó el rendimiento del modelo en el conjunto de validación y se guardó el modelo entrenado. Se realizaron gráficos de la pérdida y la precisión durante el entrenamiento y la validación para visualizar el desempeño del modelo [34].

5.4 Librerías utilizadas

El proyecto hizo uso de varias librerías y herramientas para la implementación del modelo y el procesamiento de datos:

- **imblearn:** la librería `imblearn` proporciona herramientas para el manejo de desequilibrio en los datos. En este proyecto, se utilizó la clase `SMOTETomek` para aplicar sobremuestreo de las clases minoritarias y limpieza de ruido en el conjunto de datos [16].
- **scikit-learn:** la biblioteca `scikit-learn` ofrece funcionalidades para la manipulación y evaluación de datos. En este caso, se utilizó `compute_class_weight` para calcular los pesos de las clases balanceadas y `StratifiedKFold` para realizar la validación cruzada [19].
- **TensorFlow y Keras:** `TensorFlow` junto con `Keras` se usaron para construir y entrenar la red neuronal convolucional. Se emplearon las funciones `BinaryFocalCrossentropy` para la pérdida, `l2` para la regularización, y `EarlyStopping` para detener el entrenamiento en caso de sobreajuste [21].
- **NumPy:** la librería `NumPy` fue utilizada para la manipulación y procesamiento de arreglos numéricos, incluyendo la adición de ruido a los datos [17].
- **Matplotlib:** para la visualización de los resultados del entrenamiento, se utilizó `Matplotlib` para crear gráficos que muestran la pérdida y la precisión durante el entrenamiento y la validación [34].
- **Pandas:** es una biblioteca de *software* escrita en Python para la manipulación y análisis de datos. Proporciona estructuras de datos y operaciones para trabajar con datos estructurados y tablas, como hojas de cálculo y bases de datos. La biblioteca se ha convertido en una herramienta fundamental en el ecosistema de análisis de datos en Python debido a su flexibilidad y rendimiento. Ha sido crucial para el preprocesado de datos en el proyecto [46].

CAPÍTULO 6
Materiales

En el desarrollo y análisis de este estudio, se utilizaron varios formatos de archivos para almacenar y gestionar los datos necesarios. Cada formato tiene características específicas que lo hacen adecuado para diferentes tipos de información y análisis. A continuación se describen los formatos de archivo utilizados en este trabajo, detallando sus aplicaciones y la importancia de cada uno en el contexto de la investigación.

Formato `XLSX`: es un formato de archivo utilizado por Microsoft Excel para almacenar hojas de cálculo. Este formato permite organizar datos en tablas, facilitando su análisis y manipulación. El archivo `XLSX` utilizado en este estudio contiene datos tabulados de pacientes, incluyendo información genética y clínica [47].

Formato `VCF` **(*Variant Call Format*)**: es un formato estándar de archivo utilizado en bioinformática para almacenar variaciones genéticas detectadas en estudios de secuenciación de ADN. Los archivos `VCF` contienen información detallada sobre las variantes genéticas, como su posición en el genoma, la secuencia de referencia y la alteración observada. Este formato es esencial para la gestión de grandes volúmenes de datos genómicos y para realizar análisis posteriores [15].

Formato `TXT`: es un formato de texto plano que permite almacenar información sin formato en líneas de texto. En este estudio, los archivos `TXT` contienen anotaciones detalladas de las variaciones genéticas identificadas en los pacientes. Estas anotaciones incluyen información complementaria de bases de datos externas y detalles que no están presentes en los archivos `VCF` [22].

6.1 Archivo `XLSX`

El archivo `ResultadoExomas.xlsx` es un archivo en formato `XLSX` que contiene información detallada sobre 55 pacientes diagnosticados con DHR. Este archivo centraliza los datos obtenidos a partir de la secuenciación completa del exoma (`WES`, por sus siglas en inglés) realizada en el IIS La Fe de Valencia. Los datos incluyen el identificador del paciente, el diagnóstico clínico, las mutaciones detectadas a nivel de ADN complementario (`cDNA`) y a nivel proteico, el gen afectado, el genotipo, la confirmación del diagnóstico genético y anotaciones adicionales.

Cada paciente está identificado con un código único en la columna `ID Paciente`, utilizando el formato `RP-XXXX` o `RPN-XXXX`. Este formato asegura la anonimidad de los pacientes y facilita su seguimiento a lo largo del estudio.

La columna `Diagnóstico Clínico` detalla la patología diagnosticada a cada paciente a través de un examen físico. En este archivo, 42 pacientes fueron diagnosticados con

distrofia de retina y 13 con Síndrome de Usher, un trastorno caracterizado por distrofia de retina, pérdida auditiva y problemas de equilibrio.

En cuanto a los datos genéticos, las columnas `Mutaciones`, `Gen` y `Genotipo` proporcionan información crucial. La columna `Mutaciones` describe la variación genética responsable de la patología, en dos formatos posibles: basado en la secuencia de referencia del ADN codificante (`c.xxxx`) o en la secuencia de referencia proteica (`p.xxxxx`), ambos siguiendo la nomenclatura recomendada por la *Human Genome Society* (HGVS). La columna `Gen` especifica el gen afectado, de acuerdo con la nomenclatura del *HUGO Gene Nomenclature Committee* (HGNC). La columna `Genotipo` indica si la variación está presente en uno o ambos alelos del gen, clasificándose como homocigoto (`Hom`) si está presente en ambos, o heterocigoto (`Het`) si está presente solo en uno.

Finalmente, la columna `Familia` señala si existe alguna relación de parentesco entre los pacientes, identificando en este caso cinco pares de hermanos.

ID Paciente	Diagnóstico Clínico	Mutaciones	Gen	Genotipo	Familia
RP637	Distrofia de Retina	p.(Arg303His)	AGBL5	Het	Hermanos
RP639	Distrofia de Retina	p.(Gly522Aspfs*4)	AGBL5	Het	-

Tabla 6.1: Subgrupo de ejemplo constituido por 2 de los 55 pacientes de `ResultadoExomas.xlsx`

6.2 Ficheros `VCF`

Para cada uno de los 55 pacientes mencionados en el archivo `Resultado_Exomas.xlsx`, se generó un archivo en formato `VCF` (Variant Call Format). Estos archivos `VCF` contienen todas las variaciones genéticas identificadas en sus estudios de secuenciación del exoma (`WES`). En promedio, se identificaron aproximadamente 150,000 variaciones genéticas por paciente. Los nombres de estos archivos combinan el `ID` del paciente con el texto `full_variant_table`.

El formato `VCF` se compone de tres bloques principales: meta-información, encabezado y datos de variantes. La meta-información proporciona contexto sobre el archivo, incluyendo detalles sobre el formato, la tecnología de secuenciación utilizada y el panel de genes analizados. El encabezado especifica los nombres de las columnas que describen las características de las variantes genéticas. Finalmente, el bloque de datos incluye todas las variaciones detectadas, organizadas en líneas individuales que describen cada variación.

```
##FORMAT=<ID=AD,Number=2,Type=Integer,Description="Allelic depth for the ref and alt alleles in the order listed">
##FORMAT=<ID=DP4,Number=4,Type=Integer,Description="#ref plus strand, #ref minus strand, #alt plus strand, #alt minus strand">
##ALT=<ID=DEL,Description="Deletion">
#CHROM  POSID  REF ALT QUAL  FILTER  INFO  FORMAT  31856-0008-S1_RP-50
1  12167  82385886  A  G  1033  low_variant_fraction
VARTYPE=1;BGN=0.00193212;ARL=150;DER=68;DEA=66;QR=38;QA=38;PBP=537;PBM=508;NVF=0.163;TYPE=SNP;DBXREF=dbSNP:rs112125468;
```

Figura 6.1: Ejemplo de estructura de un archivo `VCF`, mostrando líneas de meta-información, encabezado y datos de variantes.

En la imagen anterior, se observan dos líneas de meta-información, que comienzan con `##`, proporcionando detalles sobre el formato de los datos genéticos en el archivo `VCF`. La línea de encabezado, que comienza con `#`, enumera las columnas que se completarán en las líneas de datos. Las dos últimas líneas pertenecen al bloque de datos y muestran una variación genética específica en el cromosoma 1, donde el nucleótido `A` ha

sido reemplazado por G en la posición 12167, con una alta calidad de llamada de variante y varios detalles adicionales sobre la naturaleza y calidad de la variación.

6.3 Ficheros TXT

Además de los archivos VCF, la mayoría de los pacientes en Resultado_Exomas.xlsx tienen un archivo TXT correspondiente que contiene anotaciones detalladas de sus variaciones genéticas. Estas anotaciones son complementarias a la información en el archivo VCF e incluyen datos adicionales de fuentes externas, como ClinVar. Los nombres de los archivos TXT coinciden con los de los archivos VCF.

Cada variación anotada en el archivo TXT está acompañada de 58 atributos, con valores desconocidos dejados en blanco. Algunos de estos atributos coinciden con los del archivo VCF, como sgid-ID, chromosome-CHROM, genome_position-POS, ref-REF y alt-ALT. Sin embargo, el campo FILTER en el archivo VCF, que indica la calidad de la llamada de variante con PASS, se representa en el TXT con un punto (.). El campo id en el archivo TXT es un identificador único para cada variación en un paciente, mientras que sgid es un identificador compartido. Dos pacientes diferentes pueden tener el mismo sgid para la misma variación, pero no necesariamente el mismo id.

Es importante mencionar que nueve pacientes no disponen de un archivo TXT; en su lugar, las anotaciones se encuentran en el atributo INFO del archivo VCF. Estas anotaciones están formateadas de manera diferente y contienen variables distintas en comparación con los archivos TXT.

CAPÍTULO 7
Preprocesado de los datos

En este estudio se distinguen dos grupos principales de pacientes: aquellos con diagnóstico genético y aquellos sin diagnóstico genético. El primer grupo, representado en el archivo `ResultadoExomas.xlsx`, incluye atributos que identifican la mutación causante de la enfermedad, como el gen afectado y la proteína o secuencia de ADN codificante alterada. Esta información está disponible para 14 de los 55 pacientes y es crucial para la fase de entrenamiento del modelo de *Machine Learning*.

De los 14 pacientes con diagnóstico genético, solo 8 tienen también un archivo `TXT` asociado. Estos archivos `TXT` contienen anotaciones detalladas de las variaciones genéticas, proporcionando información esencial para la evaluación y validación del modelo. Por lo tanto, solo estos 8 pacientes serán considerados en el grupo con diagnóstico genético para este estudio.

Para el entrenamiento de un modelo de *Machine Learning* (ML) es esencial disponer de dos grupos de muestras: el grupo de entrenamiento y el grupo de predicción. El grupo de entrenamiento estará compuesto por variaciones genéticas con una etiqueta binaria asociada (`0` o `1`), que indicará si la variación es patogénica (causante de la enfermedad) o no patogénica. Este grupo es fundamental para que el modelo aprenda a distinguir entre variaciones patogénicas y no patogénicas.

El segundo grupo, conocido como grupo de predicción, se utilizará una vez que el modelo haya sido entrenado. Las variaciones en este grupo no tendrán etiquetas, y el objetivo será que el modelo, basado en lo aprendido durante el entrenamiento, asigne una etiqueta a cada variación, determinando así su posible patogenicidad.

Para desarrollar un modelo de *Machine Learning* robusto y preciso, es esencial que las muestras del grupo de entrenamiento sean representativas y estén equilibradas en cuanto a variaciones patogénicas y no patogénicas. También es fundamental que los datos sean preprocesados adecuadamente para eliminar cualquier sesgo y normalizar las características, asegurando así un entrenamiento efectivo del modelo y su capacidad de generalización a nuevas variaciones.

El preprocesado de datos es vital para garantizar la veracidad y representatividad de la información. Por ello, los datos de entrada deben pasar por varias etapas de modificación, que incluyen:

1. **Extracción y limpieza**: esta etapa consiste en obtener los datos relevantes de los archivos `VCF` y `TXT`, eliminando cualquier información irrelevante o errónea.

2. **Fusión de archivos `VCF` y `TXT`**: se integran los datos de ambos archivos para consolidar toda la información disponible sobre cada variación genética.

3. **Agrupación de variaciones y eliminación de duplicidades**: se agrupan las variaciones similares y se eliminan las duplicadas para evitar redundancias que podrían sesgar el modelo.

4. **Selección y transformación de datos de entrada**: se eligen las características más relevantes y se transforman adecuadamente para que sean utilizadas por el modelo de *Machine Learning*.

Este proceso de preprocesado asegura que los datos utilizados para entrenar el modelo sean de alta calidad y representen fielmente la realidad, permitiendo así al modelo aprender de manera efectiva y realizar predicciones precisas sobre nuevas variaciones genéticas. Este enfoque metodológico es fundamental para el éxito del proyecto y la validez de los resultados obtenidos.

Para asegurar la validez de cada etapa del preprocesado, se realizarán estudios específicos para evaluar la calidad de los datos, como el análisis de la completitud de cada atributo y la identificación de duplicidades. Estos estudios garantizarán que los datos procesados sean fiables y de alta calidad.

El objetivo final del preprocesado es crear tres archivos diferenciados: `variaciones.csv`, `variaciones_sin_etiqueta.csv` y `variaciones_con_etiqueta.csv`. Todos estos archivos tendrán los mismos atributos y la misma estructura, permitiendo una fácil integración y comparación.

- `variaciones.csv`: contendrá todas las variaciones genéticas combinadas, tanto de pacientes con diagnóstico como sin diagnóstico. Este archivo será utilizado para la codificación de variables categóricas mediante técnicas como la codificación *one-hot*.
- `variaciones_con_etiqueta.csv`: incluirá únicamente las variaciones de los pacientes con diagnóstico genético, donde cada variación está etiquetada como patogénica o no patogénica. Este archivo constituirá el grupo de entrenamiento para el modelo de *Machine Learning*.
- `variaciones_sin_etiqueta.csv`: comprenderá las variaciones de los pacientes sin diagnóstico genético. Estas variaciones no estarán etiquetadas y se utilizarán como el grupo de predicción, donde el modelo entrenado asignará las etiquetas.

Esta estructuración permite una clara distinción entre los datos utilizados para entrenar el modelo y aquellos que serán predichos, garantizando así un proceso ordenado y sistemático. La cuidadosa validación y preprocesamiento de los datos en cada etapa son fundamentales para el desarrollo de un modelo de *Machine Learning* eficaz y preciso, capaz de identificar la patogenicidad de nuevas variaciones genéticas de manera fiable.

Nota: Los pasos del preprocesado de datos descritos en esta sección se basan en las metodologías propuestas por [65], adaptadas para las necesidades específicas del presente estudio. Esto incluye técnicas y prácticas recomendadas para la preparación de datos genómicos y su integración en modelos predictivos.

7.1 Extracción y limpieza de los datos de entrada

Inicialmente, se procesa el archivo `XLSX` convirtiéndolo a formato `CSV` utilizando la librería `pandas`. Durante este proceso, se unifican las filas que corresponden a los mismos pacientes. Tras la conversión, se añade una columna denominada `Diagnóstico Genético`. Esta columna tomará el valor `True` si los campos `Mutaciones`, `Gen` y `Genotipo` están completos, indicando que el paciente tiene un diagnóstico genético, o `False` si estos campos

no están rellenados. A continuación, se añade otra columna llamada Anotaciones TXT, que indicará con un valor True si el paciente tiene un archivo TXT asociado, y False si no lo tiene. La columna Mutaciones se divide en dos nuevas columnas: cDNA y protein, según si la mutación afecta al ADN codificante o a una proteína. Finalmente, se realiza una limpieza manual de los datos para corregir posibles errores ortográficos en los nombres de los pacientes. Además, se utiliza el método replace para homogeneizar los valores nulos, reemplazando todos los tipos de valores nulos (campos con espacios en blanco, vacíos, guiones, etc.) por un campo vacío (’’).

A continuación, se procesan los 55 archivos CSV utilizando una combinación de los paquetes scikit-allel y pandas. Se extraerá información específica para las columnas CHROM, POS, ID, REF, ALT, QUAL y FILTER. El campo FILTER se convertirá automáticamente en un valor booleano, donde True indicará que la variación ha pasado todos los filtros y False que no lo ha hecho. Este campo se renombrará a FILTER_PASS. Además, se reemplazarán los puntos en los datos por campos vacíos (’’) para mantener la consistencia. Los 9 pacientes cuyos campos INFO están completos serán excluidos del análisis posterior, ya que no tienen un archivo TXT asociado, lo que significa que les falta información esencial. Esto deja un total de 46 archivos VCF, de los cuales 8 corresponden a pacientes con diagnóstico genético y 38 a pacientes sin diagnóstico genético. Finalmente, todos los archivos se renombrarán siguiendo el formato ID-full_variant_table.csv, donde ID es el identificador único de cada paciente.

Por último, los 46 archivos tipo TXT se convierten a CSV y se renombran con el mismo formato que los CSV pero añadiendo -txt al final: ID-full_variant_table-txt.csv.

7.2 Fusión de archivos VCF y TXT

Tras la limpieza inicial, se procedió a la fusión de los datos provenientes de los archivos VCF y TXT. Esta fusión es esencial para consolidar toda la información disponible sobre cada variación genética, permitiendo un análisis más completo al combinar los atributos disponibles en los TXT con los de los VCF.

7.2.1. Estudio de los atributos

Para fusionar los archivos, es esencial identificar los atributos comunes. Estos atributos son: sgid-ID, chromosome-CHROM, genome_position-POS, ref-REF, alt-ALT, y filter-FILTER_PASS. Se debe evaluar tanto la completitud como la coincidencia de estos atributos. Esto implica calcular el porcentaje de atributos completos y el porcentaje de coincidencia entre cada par de atributos en los archivos.

La fórmula para calcular la completitud es la siguiente:

$$\text{Completitud} = 1 - \left(\frac{\text{número de elementos no disponibles}}{\text{número de elementos totales}} \right)$$

Considerando como elementos no disponibles los campos en blanco, los resultados para los atributos seleccionados son los siguientes:

ID	CHROM	POS	REF	ALT	FILTER_PASS
0.978	1	1	1	1	1

Tabla 7.1: Completitud de los atributos en los archivos VCF para los 46 pacientes.

sgid	chromosome	genome_position	ref	alt	filter
0.978	1	1	1	1	1

Tabla 7.2: Completitud de los atributos en los archivos TXT para los 46 pacientes.

Como se observa en las Tablas 7.1 y 7.2, los pares de atributos tienen el mismo porcentaje de completitud. Todos los atributos están completos al 100 % excepto el par ID-sgid, que está al 97.8 %. Esto indica que todos son adecuados para su uso posterior.

Para evaluar la igualdad, se analiza si cada par de atributos tiene los mismos valores el mismo número de veces. Se indicará True si es así y False en caso contrario.

ID - sgid	CHROM - chromosome	POS - genome_position	REF - ref	ALT - alt	FILTER_PASS - filter
True	True	True	False	False	True

Tabla 7.3: Igualdad entre los atributos de los archivos VCF y TXT para los 46 pacientes.

El número de veces que cada valor aparece en los pares de atributos coincide en todos menos en los alelos de referencia y alternativos. Con base en esto, se dará prioridad a los atributos relacionados con la identificación de la variación, su posición, el cromosoma en el que se encuentra y si pasa los filtros establecidos.

7.2.2. Método de fusión

Primero, es necesario ordenar cada par de archivos correspondientes al mismo paciente, utilizando las columnas comunes en ambos archivos como clave de ordenación. Una vez ordenados, se emplea la función merge() de la librería Python pandas para unir las filas que tienen valores idénticos en todos los campos de los atributos comunes, tales como sgid-ID, chromosome-CHROM, genome_position-POS, y filter-FILTER_PASS. Los archivos resultantes de este proceso se denominarán siguiendo el esquema: ID-full-variant-table-fusion.csv.

7.3 Agrupación de variaciones y eliminación de duplicados

Tras la fusión de datos, se procedió a la agrupación de variaciones genéticas y la eliminación de duplicados, con el fin de evitar redundancias que pudieran afectar el análisis.

7.3.1. Identificación y eliminación de duplicados

En la gestión de grandes volúmenes de datos genéticos, como las aproximadamente 6 millones de entradas procesadas en este estudio, es esencial identificar y eliminar duplicados para garantizar la calidad y consistencia de los datos. A continuación, se detalla el proceso de detección y resolución de duplicados, asegurando que cada entrada en el conjunto de datos final sea única y precisa.

Procesamiento y manejo de datos

Primero, se determinaron los atributos necesarios para identificar dos variaciones como idénticas. Se seleccionaron tres de los cuatro atributos utilizados en la fusión inicial: el identificador único de la variación, el cromosoma afectado y la posición específica en la que ocurre. No se incluyó el atributo FILTER_PASS debido a que, por la variabilidad en la calidad de la secuenciación, una misma variación puede tener diferentes valores en

pacientes distintos. Se incorporaron los alelos de referencia y alternativos del `VCF`, junto con otros atributos clave de las anotaciones, como el gen afectado, el tipo de variación, la consecuencia codificante y la nomenclatura respecto a la secuencia de ADN codificante y la proteína. En resumen, los atributos considerados fueron: `CHROM`, `POS`, `ID`, `REF`, `ALT`, `gene`, `type`, `codingConsequence`, `cDNA`, y `protein`.

Durante el procesamiento inicial, se leyeron los datos en fragmentos (*chunks*) de 10,000 filas, almacenándolos en un diccionario global (`dic_global`) con claves compuestas por estos atributos. Al agregar una clave al diccionario, se garantizó que no volviera a aparecer, eliminando así los duplicados. Si se intentaba añadir una clave ya existente, los datos correspondientes se agregaban como valor a la clave ya almacenada. Tras identificar los duplicados con este método, se consolidaron las entradas redundantes para asegurar la integridad de los datos, generando un archivo consolidado denominado `diccionario.csv`.

Este archivo, `diccionario.csv`, contiene un total de 616,896 variantes únicas. Inicialmente, se partió de un total de 5,671,777 variantes provenientes de 46 pacientes, lo que implica una reducción del 89.124 % del número inicial de variantes, las cuales, según los criterios aplicados, estaban duplicadas.

Resolución de duplicados

El siguiente paso fue depurar los datos eliminando efectivamente los duplicados identificados. Se cargó el archivo `diccionario.csv` y se inició el proceso de selección de una representación única de variante para cada grupo de duplicados, aplicando las siguientes reglas:

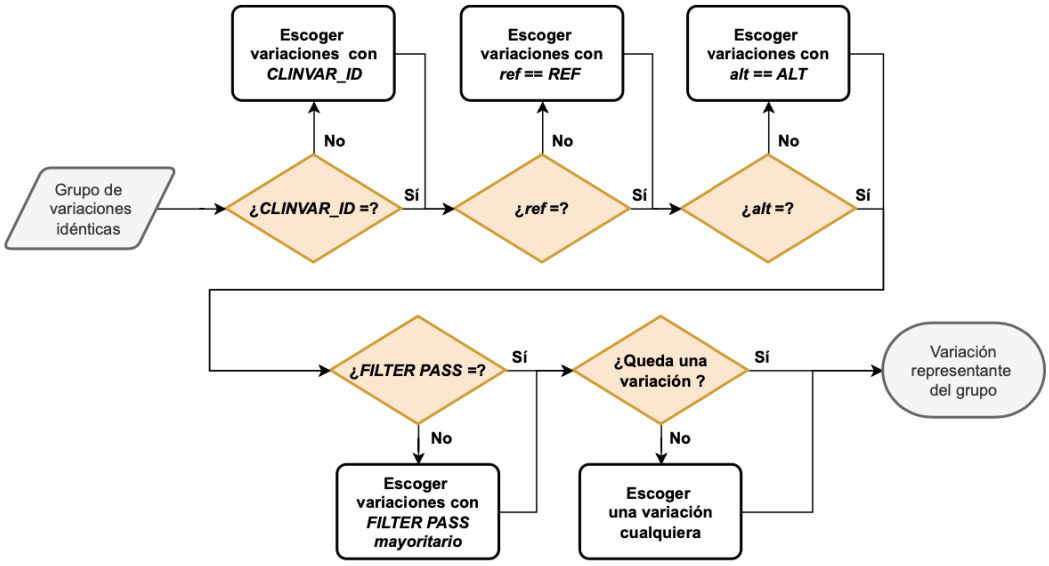

Figura 7.1: Diagrama de flujo para la elección de una variación representante de un grupo de variaciones idénticas [65].

- **Resolución por** `id_clinvar`: en casos donde un conjunto de duplicados contenía diferentes valores de `id_clinvar`, se conservaron solo las entradas con un valor no nulo para este atributo, reduciendo la ambigüedad en la identificación de variantes.
- **Validación de alelos de referencia y alternativos**: se verificaron los atributos `REF` y `ALT` para asegurar que los valores coincidieran correctamente entre los duplicados.

Se descartaron las entradas con inconsistencias en estos atributos para mejorar la calidad del conjunto de datos.

- **Filtro de calidad**: se evaluó el campo `FILTER_PASS` y se seleccionaron las entradas con el valor mayoritario. En situaciones de empate (igual cantidad de entradas con valores `True` y `False`), se seleccionó un valor al azar para resolver la duplicidad, aplicando un criterio aleatorio.
- **Selección aleatoria**: en casos donde las duplicidades no se pudieron resolver con los criterios anteriores, se realizó una selección aleatoria de una única entrada, asegurando que cada clave compuesta tuviera una sola representación en el conjunto final.

Finalmente, las entradas procesadas se almacenaron en un nuevo archivo denominado `variaciones_sin_duplicidad.csv`, que contiene un conjunto de datos depurado y listo para los análisis posteriores.

Concepto	**Cantidad**
Número total de variaciones iniciales	5,671,777
Número total de variaciones únicas	616,896
Número total de variaciones duplicadas	5,054,881
Número de variaciones únincas que no tienen duplicados	254,518
Número de variaciones únicas que tienen duplicados	362,378
Porcentaje de reducción	**89.12 %**

Tabla 7.4: Datos del procesamiento de variaciones antes y después de eliminar duplicidades.

A partir de los datos resumidos en la 7.4, se observa una significativa reducción en el número de variaciones, pasando de un total inicial de 5,671,777 a 616,896 variaciones únicas. Esto representa una reducción del 89.12 % en el número de entradas, lo cual subraya la alta incidencia de duplicados en el conjunto de datos original. La eliminación de estos duplicados es crucial, ya que garantiza la precisión y la fiabilidad de los análisis subsecuentes. Al consolidar las variaciones únicas y depurar las redundancias, se mejora la calidad del dataset, evitando sesgos y errores que podrían influir negativamente en los resultados de futuros estudios genéticos.

7.4 Selección y transformación de los datos de entrada

La selección y transformación de los datos son etapas fundamentales en el desarrollo de modelos de *Machine Learning*, ya que determinan la calidad y efectividad de los análisis y predicciones [48]. En el contexto de los datos genéticos, estas etapas son aún más críticas debido a la complejidad y el volumen de la información manejada [26].

En la fase de selección, se identifican y eligen los atributos más relevantes para el entrenamiento del modelo. La selección de características se basa en evaluar la completitud y la calidad de los atributos disponibles. Los atributos con alta completitud y relevancia se eligen para su inclusión en el análisis [5].

La transformación de datos, por otro lado, se encarga de preparar los atributos seleccionados para el modelado. Esto incluye la conversión de datos a formatos apropiados y la normalización de atributos para facilitar su procesamiento [48].

Esta preparación meticulosa asegura que los datos estén listos para el entrenamiento del modelo, maximizando su precisión y eficacia en la predicción de variaciones genéticas [26].

7.4.1. Selección de características

La selección de atributos para el entrenamiento y la predicción es esencial en el análisis genético. Esta selección se basa en la evaluación de la completitud de los atributos, lo que proporciona información valiosa sobre la calidad y disponibilidad de los datos.

Atributo	Completitud
CHROM	1
POS	1
ID	0.8897
REF	1
ALT	1
gene	0.9398
type	1
codingConsequence	1
gene_boundaries	0.9372
gene_strand	0.9372
c.DNA	82.03

Tabla 7.5: Completitud de algunos atributos del archivo `variaciones_sin_duplicidad.csv` para la selección de características.

Con base en la completitud de los datos, se han seleccionado los siguientes atributos por su relevancia y disponibilidad: `CHROM`, `POS`, `REF`, `ALT`, `type`, `codingConsequence`, `gene`, `gene_boundaries`, y `gene_strand`. En la tabla 7.5 se presenta la descripción de cada uno de estos atributos seleccionados:

Atributo	Descripción	Tipo de Dato	Ejemplo
CHROM	Cromosoma en el que se encuentra la variación genética.	Cadena de caracteres (*String*)	2
POS	Posición exacta de la variación en el cromosoma.	Entero	4373890
REF	Alelo de referencia en la variación genética.	Cadena de caracteres (*String*)	C
ALT	Alelo alternativo en la variación genética.	Cadena de caracteres (*String*)	G
type	Tipo de variación genética, como sustitución, inserción, o deleción.	Cadena de caracteres (*String*)	SNP
codingConsequence	Consecuencia codificante de la variación en el ADN.	Cadena de caracteres (*String*)	missense
gene	Gen afectado por la variación genética.	Cadena de caracteres (*String*)	IFT140
gene_boundaries	Pertenencia a los límites conocidos del gen afectado por la variación.	Cadena de caracteres (*String*)	within
gene_strand	Hebra del gen en la que ocurre la variación.	Cadena de caracteres (*String*)	+

Tabla 7.6: Descripción, tipo de dato y ejemplo de los atributos seleccionados.

La elección de estos atributos garantiza que se utilicen datos completos y relevantes para el análisis, mejorando la precisión y eficacia de los modelos predictivos en el estudio genético.

7.4.2. Transformación de los atributos

Como se muestra en la tabla 7.6, el único atributo que no es una cadena de caracteres es `POS`. Este atributo representa la posición exacta de la variación en el cromosoma y es de tipo entero. Dado el amplio rango de valores únicos en `POS`, a diferencia de `CHROM` que solo tiene 21 valores únicos, es necesario transformar este atributo para facilitar su codificación y análisis.

Para abordar esto, se aplicará la regla de Sturges para convertir el extenso rango de valores del atributo `POS` en intervalos. La regla de Sturges proporciona una fórmula para determinar el número óptimo de intervalos en un histograma, simplificando la visualización y el análisis de datos continuos.

La fórmula de la regla de Sturges para calcular el número de intervalos K es:

$$K = 1 + 3{,}3 \log_{10}(n)$$

donde n es el número total de observaciones del atributo `POS` [62]. Esta fórmula sugiere que el número de intervalos aumenta con el tamaño de la muestra, logrando un equilibrio entre la precisión y la generalización de los datos.

Una vez determinado el número de intervalos K, el ancho de cada intervalo (`Ac`) se calcula con la fórmula:

$$Ac = \frac{R}{K}$$

donde R es el rango total de los valores del atributo `POS` (es decir, la diferencia entre el valor máximo y el valor mínimo). La aplicación de estas fórmulas permitirá transformar el atributo `POS` en intervalos, simplificando su análisis y codificación, y facilitando el proceso de modelado y predicción.

Así, la variable `POS` será reemplazada por `POS_INTERVAL`, que representará el número del intervalo asignado a la posición original de la variación. La figura 7.2 ilustra la frecuencia de esta nueva variable en el conjunto de estudio:

Una vez sustituido el atributo `POS` por `POS_INTERVAL`, los atributos finales que se codificarán son: `POS_INTERVAL`, `CHROM`, `ALT`, `REF`, `gene`, `gene_boundaries`, `gene_strand`, `type`, y `codingConsequence`. Todos estos atributos están representados como cadenas de caracteres (*String*) y tienen un número razonable de valores únicos, lo que facilita la codificación. Estos atributos transformados se almacenarán en un nuevo archivo denominado `variaciones.csv`.

7.4.3. Generación de archivos finales

A partir del archivo `variaciones_sin_duplicidad.csv`, se seleccionan las variaciones correspondientes a los 38 pacientes sin diagnóstico genético, utilizando la columna `Pacientes` que se añadió previamente al descodificar `diccionario.csv`. A estas variaciones se les aplican las mismas transformaciones que a `variaciones.csv`, generando un nuevo archivo denominado `variaciones_sin_etiqueta.csv`. Este archivo contiene

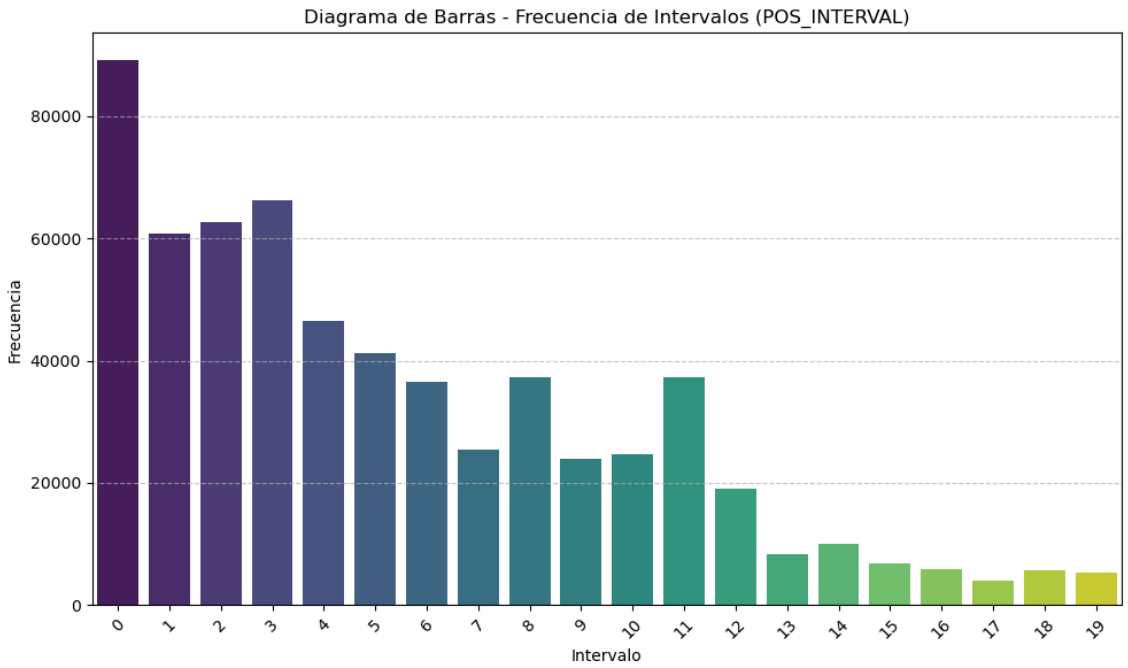

Figura 7.2: Diagrama de barras que muestra la frecuencia de la nueva variable `POS_INTERVAL`.

todas las variaciones preparadas para ser modeladas correspondientes a pacientes sin diagnóstico genético, quienes constitirán el conjunto de predicción.

Por otro lado, para las variaciones de los pacientes con diagnóstico genético extraídas de `variaciones_sin_duplicidad.csv`, es necesario realizar un paso adicional: etiquetarlas. En `resultados_exomas.xlsx` se encuentra la información relevante sobre las mutaciones responsables de las enfermedades en los 8 pacientes diagnosticados genéticamente. Se buscan iterativamente las variaciones asociadas a estas mutaciones en el archivo y se etiquetan con un valor de 1 en una nueva columna llamada "etiqueta", mientras que las demás se etiquetan con 0. Después, se aplican las mismas modificaciones que a los archivos previamente mencionados. Este conjunto etiquetado servirá como el conjunto de entrenamiento para el modelo.

Finalmente, los datos procesados se organizan en tres archivos clave: `variaciones.csv`, `variaciones_con_etiqueta.csv` y `variaciones_sin_etiqueta.csv`.

Concepto	**Cantidad**
Número total de variaciones únicas en `variaciones.csv`	616,896
Número total de variaciones únicas en `variaciones_con_etiqueta.csv`	307,273
Número total de variaciones únicas en `variaciones_sin_etiqueta.csv`	309,623

Tabla 7.7: Cantidad de variaciones tras la eliminación de duplicados en los diferentes archivos finales.

CAPÍTULO 8
Entrenamiento del modelo

8.1 Codificación y normalización de datos

En esta sección, se describe el proceso de codificación y normalización de los datos genéticos, específicamente aplicado a las columnas de variaciones genéticas presentes en el archivo `variaciones.csv`. Se emplearon varias técnicas para transformar estos datos en un formato adecuado para su análisis y modelado.

Es importante destacar que el archivo `variaciones.csv`, que contiene todas las variaciones únicas de todos los pacientes, se utiliza para ajustar el codificador. Esto asegura que todos los posibles valores de cada atributo estén representados en el codificador, evitando errores que podrían surgir si se encontraran nuevos valores no vistos durante el entrenamiento. De este modo, se transforma el conjunto de variaciones etiquetadas para la fase de entrenamiento y, posteriormente, se utiliza el mismo codificador para transformar el conjunto por etiquetar durante la fase de predicción, garantizando una codificación consistente y precisa.

8.1.1. Codificación de la columna `CHROM`

La columna `CHROM`, que representa los cromosomas, fue inicialmente mapeada a valores numéricos para facilitar su procesamiento. Se asignaron valores numéricos de 1 a 22 a los cromosomas autosómicos, mientras que los cromosomas sexuales fueron codificados con valores diferenciados: `X` como 50 y `Y` como 100.

8.1.2. Normalización de la columna `CHROM`

Posteriormente, estos valores fueron normalizados utilizando la técnica de `MinMaxScaler`, la cual escala los valores de una columna para que se distribuyan dentro del rango de 0 a 1. La fórmula utilizada para esta normalización es:

$$X_{\text{esc}} = \frac{X - X_{\text{mín}}}{X_{\text{máx}} - X_{\text{mín}}}$$

donde X es el valor original, $X_{\text{mín}}$ y $X_{\text{máx}}$ son los valores mínimo y máximo de la columna, respectivamente, y X_{esc} es el valor escalado. Esto asegura que las diferencias en la magnitud de los valores de los cromosomas no afecten desproporcionadamente a los análisis posteriores. La columna normalizada se almacenó bajo el nombre `CHROM_normalized`.

8.1.3. Normalización de la columna `Intervalo_POS`

De manera similar, la columna `Intervalo_POS`, que indica la posición genómica, fue normalizada utilizando `MinMaxScaler`. Esto resultó en la creación de una nueva columna normalizada denominada `Intervalo_POS_normalized`, que también se encuentra en el rango de 0 a 1, asegurando la consistencia en la escala de los valores de posición genómica.

8.1.4. Codificación de columnas categóricas

Se realizó la codificación de las columnas categóricas `REF`, `ALT`, `gene`, y `codingConsequence` mediante `LabelEncoder`. Cada una de estas columnas fue transformada en una representación numérica única, almacenando los codificadores en un diccionario para su reutilización en futuros procesos. Las columnas codificadas se guardaron con el sufijo `_encoded`.

8.1.5. Codificación *One-Hot*

Para las columnas categóricas `type`, `gene_strand`, y `gene_boundaries`, se utilizó la técnica de codificación `OneHotEncoder`. Esta técnica transforma cada categoría en una columna binaria, representando la presencia o ausencia de la categoría correspondiente. Las nuevas columnas codificadas se concatenaron al conjunto de datos, mientras que las originales fueron eliminadas.

8.1.6. Almacenamiento de datos y modelos

Finalmente, el conjunto de datos preprocesado, que incluye las columnas normalizadas y codificadas, se guardó en un archivo `CSV` denominado `variaciones_preprocesadas.csv`. Además, los modelos de codificación y escalado, como los `LabelEncoders`, `OneHotEncoder`, y `MinMaxScalers`, se almacenaron en archivos separados utilizando la librería `pickle`. Estos archivos permiten la reutilización de los modelos en futuros procesos de análisis de datos.

Con los codificadores previamente almacenados, se realizó la codificación de los archivos `variaciones_con_etiqueta.csv` y `variaciones_sin_etiqueta.csv`, que luego se guardaron en nuevos archivos con el sufijo _preprocesado añadido a sus nombres originales.

8.2 Entrenamiento

El modelo se entrenó utilizando un conjunto de datos genéticos previamente codificado y normalizado, extraído del archivo `variaciones_con_etiqueta.csv`, que contiene las variaciones genéticas **etiquetadas** de un grupo de pacientes. Uno de los principales desafíos de este conjunto de datos es el marcado desequilibrio de clases: la etiqueta `1` está significativamente subrepresentada en comparación con la etiqueta `0`, con una proporción aproximada de 8:307273 [4]. Esta situación refleja la limitada información disponible, ya que se trata de una enfermedad rara, lo que constituye el principal reto de este trabajo. La arquitectura de la red, implementada en Python, se detalla en el Anexo A.

Para abordar este problema de desequilibrio de clases y mejorar la capacidad del modelo para generalizar en la clase minoritaria, se realizaron los siguientes pasos:

8.2.1. Sobremuestreo y submuestreo con `SMOTETomek`

El primer paso fue aplicar la técnica combinada de sobremuestreo y submuestreo utilizando el método `SMOTETomek` [28]. `SMOTETomek` es una técnica que combina el Sobremuestreo de Minoría Sintética (`SMOTE`) [12] y el submuestreo de la clase mayoritaria mediante *Tomek Links* [63]. `SMOTE` genera nuevas muestras sintéticas de la clase minoritaria, mientras que *Tomek Links* limpia el ruido en los datos eliminando ejemplos que están muy cerca de la frontera de decisión entre las clases. Esta combinación ayuda a balancear mejor el conjunto de datos y a mejorar la calidad de las muestras para el entrenamiento.

8.2.2. Aumento de datos con ruido *Gaussiano*

Además del balanceo de clases, se añadió ruido gaussiano a las muestras generadas por `SMOTETomek`. Este proceso de aumento de datos consiste en perturbar las muestras con ruido controlado (escalado por un factor de 0.05 en este caso), para incrementar la diversidad del conjunto de datos y mejorar la robustez del modelo frente a variaciones no vistas durante el entrenamiento [60]. El ruido se añadió de forma que los valores perturbados se mantuvieran dentro del rango original de los datos, asegurando así que las características de los datos no se distorsionaran significativamente.

Finalmente, se combinaron las muestras originales y las muestras perturbadas para formar un conjunto de datos final que fue utilizado para el entrenamiento del modelo.

8.2.3. División del conjunto de datos en entrenamiento y validación

El conjunto de datos final, que incluye tanto las muestras balanceadas como las aumentadas, fue dividido en subconjuntos de entrenamiento y validación, utilizando una proporción de 80:20. Esta división se realizó utilizando la función `train_test_split`, asegurando que el modelo pudiera ser evaluado en un subconjunto de datos que no ha sido visto durante el entrenamiento, permitiendo así medir la capacidad de generalización del modelo [53].

8.2.4. Arquitectura del modelo y entrenamiento

Se diseñó una red neuronal convolucional (`CNN`) como modelo de clasificación binaria. La arquitectura del modelo incluye capas convolucionales seguidas de capas de normalización por lotes y capas de *pooling*. Se emplearon técnicas de regularización como `Dropout` y la regularización $L2$ para prevenir el sobreajuste [61]. Además, se añadió una capa de `Global Average Pooling` antes de las capas densas finales.

El modelo fue compilado con el optimizador `Adam`, ajustando la tasa de aprendizaje a 0,0005 para facilitar una convergencia estable [39]. Se utilizó la función de pérdida `binary_crossentropy` y la métrica de precisión para evaluar el rendimiento del modelo durante el entrenamiento.

8.2.5. Manejo del desequilibrio de clases con pesos de clase

Dado el desequilibrioinherente en los datos, incluso después de aplicar `SMOTETomek`, se calcularon pesos de clase para compensar la desproporción de etiquetas [4]. Estos pesos fueron aplicados durante el entrenamiento para asegurar que el modelo prestara suficiente atención a la clase minoritaria.

8.2.6. Estrategias de regularización y *callbacks*

Para mejorar el rendimiento del modelo y evitar el sobreajuste, que había sido un problema recurrente en el resto de modelos probados previamente, se implementaron estrategias de regularización adicionales, como `Early Stopping` y el ajuste dinámico de la tasa de aprendizaje (`ReduceLROnPlateau`). El `Early Stopping` monitorizó la pérdida de validación, deteniendo el entrenamiento si la pérdida no mejoraba después de 5 épocas consecutivas, y restaurando los mejores pesos alcanzados. El `ReduceLROnPlateau` redujo la tasa de aprendizaje a la mitad si no se observaban mejoras en la pérdida de validación durante 3 épocas consecutivas, con un límite mínimo de $1e-6$ [24].

8.2.7. Resultados del entrenamiento y evaluación

El modelo fue entrenado durante un máximo de 50 épocas, aunque el `Early Stopping` evitó entrenamientos innecesarios al detenerse una vez que no se observaban mejoras en la validación. Tras completar el entrenamiento, el modelo fue evaluado en el conjunto de validación, alcanzando una precisión de validación que indica su capacidad para generalizar sobre datos no vistos.

El modelo entrenado se guardó como `modelo_entrenado_cnn_mejorado_v2.h5` para su uso posterior en la fase de predicción.

8.2.8. Visualización del proceso de entrenamiento

Se graficaron, usando la librería `matplotlib.pyplot`, tanto la pérdida como la precisión durante las épocas de entrenamiento para visualizar la evolución del modelo y detectar posibles signos de sobreajuste o problemas de convergencia [13].

En resumen, el proceso de entrenamiento fue cuidadosamente diseñado para abordar el desequilibrio de clases y mejorar la capacidad de generalización del modelo, empleando técnicas de aumento de datos, regularización y ajuste fino del aprendizaje. Estos esfuerzos resultaron en un modelo robusto y preparado para la fase de predicción.

8.3 Predicción con el modelo entrenado

Una vez completado el entrenamiento del modelo, el siguiente paso es utilizarlo para realizar predicciones sobre los datos no etiquetados. En este caso, se utiliza el archivo `variaciones_sin_etiqueta.csv`, que contiene las variaciones genéticas de 48 pacientes sin diagnóstico genético. El proceso de predicción se detalla a continuación.

8.3.1. Carga del modelo y datos

Para llevar a cabo las predicciones, primero se deben cargar los datos preprocesados que se desean etiquetar. Estos datos se encuentran en el archivo `CSV` denominado `variaciones_sin_etiqueta.csv`.

8.3.2. Realización de predicciones

Con el modelo y los datos cargados, se procede a realizar las predicciones. El modelo evalúa las variaciones genéticas y produce resultados que se convierten en etiquetas binarias (0 o 1) para la clasificación. La finalidad es identificar las variaciones que el mode-

lo considera potencialmente patogénicas, es decir, aquellas que podrían estar asociadas con la retinosis pigmentaria. Las predicciones se comparan con un umbral de 0.5 para determinar las etiquetas binarias: las predicciones superiores a 0.5 se etiquetan como 1, mientras que las inferiores a 0.5 se etiquetan como 0.

8.3.3. Almacenamiento de resultados

Finalmente, las etiquetas predichas se añaden al *DataFrame* original y se guardan en un nuevo archivo `CSV` denominado `etiquetas_predichas.csv`. Este archivo contiene los datos originales junto con las etiquetas predichas por el modelo, permitiendo su posterior análisis y evaluación.

CAPÍTULO 9
Evaluación del modelo

La evaluación de un modelo de aprendizaje automático es esencial para entender su rendimiento y su capacidad para generalizar a nuevos datos. A continuación, se describen las métricas utilizadas para evaluar el modelo entrenado en este trabajo.

9.1 Métricas utilizadas

Las métricas de evaluación proporcionan información crítica sobre cómo se está desempeñando un modelo y ayudan a identificar áreas de mejora. Las dos categorías principales de métricas discutidas en este capítulo son la pérdida y la precisión, tanto en el contexto de validación como de entrenamiento.

9.1.1. Pérdida y precisión de entrenamiento

Pérdida de entrenamiento: mide la calidad del modelo durante el proceso de entrenamiento en el conjunto de datos de entrenamiento. Al igual que la pérdida de validación, se calcula utilizando la función de pérdida aplicada a las predicciones del modelo sobre los datos de entrenamiento. La pérdida de entrenamiento se utiliza para ajustar los parámetros del modelo y minimizar el error. Un descenso continuo en la pérdida de entrenamiento a lo largo de las épocas generalmente indica que el modelo está aprendiendo y ajustando adecuadamente sus parámetros. Sin embargo, una pérdida de entrenamiento extremadamente baja combinada con una alta pérdida de validación puede ser indicativa de sobreajuste.

Precisión de entrenamiento: indica la proporción de predicciones correctas del modelo en el conjunto de datos de entrenamiento. Se calcula de manera similar a la precisión de validación, con el número de verdaderos positivos más verdaderos negativos dividido por el total de instancias en el conjunto de entrenamiento. La precisión de entrenamiento permite evaluar qué tan bien el modelo se está adaptando a los datos de entrenamiento. Un modelo con alta precisión de entrenamiento y baja pérdida de entrenamiento sugiere un buen ajuste a los datos de entrenamiento, aunque es importante asegurarse de que esto no implique sobreajuste [41].

9.1.2. Pérdida y precisión de validación

Pérdida de validación: es una métrica que cuantifica la calidad del modelo en el conjunto de datos de validación durante el proceso de entrenamiento. Se calcula como el valor de la función de pérdida aplicada a las predicciones del modelo en el conjunto de

validación. La función de pérdida mide la discrepancia entre las predicciones del modelo y las etiquetas verdaderas. Un valor bajo de pérdida de validación indica que el modelo está haciendo buenas predicciones, mientras que un valor alto sugiere que el modelo podría estar cometiendo errores significativos [24]. Monitorear esta métrica es crucial para detectar si el modelo está sobreajustado (*overfitting*) o subajustado (*underfitting*).

Precisión de validación: mide la proporción de predicciones correctas realizadas por el modelo en el conjunto de datos de validación. Se calcula como el número de verdaderos positivos más verdaderos negativos dividido por el total de instancias en el conjunto de validación. Una alta precisión de validación indica que el modelo es capaz de generalizar bien y hacer predicciones precisas en datos no vistos durante el entrenamiento [7].

9.2 Resultados obtenidos

9.2.1. Pérdida y precisión del modelo

Tras notables intentos, experimentación y ajustes, se logró desarrollar un modelo de red neuronal convolucional que presenta un rendimiento notable en términos de precisión y pérdida, superando todas las configuraciones anteriores. La Figura 9.1 muestra las curvas de pérdida y precisión tanto para el conjunto de entrenamiento como para el conjunto de validación a lo largo de las épocas de entrenamiento. Estos resultados reflejan el éxito de las decisiones tomadas respecto a la arquitectura del modelo, la selección de hiperparámetros y la estrategia de preprocesamiento de datos.

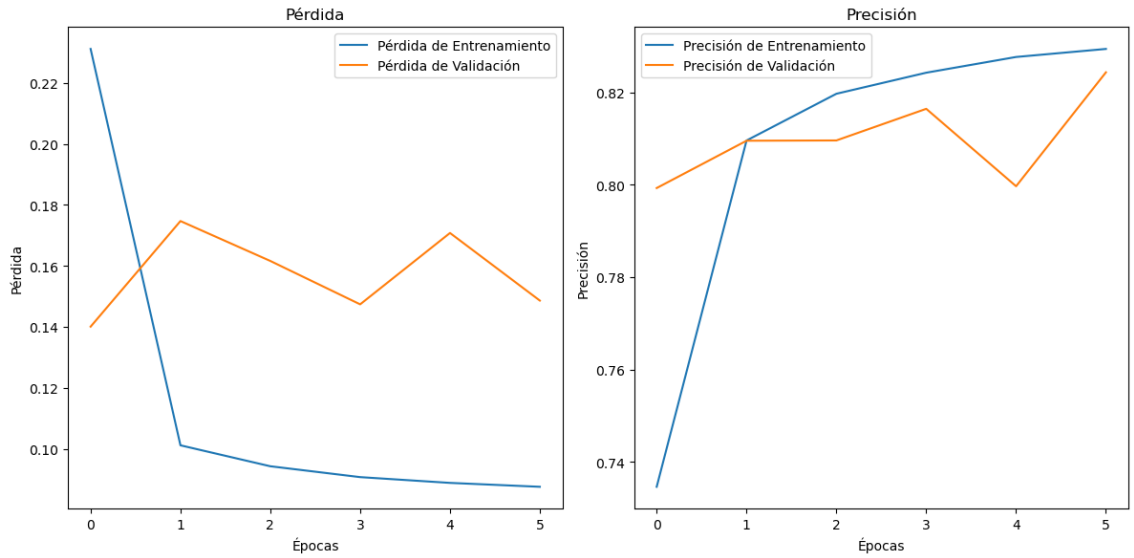

Figura 9.1: Curvas de pérdida y precisión durante el entrenamiento y la validación.

Pérdida

La gráfica de pérdida, como se muestra en la Figura 9.1, evidencia una disminución continua y significativa de la pérdida en el conjunto de entrenamiento, partiendo de un valor inicial de 0.2311 en la primera época hasta alcanzar 0.0875 en la sexta época. Esta tendencia a la baja es indicativa de que el modelo está optimizando sus parámetros de manera efectiva para ajustarse a los datos de entrenamiento.

En el conjunto de validación, la pérdida muestra una mayor variabilidad, comenzando en 0.1400 y fluctuando en un rango estrecho entre 0.1400 y 0.1747 a lo largo de las

primeras seis épocas. Sin embargo, es crucial destacar que estas fluctuaciones no indican sobreajuste, sino que reflejan una capacidad robusta del modelo para generalizar. Las estrategias aplicadas para combatir el sobreajuste, como la regularización L2, el uso de *Dropout* y el ajuste de la tasa de aprendizaje, han sido fundamentales para mantener estas pérdidas controladas, sin permitir que aumenten significativamente, como sucedió en intentos anteriores.

Precisión

En cuanto a la precisión, los resultados son notables. La precisión del conjunto de entrenamiento muestra un incremento constante, pasando de 0.7346 en la primera época a 0.8295 en la sexta. Este aumento progresivo indica que el modelo mejora consistentemente su capacidad para aprender las características relevantes de los datos de entrenamiento.

La precisión en el conjunto de validación también refleja una tendencia positiva, comenzando en 0.7993 y alcanzando un máximo de 0.8244. Aunque se observa una ligera disminución en la quinta época, esta es seguida por una recuperación en la sexta, lo cual es un indicio de la estabilidad del modelo. Cabe resaltar que este nivel de precisión es el más alto obtenido en todos los experimentos realizados, lo que subraya el éxito de las técnicas de preprocesamiento y ajuste de hiperparámetros implementadas.

9.2.2. Evaluación final del modelo

El modelo fue evaluado en el conjunto de validación, obteniendo una pérdida final de 0.1400 y una precisión de 0.7993. Estos resultados indican un avance significativo en comparación con las arquitecturas y configuraciones anteriores, que a menudo resultaban en sobreajuste o en un incremento de la pérdida en las últimas épocas. La capacidad del modelo para mantener una baja pérdida y una alta precisión en el conjunto de validación sugiere que es un modelo bien equilibrado y robusto.

Estos resultados confirman que las decisiones tomadas durante el desarrollo, desde la combinación de `SMOTETomek` y la adición de ruido hasta el uso de la función de pérdida *BinaryFocalCrossentropy*, han sido correctas. Este modelo, que supera a todas las configuraciones anteriores, representa un avance importante en el proyecto y establece una base sólida para futuras aplicaciones.

Gracias a este modelo, se han etiquetado 43,567 mutaciones del archivo `variaciones_sin_etiqueta.csv` con un 1, lo que indica que se consideran potencialmente patogénicas y son candidatas a un estudio posterior por personal médico para determinar si alguna o varias de estas mutaciones causan la enfermedad. La reducción de variaciones posiblemente patogénicas ha sido notable, pasando de 309,623 a solo 43,567, lo que representa una disminución del 85.92 % en el número de candidatas.

CAPÍTULO 10
Conclusiones

Este trabajo ha logrado desarrollar un modelo de red neuronal convolucional (CNN) para la identificación de mutaciones patogénicas en un entorno caracterizado por la escasez y el desequilibrio de datos, específicamente en el contexto de la retinosis pigmentaria, una enfermedad genética rara. A lo largo del proyecto, se enfrentaron desafíos significativos que incluían la limitada disponibilidad de datos y el desequilibrio entre las clases, problemas que son comunes en la investigación de enfermedades raras. La correcta gestión de estos desafíos resultó crucial para asegurar la eficacia del modelo en la identificación precisa de mutaciones que pueden tener un impacto clínico importante.

El éxito del modelo se debió en gran parte a las técnicas de preprocesamiento de datos aplicadas, las cuales fueron esenciales para abordar las limitaciones inherentes del conjunto de datos. En particular, la técnica de sobremuestreo combinada con la limpieza de ruido mediante `SMOTETomek` desempeñó un papel clave al equilibrar las clases de mutaciones patogénicas y no patogénicas, mejorando la representatividad del conjunto de datos y reduciendo el sesgo en el modelo. Además, la adición de ruido *gaussiano* a los datos contribuyó a la robustez del modelo, ayudando a prevenir el sobreajuste, un problema frecuente en escenarios con datos limitados. Estas técnicas, junto con la regularización aplicada en las capas densas del modelo y el uso de la función de pérdida `BinaryFocalCrossentropy`, permitieron que el modelo aprendiera patrones relevantes sin depender excesivamente de las peculiaridades de los datos de entrenamiento.

Los resultados obtenidos, con una precisión de validación que alcanzó el 79.93 %, demuestran que el enfoque adoptado no solo es viable, sino también eficaz en el manejo de conjuntos de datos pequeños y desbalanceados. Este avance es especialmente significativo en el campo de la genómica computacional, donde la capacidad para identificar mutaciones patogénicas en enfermedades raras es fundamental para el desarrollo de tratamientos personalizados y para el asesoramiento genético.

Las implicaciones de este trabajo van más allá de la retinosis pigmentaria. El enfoque metodológico aquí desarrollado puede servir de base para futuras investigaciones en otras enfermedades raras que enfrentan desafíos similares en cuanto a la disponibilidad y calidad de los datos. La capacidad para preprocesar y modelar datos genómicos de manera efectiva abre nuevas posibilidades en la investigación de la genómica computacional, facilitando el descubrimiento de mutaciones relevantes que podrían haber pasado desapercibidas con métodos más tradicionales. Asimismo, este avance subraya la importancia de continuar explorando y mejorando las técnicas de preprocesamiento de datos en la investigación genómica, un área que es crítica para el éxito de los modelos de aprendizaje automático en contextos clínicos.

En futuras investigaciones, sería valioso explorar la integración de fuentes de datos adicionales, como información fenotípica o datos relacionados con otras enfermedades

raras, con el fin de mejorar la capacidad de generalización del modelo. Asimismo, el desarrollo de nuevas técnicas de preprocesamiento adaptadas a las características específicas de los datos genómicos podría contribuir significativamente a incrementar la precisión y la aplicabilidad clínica de los modelos predictivos.

En adelante, también sería interesante investigar el uso de **redes neuronales recurrentes (RNN)**, que podrían resultar útiles para modelar la naturaleza secuencial y las dependencias temporales presentes en los datos genómicos. Dado que estas redes son capaces de capturar información de secuencias de datos, podrían mejorar la identificación de mutaciones patogénicas, especialmente cuando se toman en cuenta patrones temporales o estructurales más profundos.

Otra línea de investigación prometedora sería llevar a cabo experimentos seleccionando un **mayor número de atributos** para el entrenamiento del modelo. La expansión del conjunto de características permitiría al modelo aprender patrones más complejos, lo que podría aumentar su capacidad predictiva y, en consecuencia, mejorar la precisión en la identificación de mutaciones patogénicas.

Todo el código desarrollado en este proyecto está disponible en mi repositorio de `GitHub`, accesible en el siguiente enlace:

`https://github.com/yaaaiii/tfg-yai.git`.

En resumen, este trabajo ha contribuido no solo a la mejora en la identificación de mutaciones patogénicas en la retinosis pigmentaria, sino que también ha sentado un precedente importante para el desarrollo de herramientas de diagnóstico avanzadas en el ámbito de la genómica computacional. La capacidad de los modelos de aprendizaje automático para trabajar con datos limitados y desbalanceados, cuando se preprocesan adecuadamente, representa un avance significativo en la investigación de enfermedades raras, con el potencial de mejorar la calidad de vida de los pacientes y de impulsar nuevas estrategias terapéuticas personalizadas.

Bibliografía

[1] Administrador. *Diferencia entre neurona y neuroglía - Centros EQ & Psycolab, centro de Psicología, Neuropsicología, Logopedia, Pedagogía en Benalmádena y Málaga*. https://www.psycolab.com/diferencia-entre-neurona-y-neuroglia/. [Accessed 21-08-2024].

[2] Francisco Xavier Palacios Andrade, Lizette Espinosa Martín y Karla María Cumbe Guerrero. «Retinosis Pigmentaria Retinitis Pigmentosa». En: *Revista Estudiantil CEUS* 1.1 (2019).

[3] Christof Angermueller et al. «Deep learning for computational biology». En: *Molecular Systems Biology* 12.7 (2016), pág. 878.

[4] Gustavo E. A. P. A. Batista, Ronaldo C. Prati y Maria Carolina Monard. «A study of the behavior of several methods for balancing machine learning training data». En: *ACM SIGKDD Explorations Newsletter* 6.1 (2004), págs. 20-29.

[5] Yoav Benjamini y Daniel Yekutieli. «Controlling the false discovery rate: A practical and powerful approach to multiple testing». En: *Journal of the Royal Statistical Society: Series B (Methodological)* 57.1 (1995). Discusses methods for adjusting data analysis to control for multiple testing errors, relevant for genetic data analysis., págs. 289-300.

[6] Juan José Beunza y Enrique Puertas. «Tipos de algoritmos». En: *Manual práctico de inteligencia artificial en entornos sanitarios* (2020), pág. 35.

[7] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[8] Léon Bottou. «Large-scale machine learning with stochastic gradient descent». En: *Proceedings of COMPSTAT 2010*. Springer. 2010, págs. 177-186.

[9] Y-Lan Boureau et al. «A theoretical analysis of feature pooling in visual recognition». En: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, págs. 111-118.

[10] Sandra Brasil et al. «Artificial Intelligence (AI) in Rare Diseases: Is the Future Brighter?» En: *Genes* 10.12 (2019). ISSN: 2073-4425. DOI: 10.3390/genes10120978. URL: https://www.mdpi.com/2073-4425/10/12/978.

[11] N.V. Chawla et al. «SMOTE: Synthetic Minority Over-sampling Technique». En: *Journal of Artificial Intelligence Research* 16 (2003), págs. 321-357.

[12] Nitesh V Chawla et al. «SMOTE: Synthetic minority over-sampling technique». En: *Journal of Artificial Intelligence Research* 16 (2002), págs. 321-357.

[13] Francois Chollet. «Deep learning with Python». En: *Manning Publications Co.* 2018.

[14] Fanny Cortés M. «Las enfermedades raras». En: *Revista Médica Clínica Las Condes* 26.4 (2015), págs. 425-431.

[15] Petr Danecek et al. «The variant call format and VCFtools». En: *Bioinformatics* 27.15 (2011), págs. 2156-2158.

[16] Imbalanced-learn developers. *Imbalanced-learn Documentation*. 2021. URL: https://imbalanced-learn.org/.

[17] NumPy developers. *NumPy Documentation*. 2021. URL: https://numpy.org/.

[18] Scikit-learn developers. *compute_class_weight*. Scikit-learn Documentation. 2021. URL: https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html.

[19] Scikit-learn developers. *Scikit-learn Documentation*. 2021. URL: https://scikit-learn.org/.

[20] Scikit-learn developers. *train_test_split*. Scikit-learn Documentation. 2021. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[21] TensorFlow developers. *TensorFlow Documentation*. 2021. URL: https://www.tensorflow.org/.

[22] Free Software Foundation. *Text Files: TXT Format*. Description of the text file format. 2023. URL: https://www.gnu.org/software/libc/manual/html_node/Text-Files.html.

[23] *globalgenes.org*. https://globalgenes.org/wp-content/uploads/2013/04/ShireReport-1.pdf. [Accessed 02-08-2024].

[24] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep learning*. MIT Press, 2016.

[25] Ian Goodfellow et al. «Generative adversarial nets». En: *Advances in neural information processing systems*. Vol. 27. 2014.

[26] Trevor Hastie, Robert Tibshirani y Jerome Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2009.

[27] Simon Haykin. *Neural networks and learning machines*. Vol. 3. Pearson Education, 2009.

[28] Haibo He y Yang Bai. «ADASYN: Adaptive synthetic sampling approach for imbalanced learning». En: *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* (2008), págs. 1322-1328.

[29] Kaiming He et al. «Deep residual learning for image recognition». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, págs. 770-778.

[30] Sepp Hochreiter y Jürgen Schmidhuber. «Long short-term memory». En: *Neural computation* 9.8 (1997), págs. 1735-1780.

[31] Ignacio Moreno Hojas. *Regresión logística en Python - StatDeveloper*. https://www.statdeveloper.com/regresion-logistica-en-python/. [Accessed 21-08-2024].

[32] Andrew G. Howard, Menglong Zhu et al. «Mobilenets: Efficient convolutional neural networks for mobile vision applications». En: *arXiv preprint arXiv:1704.04861*. 2017.

[33] https://www.reproduccionasistida.org/retinosis-pigmentaria/que-es-la-retinosis-pigmentaria/. URL: https://www.reproduccionasistida.org/wp-content/uploads/2022/10/que-es-la-retinosis-pigmentaria.png.

[34] J.D. Hunter. «Matplotlib: A 2D graphics environment». En: *Computing in Science & Engineering* 9.3 (2007), págs. 90-95.

[35] Christian Janiesch, Patrick Zschech y Kai Heinrich. «Machine learning and deep learning». en. En: *Electron. Mark.* 31.3 (sep. de 2021), págs. 685-695.

[36] Ander Fernández Jauregui. *Cómo programar un árbol de decisión en Python - Ander Fernández*. https://anderfernandez.com/blog/programar-arbol-decision-python-desde-0/. [Accessed 20-08-2024].

[37] Taeho Jo. «Machine learning foundations». En: *Machine Learning Foundations. Springer Nature Switzerland AG* (2021). DOI: 10.1007/978-3-030-65900-4.

[38] James Jumper et al. «Highly accurate protein structure prediction with AlphaFold». En: *Nature* 596 (2021), págs. 583-589.

[39] Diederik P. Kingma y Jimmy Ba. «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980* (2014).

[40] Alex Krizhevsky et al. «Imagenet classification with deep convolutional neural networks». En: *Advances in Neural Information Processing Systems*. Vol. 25. 2012.

[41] Yann LeCun, Yoshua Bengio y Geoffrey Hinton. «Deep learning». En: *Nature* 521.7553 (2015), págs. 436-444.

[42] Yann LeCun et al. «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324.

[43] J. Lee, J. Kim y S. Kim. «Robustness of convolutional neural networks to input noise». En: *IEEE Transactions on Neural Networks and Learning Systems* 28.2 (2017), págs. 123-135.

[44] T.Y. Lin et al. «Focal loss for dense object detection». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2017), págs. 299-312.

[45] Jonathan Long et al. «Fully convolutional networks for semantic segmentation». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), págs. 3431-3440.

[46] Wes McKinney. *Python for Data Analysis*. Sebastopol, CA: O'Reilly Media, 2018.

[47] Microsoft. *XLSX File Format Specification*. Microsoft Excel file format specification. 2023. URL: https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-xlsx/.

[48] Michael K. Müller y Sergey Guo. *Introduction to Machine Learning*. Covers foundational concepts in machine learning, including data selection and transformation. Cambridge, MA: MIT Press, 2016.

[49] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. MIT Press, 2012.

[50] Samreen Naeem et al. «An unsupervised machine learning algorithms: Comprehensive review». En: *International Journal of Computing and Digital Systems* (2023).

[51] Vinod Nair y Geoffrey E Hinton. «Rectified linear units improve restricted Boltzmann machines». En: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, págs. 807-814.

[52] Vladimir Nasteski. «An overview of the supervised machine learning methods». En: *Horizons* 4.51-62 (2017), pág. 56.

[53] Fabian Pedregosa et al. «Scikit-learn: Machine learning in Python». En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[54] I. Perea-Romero et al. «Inherited Retinal Dystrophies in Spain: three decades of epidemiological, clinical, and genetic study». es. En: *An. R. Acad. Nac. Med. (Madr.)* 139.139(03) (2023), págs. 274-284.

[55] L. Prechelt. «Early stopping - But when?» En: *Neural Networks: Tricks of the Trade*. Berlin, Heidelberg: Springer, 1998.

[56] *Qué son las redes neuronales y sus aplicaciones | OpenWebinars*. https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/. [Accessed 21-08-2024].

[57] Joseph Redmon et al. «You only look once: Unified, real-time object detection». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, págs. 779-788.

[58] Ana Rodríguez-Muñoz et al. «Expanding the clinical and molecular heterogeneity of nonsyndromic inherited retinal dystrophies». En: *The Journal of Molecular Diagnostics* 22.4 (2020), págs. 532-543.

[59] Julia Schaefer et al. «The use of machine learning in rare diseases: a scoping review». en. En: *Orphanet J. Rare Dis.* 15.1 (jun. de 2020), pág. 145.

[60] Connor Shorten y Taghi M. Khoshgoftaar. «A survey on image data augmentation for deep learning». En: *Journal of Big Data* 6.1 (2019), págs. 1-48.

[61] Nitish Srivastava et al. «Dropout: A simple way to prevent neural networks from overfitting». En: *Journal of Machine Learning Research*. 2014, págs. 1929-1958.

[62] H. A. Sturges. «The choice of a class interval». En: *Journal of the American Statistical Association* 21.153 (1926), págs. 65-66.

[63] Ivan Tomek. «Two modifications of CNN». En: *IEEE Transactions on Systems, Man, and Cybernetics* 6 (1976), págs. 769-772.

[64] Bruna Andrea Vallejo et al. «Artificial intelligence for rare diseases: the importance of the context». En: *Genetic research* 102 (2019), e1.

[65] Andrea Vaño Ribelles. «Análisis, diseño e implementación de clasificadores genómicos para la detección de distrofias hereditarias de la retina mediante técnicas de machine learning». En: (2022).

[66] Ashish Vaswani et al. «Attention is all you need». En: *Advances in neural information processing systems*. Vol. 30. 2017.

[67] Dewi Widyawati y Amaliah Faradibah. «Comparison Analysis of Classification Model Performance in Lung Cancer Prediction Using Decision Tree, Naive Bayes, and Support Vector Machine». En: *Indonesian Journal of Data and Science* 4 (jul. de 2023), págs. 80-89. DOI: 10.56705/ijodas.v4i2.76.

[68] Tom Young et al. «Recent trends in deep learning based natural language processing». En: *IEEE Computational Intelligence Magazine* 13.3 (2018), págs. 55-75.

[69] Jinming Zou, Yi Han y Sung-Sau So. «Overview of artificial neural networks». En: *Artificial neural networks: methods and applications* (2009), págs. 14-22.

APÉNDICE A
Modelo utilizado

```
from imblearn.combine import SMOTETomek
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras.losses import BinaryFocalCrossentropy
from tensorflow.keras.regularizers import l2
from sklearn.model_selection import StratifiedKFold

smote_tomek = SMOTETomek(random_state=42)
X_res, y_res = smote_tomek.fit_resample(X, y)

def add_noise(X, noise_factor=0.05):
    noise = np.random.randn(*X.shape) * noise_factor
    X_noisy = X + noise
    X_noisy = np.clip(X_noisy, 0., 1.)
    return X_noisy

X_res_noisy = add_noise(X_res, noise_factor=0.05)

X_final = np.concatenate([X_res, X_res_noisy])
y_final = np.concatenate([y_res, y_res])

X_train, X_val, y_train, y_val = train_test_split(X_final, y_final,
    test_size=0.2, random_state=42)

X_train = np.expand_dims(X_train, axis=2)
X_val = np.expand_dims(X_val, axis=2)

class_weights = compute_class_weight(class_weight='balanced', classes=np.
    unique(y_res), y=y_res)
class_weights_dict = dict(enumerate(class_weights))

model = Sequential()

model.add(Conv1D(64, kernel_size=2, activation='relu', input_shape=(X_train
    .shape[1], 1)))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

model.add(Conv1D(128, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

model.add(Conv1D(256, kernel_size=2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Flatten())
```

```
model.add(Dense(256, activation='relu', kernel_regularizer=l2(0.01)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=Adam(learning_rate=0.0005),
              loss=BinaryFocalCrossentropy(gamma=2.0),
              metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=5,
    restore_best_weights=True)

history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=64,
                    validation_data=(X_val, y_val),
                    class_weight=class_weights_dict,
                    callbacks=[early_stopping])

val_loss, val_accuracy = model.evaluate(X_val, y_val)
print(f'Validation loss: {val_loss}')
print(f'Validation accuracy: {val_accuracy}')

model.save('/home/ec2-user/SageMaker/segundo intento/final2/preprocesado/9/
    modelo_entrenado_cnn.h5')

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Perdida de Entrenamiento')
plt.plot(history.history['val_loss'], label='Perdida de Validacion')
plt.title('Perdida')
plt.xlabel('Epocas')
plt.ylabel('Perdida')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Precision de Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Precision de Validacion')
plt.title('Precision')
plt.xlabel('Epocas')
plt.ylabel('Precision')
plt.legend()

plt.tight_layout()
plt.show()
```

Listing A.1: Código para el preprocesamiento y entrenamiento del modelo

APÉNDICE A

Relación con los Objetivos de Desarrollo Sostenible (ODS)

El 25 de septiembre de 2015, los líderes mundiales adoptan un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos, como parte de la Agenda 2030 para el Desarrollo Sostenible. Cada objetivo tiene metas específicas para los próximos 15 años. El estudio sobre la retinosis pigmentaria y el uso de técnicas avanzadas de aprendizaje automático se relaciona con varios de estos objetivos, destacando especialmente:

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza**				X
ODS 2. **Hambre cero**				X
ODS 3. **Salud y bienestar**	X			
ODS 4. **Educación de calidad**	X			
ODS 5. **Igualdad de género**				X
ODS 6. **Agua limpia y saneamiento**				X
ODS 7. **Energía asequible y no contaminante**				X
ODS 8. **Trabajo decente y crecimiento económico**				X
ODS 9. **Industria, innovación e infraestructuras**	X			
ODS 10. **Reducción de las desigualdades**	X			
ODS 11. **Ciudades y comunidades sostenibles**				X
ODS 12. **Producción y consumo responsables**				X
ODS 13. **Acción por el clima**				X
ODS 14. **Vida submarina**				X
ODS 15. **Vida de ecosistemas terrestres**				X
ODS 16. **Paz, justicia e instituciones sólidas**				X
ODS 17. **Alianzas para lograr objetivos**				X

Tabla A.1: Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

A.0.1. Salud y bienestar (ODS 3)

El estudio contribuye de manera significativa a la salud y el bienestar (ODS 3) mediante la mejora de la identificación y predicción de mutaciones patogénicas en la retinosis pigmentaria, una enfermedad genética rara que causa pérdida progresiva de visión. . La capacidad del modelo para superar el sobreajuste y ofrecer una alta precisión en datos

no vistos resalta el impacto positivo en la investigación de enfermedades raras y en la calidad de vida de los pacientes.

A.0.2. Educación de calidad (ODS 4)

El desarrollo de técnicas avanzadas en el estudio fomenta la educación de calidad (ODS 4) al proporcionar herramientas innovadoras en el campo de la biología computacional y el aprendizaje automático. La aplicación de estas técnicas en la investigación genética ofrece una formación avanzada para profesionales en la materia, promoviendo el aprendizaje de métodos que optimizan el análisis de datos desbalanceados y limitados. Este avance educativo se traduce en una mayor capacidad para abordar problemas complejos en genética y biología computacional.

A.0.3. Reducción de las desigualdades (ODS 10)

El estudio también contribuye a la reducción de desigualdades (ODS 10) al mejorar el acceso a tecnologías avanzadas para la identificación de mutaciones en contextos de datos limitados y desbalanceados. La implementación de técnicas que permiten la adaptación de modelos de aprendizaje automático a conjuntos de datos pequeños y desbalanceados abre oportunidades para la investigación en enfermedades raras, facilitando la igualdad en el acceso a herramientas científicas avanzadas y promoviendo un impacto global más equitativo en la salud.

A.0.4. Innovación e infraestructura (ODS 9)

El estudio está estrechamente relacionado con el ODS 9, centrado en la industria, la innovación y la infraestructura. El desarrollo y la implementación de técnicas de aprendizaje automático avanzadas para la predicción de mutaciones patogénicas representan un avance significativo en la infraestructura científica. La creación de un modelo de CNN altamente preciso y robusto contribuye al avance tecnológico en biología computacional, destacando la importancia de la innovación en la resolución de problemas complejos en la investigación genética.

En conclusión, el estudio de la retinosis pigmentaria y el uso de técnicas avanzadas de aprendizaje automático se alinea con varios Objetivos de Desarrollo Sostenible, al contribuir a la salud y el bienestar, mejorar la educación, reducir desigualdades y fomentar la innovación. Esta integración de objetivos de desarrollo sostenible en la investigación científica destaca la importancia de abordar desafíos globales mediante avances tecnológicos y científicos que tienen un impacto positivo en la sociedad.