



Universidad Internacional de la Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster en Inteligencia Artificial

Imputación de Datos
Genómicos Faltantes con
técnicas de Aprendizaje Au-
tomático

Trabajo Fin de Estudios

Presentado por: Celia Cabello Collado

Dirigido por: Lucía Prieto Santamaría

Co-dirigido por: Almudena Ruiz Iniesta

Ciudad: Alicante

Fecha: 11 de septiembre 2024

Índice de Contenidos

| | |
|--|-------------|
| Resumen | xi |
| Abstract | xiii |
| Agradecimientos | xv |
| 1 Introducción | 1 |
| 1.1 Descripción general | 1 |
| 1.2 Motivación | 2 |
| 1.3 Objetivos | 2 |
| 1.4 Cronología | 3 |
| 1.4.1 Fase inicial | 4 |
| 1.4.2 Fase de preparación | 4 |
| 1.4.3 Fase de implementación | 5 |
| 1.4.4 Fase de experimentación | 5 |
| 2 Estado del Arte | 7 |
| 2.1 Imputación Genómica | 7 |
| 2.1.1 Secuenciación de ADN | 7 |
| 2.2 Métodos de Imputación Genómica | 9 |
| 2.2.1 Métodos Tradicionales | 9 |
| 2.2.2 Métodos de Imputación Genómica Basados en <i>Machine Learning</i> (ML) | 12 |
| 3 Materiales y Métodos | 19 |
| 3.1 Estudio de referencia | 19 |
| 3.1.1 Datos genéticos | 19 |
| 3.2 Algoritmos de comparativa | 20 |
| 3.2.1 <i>K-Nearest Neighbors</i> (KNN) | 20 |
| 3.2.2 <i>Random Forest</i> (RF) | 21 |
| 3.2.3 XGBoost | 22 |
| 3.2.4 Sparse Convolutional Attention Denoising Autoencoder (SCADA) . | 22 |
| 3.3 Hardware y software | 23 |
| 3.4 Desarrollo y control de cambios | 23 |

| | | |
|----------|--|-----------|
| 4 | Experimentación | 25 |
| 4.1 | Manejo de datos | 25 |
| 4.1.1 | Preprocesamiento de los datos | 25 |
| 4.1.2 | Almacenamiento de los datos | 26 |
| 4.2 | Proceso de entrenamiento y prueba de modelos | 26 |
| 4.2.1 | <i>K-Nearest Neighbors</i> (KNN) | 26 |
| 4.2.2 | <i>Random Forest</i> (RF) | 27 |
| 4.2.3 | <i>Random Forest</i> (RF) por columna | 28 |
| 4.2.4 | XGBoost | 29 |
| 4.2.5 | Entrenamiento y prueba del modelo XGBoost | 29 |
| 4.2.6 | Sparse Convolutional Denoising Autoencoder | 30 |
| 4.2.7 | Sparse Convolutional Attention Denoising Autoencoder | 31 |
| 5 | Resultados | 35 |
| 5.1 | Métricas Utilizadas para la Evaluación de Modelos | 35 |
| 5.1.1 | <i>Medium Square Error</i> (MSE) | 35 |
| 5.1.2 | <i>Medium Absolut Error</i> (MAE) | 35 |
| 5.1.3 | Coefficiente de Determinación (R2) | 36 |
| 5.1.4 | Accuracy | 36 |
| 5.1.5 | F1-Score | 36 |
| 5.1.6 | Precisión | 37 |
| 5.1.7 | Recall (Sensibilidad) | 37 |
| 5.2 | Resultados individuales | 37 |
| 5.2.1 | <i>K-Nearest Neighbors</i> (KNN) | 37 |
| 5.2.2 | <i>Random Forest</i> (RF) | 39 |
| 5.2.3 | Análisis de los resultados de <i>Random Forest</i> (RF) | 41 |
| 5.2.4 | XGBoost | 42 |
| 5.2.5 | Análisis de los resultados de XGBoost | 43 |
| 5.2.6 | <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) | 44 |
| 5.2.7 | Análisis de los Resultados de SCDA | 49 |
| 5.2.8 | <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) | 51 |
| 5.2.9 | Comparación del Modelo con Conjunto Completo y Subconjunto de Datos | 56 |

| | | |
|----------|--|-----------|
| 5.3 | Comparativa General | 57 |
| 5.3.1 | Comparativa de Modelos de ML Tradicionales | 57 |
| 5.3.2 | Comparativa de Modelos de DL: <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) y <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) | 58 |
| 6 | Conclusiones y Trabajo Futuro | 61 |
| 6.1 | Proceso de Adquisición y Procesamiento de Datos | |
| | Genéticos: Retos y Limitaciones superadas | 61 |
| 6.2 | Conclusiones del Proyecto | 62 |
| 6.3 | Trabajo Futuro | 63 |
| 6.3.1 | Integración con Datos Reales y Heterogéneos | 63 |
| 6.3.2 | Mejora en la Interpretabilidad de Resultados | 63 |
| 6.3.3 | Aplicaciones Clínicas Potenciales | 63 |
| 6.3.4 | Exploración de Nuevas Tecnologías: Transformers y <i>Generative Adversarial Networks</i> (GAN) | 63 |
| 6.3.5 | Mejora de la Interpretabilidad del Modelo | 64 |
| 7 | Anexo 1. Repositorio del proyecto | 69 |
| 8 | Anexo 2. Certificado de idoneidad ética de la investigación | 71 |

Índice de Ilustraciones

| | | |
|-----|--|----|
| 1.1 | Cronología del proyecto (Fuente: Elaboración Propia) | 3 |
| 1.2 | Fase inicial del proyecto (Fuente: Elaboración Propia) | 4 |
| 1.3 | Fase de preparación del proyecto (Fuente: Elaboración Propia) | 5 |
| 1.4 | Fase de implementación del proyecto (Fuente: Elaboración Propia) | 5 |
| 1.5 | Fase de experimentación del proyecto (Fuente: Elaboración Propia) | 6 |
| 3.1 | Ejemplo de los datos: muestra de 10 filas y 5 columnas (Fuente: Elaboración Propia) | 20 |
| 5.1 | Gráficos de <i>Medium Square Error</i> (MSE), <i>Medium Absolut Error</i> (MAE) y R2 frente al número de vecinos para diferentes niveles de missingness.(Fuente: Elaboración Propia) | 39 |
| 5.2 | Gráficos comparativos de las métricas Accuracy, F1-Score, Precision y Recall frente al porcentaje de valores faltantes para el modelo de <i>Random Forest</i> (RF) único (Fuente: Elaboración Propia). | 40 |
| 5.3 | Gráficos comparativos de las métricas Accuracy, F1-Score, Precision y Recall frente al porcentaje de valores faltantes para el modelo de <i>Random Forest</i> (RF) por columna (Fuente: Elaboración Propia). | 41 |
| 5.4 | Gráficos comparativos de las métricas Accuracy, F1-Score, Precision y Recall frente al porcentaje de valores faltantes para el modelo de XGBoost (Fuente: Elaboración Propia) | 43 |
| 5.5 | Pérdida de entrenamiento y validación para el conjunto completo en <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) (Fuente: Elaboración Propia). | 45 |
| 5.6 | Precisión de entrenamiento y validación para el conjunto completo en SCDA.(Fuente: Elaboración Propia) | 46 |
| 5.7 | Precisión promedio por porcentaje de pérdida de datos entrenado con conjunto completo de datos (Fuente: Elaboración Propia) | 47 |
| 5.8 | Pérdida de entrenamiento y validación en SCDA (Fuente: Elaboración propia) | 48 |
| 5.9 | Precisión de entrenamiento y validación para subconjuntos en SCDA, reflejando una alta capacidad de clasificación a pesar de la reducción de datos (Fuente: Elaboración Propia). | 48 |

| | | |
|------|---|----|
| 5.10 | Precisión promedio por porcentaje de pérdida de datos entrenado con subconjunto de datos(Fuente: Elaboración Propia). | 49 |
| 5.11 | Pérdida de entrenamiento y validación para el conjunto completo en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) (Fuente: Elaboración Propia) | 52 |
| 5.12 | Precisión de entrenamiento y validación para el conjunto completo en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) (Fuente: Elaboración Propia). | 52 |
| 5.13 | Gráfico de la precisión promedio en función del porcentaje de pérdida de datos para con el conjunto completo (Fuente: Elaboración Propia). | 53 |
| 5.14 | Gráfico de la evolución de la pérdida de entrenamiento y validación durante el entrenamiento con subconjunto de datos en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) (Fuente: Elaboración Propia) . | 54 |
| 5.15 | Gráfico de la evolución del accuracy de entrenamiento y validación durante el entrenamiento con subconjunto de datos en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) (Fuente: Elaboración Propia). | 55 |
| 5.16 | Gráfico de la precisión promedio en test observada a medida que aumenta el porcentaje de pérdida de datos en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) (Fuente: Elaboración Propia). | 56 |

Índice de Tablas

| | | |
|------|--|----|
| 3.1 | Especificaciones técnicas MacBook Pro 16", 2021 | 23 |
| 5.1 | Tabla de métricas <i>Medium Square Error</i> (MSE), <i>Medium Absolut Error</i> (MAE) y R2 para diferentes configuraciones de número de vecinos y niveles de missingness en el modelo <i>K-Nearest Neighbors</i> (KNN) | 38 |
| 5.2 | Métricas de rendimiento del modelo de <i>Random Forest</i> (RF) entrenado como un solo modelo para todas las columnas. | 40 |
| 5.3 | Métricas de rendimiento de los modelos de <i>Random Forest</i> (RF) entrenados por columna. | 41 |
| 5.4 | Métricas de rendimiento del modelo de XGBoost para la imputación de valores faltantes en sets de datos con distinto porcentaje de pérdida. | 43 |
| 5.5 | Métricas finales de entrenamiento y validación para <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) con el conjunto completo de datos. | 44 |
| 5.6 | Precisión promedio por porcentaje de pérdida de datos entrenado con el conjunto completo. | 46 |
| 5.7 | Métricas finales de entrenamiento y validación para SCDA estranado con subconjunto de datos. | 47 |
| 5.8 | Precisión promedio por porcentaje de pérdida de datos entrenado con subconjuntos de datos. | 49 |
| 5.9 | Métricas finales de entrenamiento y validación para <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) con el conjunto completo de datos. | 51 |
| 5.10 | Precisión promedio observada a medida que aumenta el porcentaje de pérdida de datos en el conjunto completo en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA). | 53 |
| 5.11 | Métricas finales en entrenamiento y validación para <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) con subconjunto de datos. | 54 |
| 5.12 | Precisión promedio en test observada a medida que aumenta el porcentaje de pérdida de datos en <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA). | 55 |

| | | |
|------|---|----|
| 5.13 | Comparativa de métricas de rendimiento para los modelos <i>Random Forest</i> (RF) y XGBoost con diferentes niveles de missingness. | 58 |
| 5.14 | Comparativa de métricas de precisiones de entrenamiento y validación para <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) y <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) con conjuntos completos y reducidos. | 59 |
| 5.15 | Comparativa de métricas de precisión promedio por porcentaje de pérdida de datos para <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) y <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) entrenados con el conjunto completo. | 59 |
| 5.16 | Comparativa de métricas de precisión promedio por porcentaje de pérdida de datos para <i>Sparse Convolutional Denoising Autoencoder</i> (SCDA) y <i>Sparse Convolutional Attention Denoising Autoencoder</i> (SCADA) entrenados con subconjunto de datos. | 60 |

Lista de Acrónimos

CNN *Convolutional Neural Networks*

DL *Deep Learning*

GAN *Generative Adversarial Networks*

GWAS *Genome-wide association study*

KNN *K-Nearest Neighbors*

MAE *Medium Absolut Error*

ML *Machine Learning*

MLP *Multilayer Perceptron*

MSE *Medium Square Error*

NGS *Next Generation Sequencing*

NIDDK *National Institute of Diabetes and Digestive and Kidney Diseases*

RF *Random Forest*

RNN *Recurrent Neural Networks*

SCADA *Sparse Convolutional Attention Denoising Autoencoder*

SCDA *Sparse Convolutional Denoising Autoencoder*

SNPs *Single Nucleotide Polymorphisms*

SVM *Support Vector Machines*

Resumen

La imputación de datos genéticos es un proceso crucial en la investigación genómica, ya que los conjuntos de datos genéticos a menudo contienen valores faltantes debido a limitaciones en las tecnologías de secuenciación y muestreo. Imputar estos datos permite completar la información faltante, mejorando la calidad del análisis y maximizando el potencial para investigar enfermedades genéticas, así como su aplicación en medicina personalizada y estudios de asociación genética. Este trabajo tiene como objetivo comparar diversos enfoques para la imputación de datos genéticos de levaduras, empleando tanto métodos tradicionales de Machine Learning (*Machine Learning* (ML)) como técnicas avanzadas basadas en Deep Learning (*Deep Learning* (DL)), con un enfoque especial en los autoencoders. La investigación se centra en evaluar el rendimiento de estos modelos en términos de precisión y robustez, considerando distintos niveles de datos faltantes. Para ello, se han realizado experimentos con un conjunto de datos genéticos de levaduras, bien documentado, lo que permite una evaluación crítica y reproducible de los modelos. Los resultados muestran que los autoencoders, en particular el modelo propuesto en este estudio, Sparse Convolutional Attention Denoising Autoencoder (*Sparse Convolutional Attention Denoising Autoencoder* (SCADA)), superan al modelo base Sparse Convolutional Denoising Autoencoder (*Sparse Convolutional Denoising Autoencoder* (SCDA)), además de a los modelos tradicionales de ML en la imputación de datos genéticos faltantes, manteniendo una alta precisión incluso con un porcentaje elevado de datos perdidos. Finalmente, se identifican áreas clave de mejora y posibles expansiones para optimizar la imputación genómica en contextos clínicos y de investigación.

Palabras Clave: Imputación Genómica, Aprendizaje Automático, Aprendizaje Profundo, Autoencoders, Datos Faltantes.

Abstract

Genetic data imputation is a crucial process in genomic research, as genetic datasets often contain missing values due to limitations in sequencing technologies and sampling. Imputing this data allows for the completion of missing information, improving the quality of analysis and maximizing the potential for investigating genetic diseases, as well as its application in personalized medicine and genetic association studies. This work aims to compare various approaches to imputing genetic yeast data, employing both traditional Machine Learning (ML) methods and advanced techniques based on Deep Learning (DL), with a particular focus on autoencoders. The research focuses on evaluating the performance of these models in terms of accuracy and robustness, considering different levels of missing data. To this end, experiments were conducted with a well-documented yeast genetic dataset, allowing for a critical and reproducible evaluation of the models. The results show that autoencoders, particularly the model proposed in this study, Sparse Convolutional Attention Denoising Autoencoder (SCADA), outperform the baseline model Sparse Convolutional Denoising Autoencoder (SCDA), as well as traditional (ML) models in imputing missing genetic data, maintaining high accuracy even with a high percentage of missing data. Finally, key areas for improvement and potential expansions are identified to optimize genomic imputation in clinical and research contexts.

Keywords: Genomic Imputation, Machine Learning, Deep Learning, Autoencoders, Missing Data.

Agradecimientos

Quiero expresar mi más sincero y profundo agradecimiento a mi directora, Lucía Prieto Santamaría, cuya sabiduría, paciencia y apoyo no solo han sido pilares fundamentales en la realización de este trabajo, sino también una fuente de inspiración constante. También a mi co-directora, Almudena Ruiz Iniesta, cuyo compromiso ha sido clave para navegar con éxito las complejidades de la burocracia académica, facilitando cada paso de este viaje.

Debo mencionar con todo mi corazón el papel esencial de mis padres, Víctor y Soledad, que siempre son mi refugio. Vuestro amor incondicional, apoyo inquebrantable y ejemplos diarios de trabajo duro y dedicación, han moldeado quién soy y hacia dónde aspiro a ir en la vida.

A Alejandro, gracias por ser mi soporte constante y mi compañero de vida; por siempre darme fuerza en los días difíciles y multiplicar mi felicidad en los fáciles.

A mis hermanas Sonia y Victoria, gracias por ser un pilar fundamental de mi vida, ofreciéndome apoyo y amor sin condiciones.

Este logro es tan vuestro como mío, y estoy muy agradecida con la vida de poder compartirlo con todos vosotros.

”Esa mente que no tiene poder sobre sí misma, no tendrá poder sobre nada.”

- Ada Lovelace

1. Introducción

En el primer capítulo se presenta el tema principal de la tesis, organizándose de la siguiente manera: en primer lugar, en la Sección 1.1 se extrae la idea fundamental del proyecto, En la Sección 1.2 se explica la motivación que ha impulsado su desarrollo. En la Sección 1.3 se detallan los objetivos de la tesis. En la Sección 1.4 se muestra la cronología de las diferentes fases que han constituido el desarrollo del proyecto durante los últimos meses.

1.1 Descripción general

La imputación de datos genéticos, que consiste en estimar genotipos no observados a partir de marcadores conocidos, es un proceso crucial en el ámbito de la investigación genética y la medicina personalizada. Con el aumento exponencial de los datos genómicos, se ha incrementado la necesidad de contar con métodos eficientes, precisos y escalables que permitan imputar estos datos de manera fiable. En este contexto, los avances en técnicas de Aprendizaje Profundo (en inglés, DL) han abierto nuevas oportunidades para mejorar la precisión en la imputación de datos genéticos (Zhou and Troyanskaya, 2015).

Este proyecto se centra en evaluar y comparar distintas metodologías de imputación de datos genéticos, incluyendo enfoques tradicionales de Aprendizaje Automático (en inglés, ML) y modelos avanzados basados en DL, como los *autoencoders*. Se busca realizar un análisis crítico del desempeño de estas técnicas en diferentes escenarios, destacando tanto sus fortalezas como sus limitaciones, con el objetivo de avanzar en el conocimiento del campo de la imputación genética.

El proyecto parte de la importancia que la imputación tiene en los Estudios de Asociación de Genoma Completo (en inglés, *Genome-wide association study* (GWAS)) y su aplicación en la genómica de precisión. A continuación, se presenta una visión general sobre cómo el aprendizaje automático se ha adaptado a la naturaleza y complejidad de los datos genómicos, y cómo las técnicas más recientes, como los autoencoders, ofrecen soluciones prometedoras para abordar el reto de la imputación de datos faltantes en estos escenarios.

1.2 Motivación

Actualmente, la explosión en el volumen y la complejidad de los datos genómicos exige el desarrollo de métodos de imputación más precisos y eficientes. Los enfoques tradicionales han demostrado tener limitaciones en términos de escalabilidad y precisión, lo que ha impulsado la exploración de técnicas más avanzadas, particularmente en el campo del ML. La capacidad inherente de los modelos de DL para capturar y modelar relaciones complejas entre los marcadores genéticos ofrece una oportunidad emocionante para mejorar significativamente la imputación genómica y, por ende, su aplicabilidad en áreas tan importantes como la medicina personalizada.

Los modelos de aprendizaje profundo, como los descritos por (Goodfellow et al., 2014a) y (Quang and Xie, 2016), presentaron un enfoque prometedor para abordar los desafíos actuales en la imputación genómica. Al capturar y analizar de manera más efectiva las complejas interacciones entre los marcadores genéticos, estos modelos tienen el potencial de ofrecer resultados más precisos y confiables en comparación con los métodos tradicionales con bases intrínsecamente estadísticas. Esta capacidad para modelar relaciones no lineales y de alta dimensionalidad en los datos genómicos es fundamental para aprovechar al máximo la información contenida en ellos y, en última instancia, mejorar la comprensión de la genética humana.

La medicina personalizada, en concreto, se beneficiaría enormemente de avances en la imputación genómica impulsados por el DL. Al proporcionar predicciones más precisas sobre la estructura genética de un individuo particular, estas técnicas presentan una potencial mejora de selección de tratamientos y terapias adaptadas a las características genéticas específicas de cada paciente.

1.3 Objetivos

El objetivo principal de este proyecto es realizar una evaluación exhaustiva de diferentes enfoques de imputación de datos genéticos, comparando tanto los modelos tradicionales de ML como los modelos basados en DL, en particular los autoencoders. El análisis se centrará en medir el desempeño de estos métodos en términos de precisión, robustez frente a diferentes niveles de datos faltantes y su escalabilidad a conjuntos de datos genéticos complejos.

Además, se busca realizar un análisis comparativo que no solo permita identificar las limitaciones y ventajas de cada enfoque, sino que también contribuya a generar un conocimiento más avanzado sobre el modelo base utilizado en este proyecto, potenciando su capacidad de imputación de datos faltantes. El proyecto aspira a optimizar y mejorar dicho modelo a partir de los resultados obtenidos, promoviendo un avance en el campo de la imputación genómica.

Con este enfoque comparativo y crítico, el proyecto tiene como objetivo ofrecer una visión clara de las soluciones actuales para la imputación de datos genéticos, y a la vez generar recomendaciones prácticas para mejorar los modelos de DL empleados en este contexto.

1.4 Cronología

La cronología del proyecto se puede diferenciar en 4 fases, comenzando el 4 de abril de 2024, con la primera reunión del proyecto y finalizando el 11 de septiembre de 2024, con la entrega del Trabajo de Fin de Máster. Se presenta una vista general de la cronología del proyecto en la figura 1.1.

TFM. Imputación de Datos Genómicos Faltantes con técnicas de Aprendizaje Automático

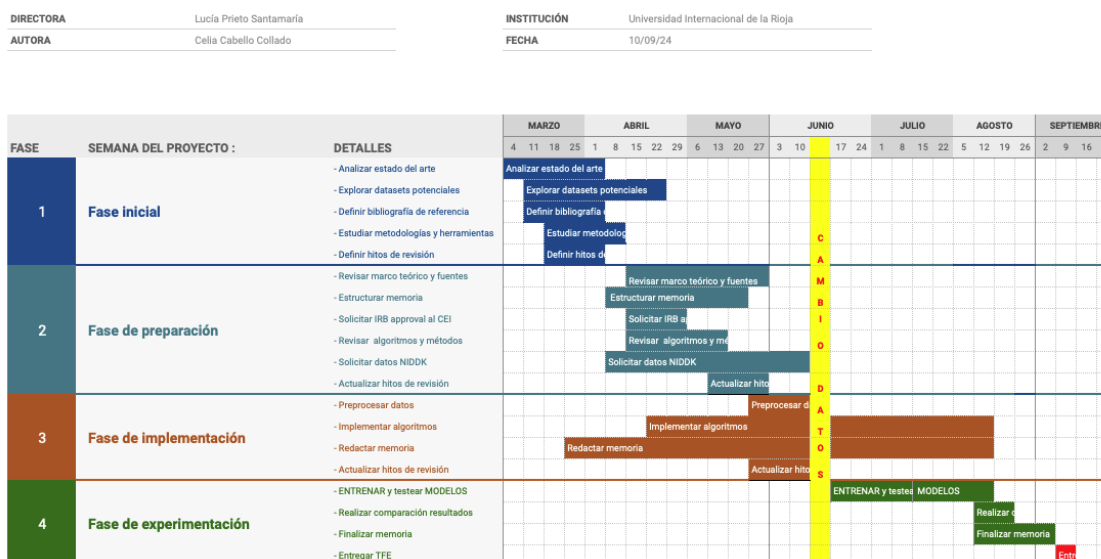


Figura 1.1: Cronología del proyecto (Fuente: Elaboración Propia)

1.4.1 Fase inicial

En esta fase (ver figura 1.2) se lleva a cabo una investigación exhaustiva sobre los temas relacionados con el proyecto, incluyendo un análisis detallado del estado del arte para decidir los datos específicos y las fuentes de donde se obtendrán. Se realiza un análisis preliminar de los datos disponibles para evaluar la viabilidad de la solicitud. Además, se inicia la estructuración del proyecto estableciendo los primeros hitos de revisión, junto con la revisión bibliográfica inicial y la selección de metodologías y herramientas a utilizar.



Figura 1.2: Fase inicial del proyecto (Fuente: Elaboración Propia)

1.4.2 Fase de preparación

En esta fase (ver figura 1.3) se establecen los cimientos del proyecto y se realiza una preparación exhaustiva. Se revisa la documentación relevante y se elabora el estado del arte. Además, se comienza a estructurar y redactar la memoria del proyecto, así como a diseñar las arquitecturas que se utilizarán en el desarrollo. Para obtener acceso a los datos genómicos privados del National Institute of Diabetes and Digestive and Kidney Diseases (*National Institute of Diabetes and Digestive and Kidney Diseases* (NIDDK)), se realiza una solicitud a la entidad gubernamental estadounidense, proceso que requiere la aprobación del comité de ética de la UNIR (CEI) a través de un acuerdo de aprobación del comité (en inglés, Institutional Review Board (IRB)). Tras proporcionar el IRB y una detallada descripción de la investigación, se obtiene la aprobación el 16 de junio de 2024 para el uso interno de los datos del NIDDK en los algoritmos del proyecto, cuyo respectivo documento se encuentra disponible en el Anexo 2 .

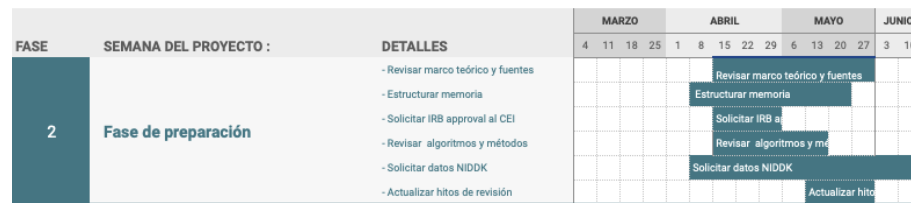


Figura 1.3: Fase de preparación del proyecto (Fuente: Elaboración Propia)

1.4.3 Fase de implementación

En esta fase (ver figura 1.4) se definen e implementan los algoritmos haciendo uso de los datos disponibles previo preprocesamiento de los mismos, a la vez que se documenta el proceso en la memoria. Inicialmente, se intentó trabajar con los datos proporcionados por el NIDDK, pero tras una evaluación detallada, se determinó que dichos datos no eran compatibles con los modelos seleccionados ni con las arquitecturas previstas para la imputación. Este análisis concluyó que las características de los datos no se alineaban con los requisitos necesarios para llevar a cabo una comparativa eficaz entre los diferentes enfoques. Así, se cambia el paradigma de datos y se empiezan a utilizar datos genéticos de levaduras del estudio SCDA (Chen and Shi, 2019), tomándose este modelo como base para la comparativa, procediendo a implementar otras arquitecturas.



Figura 1.4: Fase de implementación del proyecto (Fuente: Elaboración Propia)

1.4.4 Fase de experimentación

Durante esta fase final (ver figura 1.5), se ejecutan pruebas exhaustivas de los modelos implementados para evaluar su rendimiento en la imputación de datos genéticos. Esta etapa involucra entrenar y testear los modelos. Se llevan a cabo múltiples ciclos de evaluación para asegurar la validez y reproducibilidad de los resultados. Cada experimento se documenta meticulosamente para garantizar que los hallazgos puedan ser verificados y comparados de manera eficaz. Los resultados obtenidos de estos experimentos se sintetizan en esta memoria, resaltando las capacidades y limitaciones de cada enfoque.

La culminación de esta fase es la preparación del documento final y la entrega del

Trabajo de Fin de Máster, donde se consolidan todas las observaciones, análisis y recomendaciones derivadas de la investigación realizada.



Figura 1.5: Fase de experimentación del proyecto (Fuente: Elaboración Propia)

2. Estado del Arte

En este segundo capítulo, se abordan dos disciplinas principales. En la Sección 2.1 se explora el concepto de imputación genómica, analizando sus implicaciones y su evolución en el campo de la genética. Posteriormente, en la Sección 2.2 se realiza un análisis de los métodos clásicos de imputación, proporcionando así el fundamento necesario para discutir los enfoques basados en ML dentro del ámbito de la Inteligencia Artificial (IA), además de hacer especial énfasis en los enfoques más recientes y avanzados, fundamentados en DL.

2.1 Imputación Genómica

La imputación genómica es un proceso crucial en la investigación genética moderna, destinado a inferir genotipos faltantes a partir de información parcial de secuencias de ADN (Browning and Browning, 2009). Este enfoque de completación de datos desempeña un papel fundamental en la mejora y la integridad de conjuntos de datos genómicos, posibilitando análisis detallados de la variabilidad genética y sus asociaciones con fenotipos observables (Manolio et al., 2009). Para comprender plenamente la importancia de la imputación genómica, es fundamental primero definir el proceso de secuenciación del ADN y explorar las causas subyacentes de la ausencia de datos en estudios genéticos (Mardis, 2008).

2.1.1 Secuenciación de ADN

La secuenciación de ADN consiste en determinar el orden exacto de los nucleótidos (adenina, timina, citosina y guanina) en una molécula de ADN (Alberts et al., 2014). Este proceso es esencial para comprender la composición genética de un organismo y su relación con los fenotipos observados, es decir, las características visibles o medibles del organismo.

La secuenciación de nueva generación (en inglés, *Next Generation Sequencing* (NGS)) se fundamenta en técnicas que permiten secuenciar millones de fragmentos de ADN simultáneamente, generando grandes volúmenes de datos susceptibles de análisis computacional (Goodwin et al., 2016). Esto contrasta con las técnicas más antiguas, como la secuenciación de Sanger, desarrollada en los años 70, un proceso mucho más lento y costoso ya que solo permite secuenciar un fragmento de ADN a la vez, por lo que limita en gran medida la velocidad generar datos (Sanger et al., 1977).

La NGS utiliza diversas plataformas tecnológicas avanzadas, como secuenciadores de última generación que pueden procesar millones de fragmentos de ADN en una sola ejecución. Esta capacidad de alto rendimiento permite obtener datos genómicos completos de individuos, poblaciones o incluso especies enteras de manera rápida y eficiente.

Los datos generados por NGS son extremadamente complejos y requieren análisis computacionales avanzados para ser interpretados adecuadamente. Esto implica el uso de algoritmos bioinformáticos y herramientas estadísticas para ensamblar las secuencias, identificar variantes genéticas (diferencias en el ADN que pueden influir en características o enfermedades), realizar anotaciones funcionales (asignar funciones a diferentes partes del ADN) y comparar datos entre diferentes muestras o condiciones experimentales.

Algunos puntos clave sobre el estado actual de la secuenciación incluyen:

- **Tecnologías Emergentes:** Las tecnologías emergentes en secuenciación de ADN han avanzado más allá de la secuenciación de nueva generación (NGS) hacia lo que se conoce como tecnologías de tercera generación. Estas nuevas tecnologías, como la secuenciación de una sola molécula en tiempo real (SMRT) y la secuenciación de nanoporos, permiten secuenciar genomas completos de manera más rápida y económica. Además, pueden leer secuencias de ADN mucho más largas sin necesidad de fragmentarlas, lo cual simplifica el proceso y mejora la precisión (Shendure and Ji, 2008).
- **Aplicaciones Clínicas:** La secuenciación de nueva generación ha revolucionado el diagnóstico genético y la medicina de precisión. Permite identificar variantes genéticas asociadas con enfermedades hereditarias y complejas, facilitando la personalización de tratamientos y mejorando la predicción de respuestas terapéuticas. Esto permite a los facultativos desarrollar planes de tratamiento específicos basados en la composición genética única de cada paciente (Meienberg et al., 2016).
- **Avances en Análisis Bioinformático:** Los avances en análisis bioinformático han sido cruciales para manejar y analizar los grandes volúmenes de datos generados por las tecnologías de secuenciación de ADN. Estos avances han mejorado la precisión de la alineación de secuencias (el proceso de ensamblar fragmentos de ADN en un orden correcto), los métodos de detección de variantes genéticas y la integración de datos multiómicos (combinación de datos de ADN con otros tipos de datos biológicos) para obtener un entendimiento más completo de la biología molecular (Goodwin et al.,

2016).

2.2 Métodos de Imputación Genómica

Históricamente, se han desarrollado varios enfoques para la imputación genómica, que se pueden categorizar en métodos tradicionales y métodos basados en aprendizaje automático. Los métodos tradicionales se basan principalmente en información de haplotipos, modelos estadísticos y enfoques bayesianos, utilizando datos de referencia para predecir los genotipos faltantes. Estos métodos han sido ampliamente utilizados y validados en numerosos estudios genómicos.

Con el avance de la tecnología y el surgimiento de técnicas de aprendizaje automático, han aparecido nuevos métodos de imputación que pueden manejar grandes volúmenes de datos y capturar patrones complejos en los datos genéticos. Estos métodos incluyen técnicas de soporte vectorial, algoritmos basados en árboles, redes neuronales profundas y otros modelos avanzados que han mostrado una gran promesa en la mejora de la precisión y eficiencia de la imputación genómica.

En esta sección, se revisarán tanto los métodos tradicionales como los métodos modernos basados en aprendizaje automático para la imputación genómica, destacando sus principios fundamentales, aplicaciones y ejemplos prácticos.

2.2.1 Métodos Tradicionales

Los métodos clásicos de imputación genómica han sido fundamentales en el análisis de datos genéticos durante décadas. Estos métodos permiten inferir los genotipos faltantes en estudios genómicos, mejorando así la calidad y la potencia estadística de los análisis genéticos. A continuación, se describen algunos de los enfoques más destacados:

- **Imputación basada en haplotipos:** Este enfoque utiliza información sobre los haplotipos presentes en la población para inferir los genotipos faltantes. Un haplotipo es un conjunto de marcadores genéticos o variaciones del ADN estrechamente relacionados presentes en un cromosoma que tienden a heredarse juntos ¹. Los algoritmos como Beagle (Browning and Browning, 2009) y MACH (Li et al., 2009) construyen una referencia de haplotipos a partir de datos conocidos y luego utilizan esta referencia para imputar los datos faltantes en individuos genotipados de

¹<https://www.cancer.gov/espanol/publicaciones/diccionarios/diccionario-genetica/def/haplotipo>

manera incompleta. La idea es que los segmentos de haplotipos observados en la referencia también estarán presentes en los individuos del estudio, permitiendo así la imputación.

Si en una población se ha identificado que un conjunto de haplotipos está siempre asociado con ciertos genotipos, y sabemos que un individuo tiene parte de ese haplotipo, podemos imputar los genotipos faltantes basándonos en las probabilidades derivadas de los datos de referencia, por lo tanto, la probabilidad de que un individuo i tenga un genotipo G_i dado el conjunto de haplotipos H 2.1 se calcula como la suma de las probabilidades de G_i dado cada haplotipo H_j en la referencia

$$P(G_i|H) = \sum_{j=1}^n P(G_i|H_j) \quad (2.1)$$

- **Imputación por métodos estadísticos:** Estos métodos se basan en modelos estadísticos para predecir los genotipos ausentes a partir de la información disponible en los datos. Un enfoque común es el uso de modelos de regresión o matrices de correlación entre Polimorfismos de nucleótido único ((en inglés, *Single Nucleotide Polymorphisms* (SNPs)) .

Un SNPs es una variación en un solo nucleótido que ocurre en una posición específica del genoma. Por ejemplo, en una secuencia de ADN, la mayoría de los individuos pueden tener la base adenina (A) en una posición particular, mientras que otros pueden tener la base guanina (G). Los SNPs son los tipos más comunes de variación genética entre individuos y pueden actuar como marcadores biológicos, ayudando a localizar genes asociados con enfermedades.

Los métodos estadísticos de imputación utilizan estas variaciones comunes (SNPs) para inferir los genotipos faltantes en los datos genómicos. Por ejemplo, el algoritmo SNPTools (Soares et al., 2012) emplea un modelo de regresión para estimar los genotipos faltantes, mientras que otros métodos utilizan la matriz de correlación de los SNPs (Li et al., 2011), representada en la ecuación 2.2.

$$\hat{G} = X\beta \quad (2.2)$$

donde \hat{G} es el genotipo imputado, X es una matriz de genotipos observados y β es un vector de coeficientes de regresión estimados. En este contexto, X representa los datos observados de los SNPs para un conjunto de individuos, y β son los coeficientes que

capturan las relaciones estadísticas entre estos SNPs. Utilizando estos coeficientes, el modelo puede predecir los genotipos faltantes (\hat{G}) para los individuos en los que algunos datos genómicos no están disponibles.

- **Enfoques bayesianos:** Los métodos bayesianos utilizan la inferencia bayesiana para estimar los genotipos faltantes, incorporando información previa sobre la distribución de los genotipos en la población. Este enfoque combina datos observados con conocimientos previos para actualizar la probabilidad de los genotipos faltantes. Un algoritmo conocido que emplea este enfoque es IMPUTE2 (Howie et al., 2009), que utiliza un modelo de Markov oculto (HMM) para representar la estructura de haplotipos y aplicar inferencia bayesiana en la imputación de genotipos.

La inferencia bayesiana se basa en la siguiente fórmula, conocida como el teorema de Bayes:

$$P(G_i|D) \propto P(D|G_i)P(G_i) \quad (2.3)$$

donde $P(G_i|D)$ es la probabilidad posterior del genotipo G_i dado los datos D , $P(D|G_i)$ es la verosimilitud de los datos D dados los genotipos G_i y $P(G_i)$ es la probabilidad a priori del genotipo G_i .

En términos simples, la probabilidad posterior ($P(G_i|D)$) se obtiene multiplicando la verosimilitud ($P(D|G_i)$) por la probabilidad a priori ($P(G_i)$). Esto significa que podemos ajustar nuestras estimaciones de los genotipos faltantes a medida que incorporamos nueva información.

El algoritmo IMPUTE2 (Howie et al., 2009) utiliza un modelo de Markov oculto para capturar la dependencia entre haplotipos adyacentes. Un modelo de Markov oculto es un tipo de modelo estadístico que asume que el sistema que se modela es un proceso de Markov con estados ocultos no observables. En este contexto, los estados ocultos representan los haplotipos y las transiciones entre estados reflejan cómo los haplotipos se heredan a lo largo del genoma.

Por ejemplo, suponiendo la imputación de genotipos faltantes en un nuevo conjunto de datos genómicos, se utilizan datos previos de una población para estimar la distribución de los genotipos, lo que proporciona la probabilidad a priori ($P(G_i)$). Luego, se recopilan datos observados (D) de los individuos en el nuevo conjunto de datos. Aplicando el teorema de Bayes, se combinan los datos observados con nues-

tras estimaciones previas para obtener la probabilidad posterior ($P(G_i|D)$), que da las estimaciones más probables de los genotipos faltantes.

2.2.2 Métodos de Imputación Genómica Basados en ML

Con los avances en el campo del ML, han surgido nuevos enfoques para la imputación genómica que aprovechan la capacidad de estos modelos para capturar patrones complejos en los datos genéticos, que son difíciles de capturar. Algunos de estos métodos incluyen:

- **K-Nearest Neighbors (*K-Nearest Neighbors* (KNN))**: es un algoritmo de aprendizaje basado en instancias que predice los valores faltantes de una muestra basándose en los genotipos de sus vecinos más cercanos. KNN) utiliza una métrica de distancia, como la distancia euclidiana, para identificar las muestras más similares en el conjunto de datos de entrenamiento. Luego, el valor del genotipo faltante se imputa en función de una votación mayoritaria o una media ponderada de los genotipos de los vecinos más cercanos (Troyanskaya et al., 2001a). En el contexto de la imputación genómica, KNN) es útil debido a su capacidad para aprovechar la correlación entre muestras genéticas.

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2} \quad (2.4)$$

donde $d(x_i, x_j)$ es la distancia entre las muestras x_i y x_j , y n es el número de características observadas (genotipos).

KNN) se ha utilizado en numerosos estudios (Troyanskaya et al., 2001b),(McKay et al., 2007),(Kojima et al., 2020) para imputar datos genómicos, ya que permite estimar genotipos faltantes basándose en la similitud genética entre los individuos, sin necesidad de asumir una distribución específica de los datos.

- **Modelos de Máquinas de vectores de Soporte (en ingles, *Support Vector Machines* (SVM))**: son técnicas de aprendizaje automático que funcionan encontrando un hiperplano en un espacio de alta dimensión que separa las clases de datos. En el contexto de la imputación genómica, las SVM pueden ser entrenadas para distinguir entre diferentes genotipos y predecir los genotipos faltantes de un individuo basándose en sus características observadas (Browning and Browning, 2009).

El objetivo de una SVM es maximizar el margen, es decir, la distancia entre el hiperplano separador y los puntos de datos más cercanos de cualquier clase, conocidos como vectores de soporte. La función objetivo que se minimiza durante el entrenamiento de una SVM se representa como:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i(w \cdot x_i - b)) \quad (2.5)$$

donde: - w es el vector de pesos, - b es el sesgo, - C es un parámetro de regularización que controla el equilibrio entre maximizar el margen y minimizar el error de clasificación, - y_i es la etiqueta de clase del i -ésimo individuo, - x_i es el vector de características del i -ésimo individuo.

Simplificando, la fórmula busca un hiperplano que no solo separe las clases de manera efectiva sino que también minimice los errores de clasificación, ponderados por C .

En un estudio genómico, se puede utilizar una SVM para imputar genotipos faltantes entrenando el modelo con datos de individuos cuyos genotipos son completamente conocidos. Una vez entrenado, el modelo puede aplicarse a individuos con datos genotípicos incompletos. Por ejemplo, si se tiene información sobre ciertos marcadores genéticos observados, la SVM puede predecir los marcadores faltantes con base en las relaciones aprendidas durante el entrenamiento.

- **Métodos basados en árboles:** técnicas de ML que utilizan estructuras de árbol para tomar decisiones. Un árbol de decisión es un modelo predictivo que divide repetidamente los datos en subconjuntos más pequeños basándose en un conjunto de reglas de decisión derivadas de las características de los datos. Estos métodos son populares debido a su simplicidad, interpretabilidad y capacidad para manejar tanto datos numéricos como categóricos.

- **Random Forest (*Random Forest* (RF)):** método de ensemble que utiliza múltiples árboles de decisión para mejorar la precisión y la robustez de las predicciones (Breiman, 2001). Cada árbol en el bosque se entrena en un subconjunto diferente de los datos mediante un proceso llamado bootstrap aggregating (bagging), donde se seleccionan aleatoriamente muestras con reemplazo. Las predicciones de todos los árboles se promedian para obtener la predicción final.

La fórmula para la predicción agregada es:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^T f_t(x) \quad (2.6)$$

donde T es el número de árboles en el bosque y $f_t(x)$ es la predicción del t -ésimo árbol para la entrada x .

Al utilizar RF para imputar genoma, cada árbol de decisión en el bosque puede imputar diferentes partes del genoma basándose en las relaciones observadas en los datos de entrenamiento. Al promediar las predicciones de todos los árboles, se obtiene una imputación robusta, mejorando la generalización (y por ende disminuyendo el sobreajuste).

- **XGBoost**: algoritmo de boosting basado en árboles que ha demostrado ser altamente efectivo para la imputación genómica debido a su capacidad para manejar grandes conjuntos de datos y capturar relaciones no lineales (Chen and Guestrin, 2016). A diferencia de RF, que construye árboles de decisión independientes, XGBoost construye árboles secuencialmente, donde cada nuevo árbol corrige los errores de los árboles anteriores. XGBoost optimiza una función objetivo que combina una pérdida de entrenamiento y un término de regularización para evitar el sobreajuste:

$$\mathcal{L}(\theta) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \quad (2.7)$$

donde l es la pérdida de entrenamiento, \hat{y}_i es la predicción para el i -ésimo individuo, Ω es el término de regularización, y f_k representa las funciones de decisión del modelo.

En este caso, al construir los árboles de manera secuencial y ajustar iterativamente los errores de predicción, XGBoost logra una mayor precisión en la imputación de genotipos que métodos como RF, siendo especialmente útil para manejar grandes volúmenes de información genómica y capturar interacciones complejas entre las variables.

- **Métodos basados en DL**: se trata de técnicas avanzadas de ML que utilizan redes neuronales profundas para modelar y aprender patrones complejos en los datos. Estos métodos han demostrado ser particularmente eficaces para tareas de imputación genómica debido a su capacidad para manejar grandes volúmenes de datos y capturar relaciones no lineales entre los genotipos.

- **Perceptrón multicapa (en inglés, *Multilayer Perceptron* (MLP))**: es un tipo de red neuronal profunda que consta de múltiples capas de neuronas. Cada capa aplica una transformación lineal seguida de una función de activación no lineal, lo que permite al MLP aprender representaciones complejas de los datos genéticos (Angermueller et al., 2016).

$$y = \sigma(Wx + b) \quad (2.8)$$

donde W es la matriz de pesos, x es el vector de entrada, b es el vector de sesgos, y σ es la función de activación.

Un MLP puede ser entrenado para aprender las relaciones entre diferentes SNPs y utilizar esta información para predecir los genotipos faltantes. Esto se logra alimentando la red con datos conocidos y ajustando los pesos para minimizar el error de predicción.

- **Redes Neuronales Convolucionales (en inglés, *Convolutional Neural Networks* (CNN))**: son especialmente útiles para la imputación de datos genómicos de imágenes, como los datos de microscopía de barrido de ADN (Libbrecht and Noble, 2015). Las CNN aplican convoluciones con filtros aprendidos para extraer características locales de los datos, lo que les permite capturar patrones espaciales en las imágenes.

$$y = \sigma(W * x + b) \quad (2.9)$$

donde $*$ denota la operación de convolución.

Las CNN pueden ser utilizadas para analizar imágenes de secuencias de ADN y extraer características relevantes que pueden ser utilizadas para imputar genotipos faltantes. Al aprender patrones visuales específicos de los datos genómicos, las CNN pueden realizar imputaciones precisas basadas en las características observadas.

- **Redes Neuronales Recurrentes (en inglés, *Recurrent Neural Networks* (RNN))**: son redes neuronales que utilizan conexiones recurrentes para mantener un estado interno que captura información secuencial, por lo que su salida depende de los elementos anteriores dentro de una secuencia (Min et al., 2017).

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b) \quad (2.10)$$

donde h_t es el estado oculto en el tiempo t , W_h y W_x son matrices de pesos, x_t es la entrada en el tiempo t , y b es el sesgo.

Las RNN pueden ser utilizadas para modelar la dependencia secuencial entre los nucleótidos en una secuencia de ADN, lo cual es útil para imputar genotipos en secuencias largas. Esto permite capturar las relaciones temporales entre los datos genómicos y realizar predicciones basadas en la información secuencial.

- **Autoencoders:** son redes neuronales diseñadas para aprender representaciones comprimidas de los datos. Un autoencoder consiste en una red de codificación que reduce la dimensionalidad de los datos y una red de decodificación que reconstruye los datos a partir de la representación comprimida (Xing and Jordan, 2016).

$$y = \sigma(W_d \sigma(W_e x + b_e) + b_d) \quad (2.11)$$

donde W_e y W_d son las matrices de pesos de codificación y decodificación, respectivamente, y b_e y b_d son los sesgos.

Los autoencoders pueden aprender una representación comprimida de los datos genómicos y luego reconstruir los genotipos faltantes a partir de esta representación

- **Transformers:** El Transformer Vaswani et al. (2017) es una arquitectura de DL que evita la recurrencia y permite una paralelización significativamente mayor al utilizar únicamente un mecanismo de atención para capturar dependencias entre la entrada y la salida. Este modelo se ha destacado en tareas de procesamiento de lenguaje natural, aunque también presenta potencial para la imputación genómica debido a su capacidad para modelar relaciones complejas en los datos (Poplin et al., 2018).

El componente central del Transformer es el mecanismo de autoatención, que relaciona distintas posiciones de una misma secuencia para calcular una representación de la secuencia. La autoatención permite al modelo asignar diferentes pesos a diferentes partes de la secuencia de entrada al generar la salida, facilitando así la captura de dependencias a largo plazo sin necesidad de recurrencia. La arquitectura del Transformer consta de dos bloques principales: un codificador y un decodificador, cada uno formado por múltiples capas.

- * **Codificador:** Cada capa del codificador consta de dos subcapas princi-

pales: una capa de autoatención y una red neuronal feed-forward completamente conectada. La capa de autoatención permite que el codificador enfoque diferentes partes de la secuencia de entrada al mismo tiempo, mientras que la red feed-forward proporciona capacidad adicional de modelado no lineal.

- * **Decodificador:** Similar al codificador, cada capa del decodificador también incluye una capa de autoatención y una red feed-forward. Además, el decodificador incluye una capa de atención que enfoca las salidas del codificador, permitiendo que el modelo genere la secuencia de salida basada en la información codificada.

El mecanismo de atención del Transformer se define matemáticamente como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.12)$$

donde Q (queries) son las consultas, K (keys) son las claves, V (values) son los valores y d_k es la dimensionalidad de las claves.

Este mecanismo de atención calcula una ponderación para cada posición en la secuencia de entrada, permitiendo que el modelo enfoque las partes más relevantes de la secuencia al generar la salida.

- **Graph Neural Networks ():** son modelos diseñados específicamente para trabajar con datos estructurados en forma de grafos, por lo que su utilidad está en imputar datos genéticos que tienen una estructura de grafo, como los datos de interacciones gen-gen (Velickovic et al., 2018). En un grafo, los nodos representan entidades (por ejemplo, genes) y las aristas representan las relaciones entre estas entidades (por ejemplo, interacciones genéticas).

Las funcionan propagando información a través de las aristas del grafo para actualizar las representaciones de los nodos. Este proceso se lleva a cabo en múltiples capas, donde cada capa agrega información de los nodos vecinos para refinar las representaciones de los nodos. La actualización de las representaciones de los nodos en la capa $k + 1$ se puede expresar como:

$$h_i^{(k+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} W^{(k)} h_j^{(k)} + b^{(k)}\right) \quad (2.13)$$

donde $h_i^{(k)}$ es la representación del nodo i en la capa k , $\mathcal{N}(i)$ son los vecinos del

nodo i , $W^{(k)}$ son los pesos de la capa k , $b^{(k)}$ es el sesgo de la capa k y σ es una función de activación no lineal, como ReLU (Rectified Linear Unit).

En cada capa, las $W^{(k)}$ combinan las representaciones de los nodos vecinos para actualizar la representación del nodo actual, permitiendo que la red capture la estructura local del grafo y las relaciones entre los nodos.

- **Redes Generativas Adversarias (en inglés, Generative Adversarial Networks *Generative Adversarial Networks* (GAN))**: arquitectura que consiste en dos redes neuronales, una red generadora y una red discriminadora, que compiten entre sí en un proceso de aprendizaje entre adversarios. Este enfoque fue introducido por Goodfellow et al. (2014b). La red generadora (G) produce datos sintéticos, mientras que la red discriminadora (D) evalúa si los datos son reales o generados.

El objetivo de la red generadora es crear datos que sean indistinguibles de los datos reales, mientras que la red discriminadora intenta diferenciar entre los datos reales y los generados. El proceso de entrenamiento se puede formular como un problema de minimax, donde G y D juegan un juego de suma cero:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2.14)$$

donde D es la red discriminadora, G es la red generadora, x son datos reales, z es un vector de ruido muestreado de una distribución $p_z(z)$, $D(x)$ es la probabilidad asignada por la discriminadora a que x sea un dato real y $G(z)$ es el dato generado a partir del ruido z .

Durante el entrenamiento, la red generadora mejora iterativamente para producir datos más realistas, mientras que la red discriminadora se vuelve mejor en identificar datos falsos, lo que lleva a una mejora continua en la calidad de los datos generados. Las GAN pueden ser utilizadas para generar nuevas secuencias de ADN basadas en datos existentes. Esto se logra entrenando la red generadora para producir secuencias genéticas que la red discriminadora no pueda distinguir de las secuencias reales. Una vez entrenada, la red generadora puede crear datos sintéticos adicionales que complementan los datos faltantes, mejorando la integridad del conjunto de datos.

3. Materiales y Métodos

El desarrollo de este proyecto requirió una combinación de hardware adecuado, herramientas de software específicas y metodologías avanzadas para la implementación y evaluación de modelos. El capítulo se organiza en cinco secciones principales: la Sección 3.1 describe el estudio de referencia utilizado para este trabajo; la Sección 3.2 presenta los algoritmos utilizados para la comparativa; la Sección 3.3 detalla las especificaciones de hardware y software utilizados; y finalmente, la Sección 3.4 cubre el entorno de desarrollo y el control de cambios implementado a lo largo del proyecto.

3.1 Estudio de referencia

El presente trabajo se basa en el estudio de Sparse Convolutional Denoising Autoencoders (SCDA) propuesto por (Chen and Shi, 2019), que desarrolla un método de imputación genotípica mediante autoencoders convolucionales dispersos y técnicas de denoising. Este enfoque utiliza redes neuronales profundas para aprender representaciones de los datos con el objetivo de imputar variantes genéticas faltantes.

El estudio incluye un repositorio público disponible en el Anexo 2, donde se encuentran los métodos y el código empleados, lo que permite la replicación de los experimentos. Debido a la claridad en la presentación del código y a su accesibilidad, este estudio ha sido tomado como base para la implementación y comparación de diferentes métodos de imputación utilizando Machine Learning (ML) en este proyecto.

3.1.1 Datos genéticos

Los datos utilizados en este trabajo provienen del repositorio público asociado al estudio de SCDA¹ (Chen and Shi, 2019). Estos datos corresponden a genotipos de levaduras, donde las variantes genéticas están codificadas en un formato tabular. Las columnas representan las posiciones de los SNPs (Single Nucleotide Polymorphisms), mientras que las filas corresponden a los identificadores de las muestras.

En estos datos, los valores 1 y 2 indican variantes genéticas de dos cepas diferentes: BY (1), que es una cepa de laboratorio, y RM (2), una cepa aislada de un viñedo. Los valores 0 representan SNPs faltantes. A continuación se muestra un ejemplo de estos datos

¹<https://github.com/work-hard-play-harder/SCDA/tree/master/data>

en la figura 3.1.

| | 33070_chrl_33070_A_T | 33147_chrl_33147_G_T | 33152_chrl_33152_T_C | 33200_chrl_33200_C_T | 33293_chrl_33293_A_T |
|-------|----------------------|----------------------|----------------------|----------------------|----------------------|
| SAMID | | | | | |
| 01_01 | 1 | 1 | 1 | 1 | 1 |
| 01_02 | 1 | 1 | 1 | 1 | 1 |
| 01_03 | 2 | 2 | 2 | 2 | 2 |
| 01_04 | 1 | 1 | 1 | 1 | 1 |
| 01_06 | 2 | 2 | 2 | 2 | 2 |
| 01_07 | 1 | 1 | 1 | 1 | 1 |
| 01_08 | 2 | 2 | 2 | 2 | 2 |
| 01_09 | 2 | 2 | 2 | 2 | 2 |
| 01_10 | 1 | 1 | 1 | 1 | 1 |
| 01_11 | 2 | 2 | 2 | 2 | 2 |

Figura 3.1: Ejemplo de los datos: muestra de 10 filas y 5 columnas (Fuente: Elaboración Propia)

Los datos están divididos en dos conjuntos diferenciados: uno para entrenamiento (train) y otro para prueba (test). El conjunto de train cuenta con 3513 filas (muestras) y 28,220 columnas (SNPs), mientras que el conjunto de test tiene 877 filas y 28,220 columnas. Ambos conjuntos están completos, sin datos faltantes.

3.2 Algoritmos de comparativa

Esta sección describe los modelos utilizados para la imputación de genotipos faltantes, destacando su relevancia en el contexto de variantes genéticas. Cada modelo fue seleccionado por su capacidad de manejar la dimensionalidad de los datos y capturar relaciones complejas entre las características, como ocurre en los datos de *SNPs*. Se detalla cómo cada uno de estos modelos contribuye de manera efectiva a la imputación en este contexto específico.

3.2.1 KNN

El algoritmo de (*KNN*) se utiliza en este trabajo como un enfoque simple para la imputación de valores faltantes. KNN asigna un valor a un *SNPs* faltante en función de los valores de sus vecinos más cercanos en el espacio de características. La proximidad entre muestras genéticamente similares sugiere que las variantes de un *SNPs* compartido por muestras cercanas pueden ser un buen indicador de los valores faltantes. La contribución potencial en el contexto es:

- **Simplicidad y eficiencia en la captura de relaciones locales:** KNN es adecuado cuando existen patrones de similitud local entre las muestras. En genómica, las muestras genéticamente cercanas tienden a compartir patrones en los SNPs, lo que hace que KNN pueda imputar valores de manera efectiva en casos donde las relaciones globales no sean predominantes.
- **Reducción de complejidad:** Debido a que no requiere un modelo entrenado y utiliza distancias simples, KNN puede ofrecer una solución rápida para la imputación, especialmente en escenarios donde la proximidad local es suficiente para capturar las relaciones entre los datos.

3.2.2 RF

RF es un modelo basado en la creación de múltiples árboles de decisión que pueden capturar interacciones no lineales entre las características. Su capacidad para manejar grandes conjuntos de datos con muchas variables lo convierte en un modelo adecuado para la imputación de genotipos faltantes. La contribución potencial en este contexto es:

- **Captura de interacciones no lineales:** Los SNPs pueden presentar relaciones complejas y no triviales. RF es capaz de modelar estas interacciones sin necesidad de una especificación previa, lo que es útil cuando las relaciones entre los SNPs no siguen un patrón lineal.
- **Manejo de alta dimensionalidad:** RF es particularmente eficaz en datos de alta dimensionalidad como los genotípicos, ya que selecciona las características más relevantes durante el proceso de imputación, lo que permite que los árboles se centren en las variantes más significativas para cada predicción.
- **Robustez frente a valores atípicos y ruido:** Gracias a su naturaleza de ensamblaje, RF es menos sensible a los valores atípicos, lo que puede mejorar la calidad de la imputación en datos ruidosos o incompletos.

RF por columna

El entrenamiento de un modelo independiente de RF para cada columna con valores faltantes permite que el modelo se especialice en la predicción de una única variante, lo que puede resultar en imputaciones más precisas. La contribución potencial en este contexto es:

- **Especialización en variantes específicas:** Al entrenar un modelo por cada columna, el modelo se adapta mejor a las relaciones particulares de cada SNPs con las demás variantes. Esto permite una mayor precisión en la imputación, especialmente cuando diferentes SNPs muestran correlaciones distintas con otras variantes.
- **Imputación escalonada:** El uso de una estrategia escalonada, en la que los valores imputados en una columna se reutilizan como predictores para las siguientes, permite que el modelo integre progresivamente la información disponible y mejore la calidad de la imputación.

3.2.3 XGBoost

XGBoost es una técnica de *boosting* que combina múltiples árboles de decisión, construidos de manera secuencial, para corregir los errores de los árboles anteriores. Esto permite que XGBoost ajuste sus predicciones de forma iterativa, lo que puede ser útil en la imputación de datos complejos como los genotípicos. La contribución potencial en este contexto es:

- **Corrección iterativa de errores:** En datos genotípicos donde las relaciones entre variantes son complejas, XGBoost ajusta las predicciones de manera secuencial. Esto permite corregir errores y mejorar la precisión a través de múltiples iteraciones, lo que es particularmente útil en escenarios con muchos valores faltantes.
- **Modelado de relaciones no lineales:** XGBoost captura tanto relaciones lineales como no lineales entre los SNPs, lo que puede mejorar la imputación en casos donde las interacciones entre variantes no son evidentes.

3.2.4 Sparse Convolutional Attention Denoising Autoencoder (SCADA)

SCADA es una extensión del modelo original SCDA que se presenta en este estudio, que introduce un mecanismo de atención para mejorar la imputación de valores faltantes en datos genotípicos. Los autoencoders con atención permiten aprender representaciones comprimidas de los datos y reconstruirlos, asignando mayor importancia a las características más relevantes. En el contexto de la imputación de genotipos, SCADA captura patrones en los SNPs y utiliza la atención para enfocarse en las variantes clave para la imputación. La contribución de este enfoque mejorado en el contexto genotípico es la siguiente:

- **Focalización en SNPs clave:** El mecanismo de atención asigna más peso a las características más importantes, lo que permite que el modelo se enfoque en los

SNPs que tienen mayor relevancia biológica o estadística para la imputación. Esto es crucial en escenarios donde algunos SNPs aportan mayor variabilidad e información que otros.

- **Captura de relaciones complejas:** Los autoencoders, combinados con la atención, pueden detectar relaciones no triviales entre variantes genéticas. Esto mejora la capacidad del modelo para imputar valores faltantes en datos genotípicos, donde la correlación entre SNPs puede ser difícil de modelar mediante enfoques más tradicionales.

3.3 Hardware y software

La máquina destinada a el procesamiento ha sido un MacBook Pro (16 pulgadas, 2021), mediante el cual se han llevado a cabo todas las tareas relacionadas con el proyecto. Las especificaciones técnicas se muestran en la tabla 3.1.

| Componente | Especificaciones |
|----------------|---|
| CPU | Chip M1 Pro de Apple, CPU de 10 núcleos |
| RAM | 16 GB de memoria unificada |
| GPU integrada | GPU de 16 núcleos |
| Almacenamiento | SSD 512 GB |
| SO | macOS Sonoma 14.6 |

Tabla 3.1: Especificaciones técnicas MacBook Pro 16", 2021

2

3.4 Desarrollo y control de cambios

El entorno de desarrollo utilizado para la implementación del proyecto fue Visual Studio Code (VSCode), ejecutado localmente en el equipo mencionado. Para el control de versiones, se ha utilizado GitHub como repositorio remoto, donde se alojan tanto el código del proyecto como los datos generados en las distintas etapas del proceso. El repositorio es público y está disponible en la plataforma GitHub, con total transparencia y reproducibilidad del proyecto.

El repositorio, accesible en https://github.com/celiac/ML_yeast_dna_imputation, incluye instrucciones detalladas sobre el contenido, la ejecución de los notebooks y las con-

figuraciones necesarias para replicar los experimentos.

4. Experimentación

En este capítulo se describe el proceso de experimentación llevado a cabo para evaluar y comparar los diferentes modelos de imputación de datos genotípicos. La fase de experimentación se estructura en las siguientes etapas: en la Sección 4.1, se lleva a cabo el almacenamiento y preprocesamiento de los datos. A continuación, en la Sección 4.2, se detallan las decisiones técnicas clave para la implementación de los modelos de imputación.

4.1 Manejo de datos

4.1.1 Preprocesamiento de los datos

Los modelos tradicionales de ML, como XGBoost y KNN, tienen limitaciones más estrictas en cuanto al manejo de datos en memoria en comparación con los modelos basados en *Deep Learning* (DL). Debido a las restricciones de hardware disponibles, las altas demandas de procesamiento asociadas con estos modelos dificultan el uso de grandes volúmenes de datos sin reducir su dimensionalidad. Por esta razón, se adopta una estrategia de reducción de dimensionalidad para optimizar el uso de los recursos computacionales.

Se decide reducir el número de columnas de 28,000 a 1,000 en los conjuntos de datos de entrenamiento y prueba. La técnica utilizada para seleccionar las columnas es la basada en la **varianza**. Se supone que las columnas con mayor varianza son aquellas que contienen más información relevante para los modelos, por lo que se seleccionan las 1,000 columnas con mayor varianza. Estos archivos procesados se guardan para ser utilizados posteriormente durante el entrenamiento y prueba de los modelos.

En cuanto a los modelos basados en autoencoders, se realiza una experimentación utilizando ambos conjuntos de datos: uno reducido y otro con la dimensionalidad completa. Esto permite evaluar el rendimiento de los autoencoders en función de la dimensionalidad del conjunto de datos, dado que los autoencoders tienen la capacidad de manejar una mayor cantidad de características.

Adicionalmente, a partir del conjunto de prueba reducido, se generan cuatro subconjuntos con distintos porcentajes de datos faltantes (NaN), en proporciones de 10%, 20%, 30% y 40%. Estos subconjuntos también se guardan para su posterior uso en la evaluación de la calidad de imputación. El modelo entrenado con el conjunto de entrenamiento completo será utilizado para imputar los datos faltantes en estos subconjuntos, y posteri-

ormente se evaluará la calidad de la imputación utilizando el conjunto de prueba original como referencia.

4.1.2 Almacenamiento de los datos

Antes del preprocesamiento, los archivos de datos originales, `train.txt` y `test.txt`, se almacenan comprimidos en formato `.zip` en la carpeta `data/raw`, con el fin de optimizar el almacenamiento en el repositorio del proyecto cuya información se encuentra en el Anexo 2.

Después de reducir el número de columnas en los conjuntos de entrenamiento y prueba, y de generar los subconjuntos con distintos niveles de pérdida de datos, los archivos resultantes se guardan en la carpeta `data/processed`. El formato elegido para el almacenamiento de los archivos es `.parquet`, debido a su alta capacidad de compresión y a la velocidad de lectura y escritura, lo que facilita el manejo de grandes volúmenes de datos de forma eficiente.

4.2 Proceso de entrenamiento y prueba de modelos

En esta sección se detallan las decisiones técnicas clave que se tomaron durante la implementación de los modelos de imputación de genotipos faltantes. Estas decisiones abarcan desde la configuración de los hiperparámetros hasta la selección de enfoques específicos de modelado, teniendo en cuenta las características de los datos de *SNPs* y los objetivos de imputación.

4.2.1 KNN

El modelo de (*KNN*) fue implementado utilizando la clase `KNNImputer` de la librería `Scikit-learn`¹, con el propósito de imputar los valores faltantes en los datos. El proceso de entrenamiento y prueba se estructuró en las siguientes etapas:

- **Configuración del modelo:** Como KNN es un método basado en instancias, no requiere un proceso de entrenamiento explícito. En cada caso, el imputador `KNNImputer` fue configurado para diferentes valores de k (número de vecinos), ajustando las relaciones entre las muestras utilizando los datos completos de entre-

¹<https://scikit-learn.org/1.4/modules/generated/sklearn.impute.KNNImputer.html>sklearn.impute.KNNImputer

namiento. Esto permitió definir las similitudes entre las muestras para aplicar la imputación de los valores faltantes en los conjuntos de prueba.

- **Ejecución del modelo en los conjuntos de prueba:** La imputación de los valores faltantes se realizó en los distintos conjuntos de prueba, combinando varios valores de k y niveles de pérdida de datos. El proceso consistió en calcular la similitud entre las muestras basándose en las características conocidas y en los k vecinos más cercanos, imputando los valores ausentes de acuerdo con estas relaciones.
- **Almacenamiento de los conjuntos imputados:** Tras completar la imputación en cada conjunto de prueba, los resultados se almacenaron para cada valor de k y cada nivel de pérdida de datos. Estos conjuntos imputados se reservaron para su posterior evaluación comparativa con los valores originales del conjunto de prueba, a fin de cuantificar la precisión de la imputación realizada.

El proceso de imputación fue ejecutado de manera iterativa para cada combinación de k y nivel de pérdida de datos, generando múltiples conjuntos de prueba con imputaciones de valores faltantes. Gracias a la naturaleza basada en instancias de KNN, no fue necesario un proceso intensivo de entrenamiento, sino que el enfoque se basó en la definición de similitudes entre las muestras conocidas.

4.2.2 RF

El modelo de *RF* se implementó utilizando la clase *RandomForestClassifier* de *Scikit-learn*². Se realizaron varias pruebas para optimizar el rendimiento del modelo, siendo las decisiones clave:

- **Configuración del modelo:** El modelo fue configurado con un número de **200 árboles** (*n_estimators*) y una **profundidad máxima de 6 niveles** (*max_depth*), parámetros seleccionados para equilibrar precisión y eficiencia computacional, evitando el sobreajuste. Además, se utilizó un **valor de semilla aleatoria de 42** (*random_state*) para garantizar la reproducibilidad de los resultados, y la opción **n_jobs = -1** para aprovechar todos los núcleos disponibles del sistema en el proceso de entrenamiento.

²<https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.RandomForestClassifier.html>sklearn-ensemble-randomforestclassifier

- **Entrenamiento del modelo:** El conjunto de datos completo de entrenamiento, que no contenía valores faltantes, fue utilizado para entrenar el modelo de *RF*. El entrenamiento se llevó a cabo utilizando todas las posiciones de SNPs como predictores para aprender las relaciones entre las diferentes variantes genéticas. El modelo ajustó múltiples árboles de decisión para maximizar la precisión en la imputación de los valores faltantes.
- **Prueba del modelo:** Una vez entrenado el modelo de *RF*, este fue aplicado a los conjuntos de prueba con pérdida de datos. El modelo utilizó las posiciones de los SNPs conocidas para predecir los valores faltantes en las posiciones correspondientes. Este enfoque permitió imputar los datos en función de las relaciones aprendidas en el conjunto de entrenamiento.
- **Almacenamiento de los datos imputados:** Tras la imputación de los valores faltantes, los conjuntos de prueba imputados fueron almacenados para su posterior análisis y evaluación en comparación con los valores originales, lo que permitirá evaluar la precisión de las imputaciones generadas por el modelo.

El proceso de imputación se realizó iterativamente para cada conjunto de prueba con diferentes niveles de pérdida de datos, utilizando un único modelo de *RF* entrenado previamente en todas las posiciones de SNPs.

4.2.3 RF por columna

En este enfoque, se entrenó un modelo de *RF* de manera independiente para cada columna del conjunto de datos. Esta estrategia permite que cada modelo se especialice en la imputación de una columna específica, utilizando las demás columnas como predictores. El proceso general de entrenamiento y prueba sigue la estructura del modelo único de *RF*, pero presenta las siguientes diferencias clave:

- **Imputación escalonada:** Los valores imputados en una columna se reutilizan como predictores para las siguientes columnas. Esto pretende mejorar la calidad de las predicciones, ya que el modelo incorpora progresivamente la información generada en las imputaciones previas, lo que aporta información del conjunto de características disponibles para las siguientes imputaciones.

- **Optimización individual:** Cada modelo se ajusta de manera independiente para cada columna de SNPs, lo que pretende capturar de manera más precisa las relaciones particulares entre los SNPs, ya que se adapta específicamente a las características de cada columna en lugar de tratar el conjunto de datos de manera global.

4.2.4 XGBoost

4.2.5 Entrenamiento y prueba del modelo XGBoost

El modelo de *XGBoost* fue implementado utilizando la clase `XGBClassifier` de la librería `xgboost`, con el objetivo de realizar la imputación de valores faltantes en los datos de SNPs. A continuación, se describe el proceso de entrenamiento y prueba, detallando los parámetros seleccionados y las decisiones técnicas clave:

- **Configuración del modelo:** El modelo de `XGBClassifier` fue configurado con los siguientes parámetros: `n_estimators = 200` (200 árboles), `max_depth = 6` (profundidad máxima de los árboles), `random_state = 42` para garantizar la reproducibilidad de los resultados, y `n_jobs = -1` para aprovechar todos los núcleos disponibles del sistema y optimizar el tiempo de ejecución.
- **Entrenamiento del modelo:** El entrenamiento del modelo se realizó utilizando el conjunto de datos de entrenamiento completo, que no contenía valores faltantes. En este caso, las etiquetas originales del conjunto de datos, codificadas como [1, 2], se ajustaron a [0, 1] para el proceso de clasificación binaria requerido por el modelo. El algoritmo de *XGBoost* entrenó múltiples árboles de decisión, capturando las relaciones entre las características genotípicas para predecir los valores faltantes en las columnas de SNPs durante la fase de prueba.
- **Imputación de valores faltantes:** Para la imputación, se utilizó un proceso de predicción fila por fila. En cada fila con valores faltantes, el modelo aprovechó las características conocidas (no faltantes) para predecir las características faltantes (NaN). Este proceso permitió reconstruir las columnas de SNPs incompletas utilizando la información que el modelo había aprendido durante el entrenamiento. La predicción fue realizada utilizando las características conocidas de cada fila, y los valores imputados se almacenaron en el conjunto de prueba.

- **Evaluación de las imputaciones:** Una vez completada la imputación, se calcularon diversas métricas para evaluar la calidad de la imputación. El conjunto imputado fue comparado con el conjunto de prueba original, permitiendo calcular métricas como la precisión (*accuracy*), *F1-Score*, *precisión* y *recall*. Estas métricas se calcularon utilizando las predicciones generadas por el modelo para los valores faltantes en cada fila del conjunto de prueba.
- **Generación de conjuntos de prueba con datos faltantes:** Se generaron subconjuntos del conjunto de prueba original con distintos porcentajes de valores faltantes, en niveles del 10%, 20%, 30% y 40%. Estos conjuntos se utilizaron para simular distintos grados de incompletitud en los datos, y el modelo de *XGBoost* fue evaluado en cada uno de ellos.
- **Almacenamiento de los datos imputados:** Los datos imputados por el modelo fueron almacenados en conjuntos separados para cada nivel de datos faltantes. Estos conjuntos fueron generados para permitir una evaluación exhaustiva de la imputación realizada por el modelo de *XGBoost*, facilitando el análisis comparativo de las predicciones frente a los valores reales en los diferentes escenarios de incompletitud.

El proceso de imputación y prueba del modelo *XGBoost* se ejecutó iterativamente para cada uno de los niveles de datos faltantes (10%, 20%, 30%, 40%), utilizando el modelo entrenado para realizar la imputación de los valores ausentes y evaluar la calidad de las predicciones en diferentes contextos de pérdida de información.

4.2.6 Sparse Convolutional Denoising Autoencoder

El modelo *SCDA* (Chen and Shi, 2019) combina autoencoders convolucionales con regularización dispersa (L1) para manejar datos de alta dimensionalidad y abordar la imputación de valores faltantes en datos genéticos. Las principales características del modelo *SCDA* son las siguientes:

- **Autoencoder convolucional:** *SCDA* utiliza capas convolucionales tanto en el codificador como en el decodificador. Las capas convolucionales permiten al modelo capturar patrones locales y relaciones espaciales en los datos genéticos..

- **Denoising con generación dinámica de datos faltantes:** SCDA emplea una técnica de *denoising*, en la cual se introduce de manera dinámica ruido en los datos de entrada durante el entrenamiento, eliminando aleatoriamente algunos valores genotípicos. Esto se logra mediante un generador de datos (**DataGenerator**), que introduce un 10% de valores faltantes en cada batch de entrenamiento. Este enfoque permite al modelo aprender a reconstruir los datos originales a partir de versiones incompletas o ruidosas, simulando diferentes configuraciones de pérdida de datos en cada iteración. Al entrenarse con estos patrones dinámicos de incompletitud, el modelo mejora su capacidad para generalizar y realizar imputaciones precisas en escenarios reales de datos faltantes.
- **Regularización dispersa (L1):** SCDA aplica regularización L1 para penalizar los pesos grandes en el modelo, lo que obliga a que muchos pesos sean cercanos a cero. Este mecanismo hace que el modelo utilice solo las características más relevantes para la imputación, además de evitar el sobreajuste.
- **Imputación genotípica:** El objetivo de SCDA es imputar los valores faltantes en los datos genotípicos. Después de entrenar el modelo con datos incompletos generados durante el proceso de denoising, SCDA puede predecir los valores genotípicos faltantes en un conjunto de datos de prueba, utilizando la información conocida de los SNPs cercanos para realizar la imputación.

Aunque SCDA ha demostrado ser efectivo en la imputación de valores faltantes en datos genotípicos, existe la posibilidad de mejorar aún más su rendimiento. Con el fin de incrementar la precisión y la capacidad del modelo para centrarse en las características más relevantes, se propone la incorporación de un mecanismo de atención. El siguiente apartado detalla la implementación de *Autoencoders con capa de atención y denoising*, un modelo que extiende el enfoque de SCDA.

4.2.7 Sparse Convolutional Attention Denoising Autoencoder

El modelo al que llamamos Sparse Convolutional Attention Denoising Autoencoder (SCADA) se ha implementado como una versión avanzada del modelo SCDA, incorporando un mecanismo de atención que mejora la capacidad del autoencoder para enfocarse en las características más relevantes durante el proceso de imputación de valores faltantes. A continuación, se detallan las etapas clave del proceso de entrenamiento y prueba:

- **Arquitectura del modelo:** El modelo incluye una secuencia de capas convolucionales en el codificador, similares a las utilizadas en SCDA, que permiten capturar patrones espaciales en los datos genotípicos. Sin embargo, se introduce un **mecanismo de atención**, que ajusta dinámicamente los pesos de las características más importantes lo que permite al modelo priorizar las características más relevantes para la imputación. Posteriormente, el decodificador reconstruye los datos faltantes utilizando técnicas como *UpSampling* para restaurar las dimensiones de las características originales.
- **Mejoras respecto a SCDA:** A diferencia de SCDA, este modelo incorpora un mecanismo de atención que optimiza la imputación al centrarse en las características más influyentes en los datos. Esto resulta en imputaciones más precisas, especialmente en situaciones donde ciertos SNPs tienen un impacto desproporcionado en la información genética faltante. Además, se han ajustado los hiperparámetros del modelo para incluir regularización
- **Configuración del modelo:** Los principales hiperparámetros seleccionados para el entrenamiento fueron:
 - **batch_size = 32:** Tamaño del batch para definir cuántas muestras se procesan antes de actualizar los pesos.
 - **lr = 1e-3:** Tasa de aprendizaje para controlar los cambios en los pesos durante el entrenamiento.
 - **epochs = 10:** Número de épocas que el modelo verá el conjunto de datos completo.
 - **kr = 1e-4:** Regularización L1 para penalizar los pesos grandes y evitar el sobreajuste.
 - **drop_prec = 0.25:** *Dropout* para desactivar neuronas y reducir el riesgo de sobreajuste.
- **Entrenamiento y generación dinámica de datos faltantes:** Al igual que en SCDA, el modelo se entrenó utilizando el optimizador **Adam** y *categorical crossentropy* como función de pérdida. Se implementó un mecanismo de *early stopping* para evitar el sobreajuste si la pérdida de validación no mejora. Durante el entrenamiento, se

empleó un `DataGenerator` que introdujo dinámicamente valores faltantes en cada batch, simulando distintos niveles de incompletitud en los datos.

- **Evaluación, imputación y predicción:** La evaluación del modelo se realizó utilizando conjuntos de prueba con valores faltantes predefinidos en porcentajes variables. Las predicciones de los valores faltantes se realizaron de manera eficiente en lotes mediante el método `predict` de la clase *Model* de Keras³. Las características conocidas fueron utilizadas para predecir las faltantes, y posteriormente se compararon las etiquetas predichas con las reales, calculando la precisión promedio.

³https://www.tensorflow.org/api_docs/python/tf/keras/Model

5. Resultados

En este capítulo se lleva a cabo una presentación y comparativa de resultados de los modelos. La fase de resultados se estructura en las siguientes etapas: en la Sección 5.1, se explican las métricas principales utilizadas para evaluar los modelos. Consecutivamente, en la Sección 5.2, se lleva a cabo la presentación de los resultados individuales de cada modelo. Finalmente, en la Sección 5.3, se desarrolla una comparativa general entre los modelos de ML tradicional por un lado y los de DL por el otro.

5.1 Métricas Utilizadas para la Evaluación de Modelos

En la evaluación de los modelos utilizados para la imputación de datos, se han empleado varias métricas estándar en la literatura del ML. A continuación se presenta una descripción técnica de cada una de estas métricas, junto con las fórmulas empleadas.

5.1.1 *Medium Square Error* (MSE)

El Error Cuadrático Medio (en inglés, Medium Square Error (MSE)) mide la media de los cuadrados de los errores, es decir, las diferencias entre los valores imputados y los valores reales. Cuantifica el error promedio de un estimador y es sensible a los valores atípicos debido a la operación de elevación al cuadrado.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

donde y_i es el valor verdadero, \hat{y}_i es el valor imputado o predicho, y n es el número de muestras.

5.1.2 *Medium Absolut Error* (MAE)

El Error Absoluto Medio (en inglés, Medium Absolut Error (MSE)) es la media de los valores absolutos de las diferencias entre los valores reales y los imputados. A diferencia del MSE, el MAE no penaliza en exceso los valores atípicos, ya que no utiliza la operación de elevación al cuadrado.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

donde y_i es el valor verdadero, \hat{y}_i es el valor imputado o predicho, y n es el número de muestras.

5.1.3 Coeficiente de Determinación (R^2)

El coeficiente R^2 mide la proporción de la varianza total de la variable dependiente que es explicada por el modelo. Tiene un valor máximo de 1, lo que indica una predicción perfecta, y puede tomar valores negativos si el modelo tiene un rendimiento peor que un modelo trivial que predice la media.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

donde y_i es el valor verdadero, \hat{y}_i es el valor imputado o predicho, y \bar{y} es la media de los valores reales.

5.1.4 Accuracy

El Accuracy mide la proporción de predicciones correctas sobre el total de muestras. En problemas de clasificación binaria (como se puede considerar a la imputación de dos variables genéticas (1 ó 2) por probabilidades), se calcula de la siguiente manera:

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

donde TP es el número de verdaderos positivos, TN es el número de verdaderos negativos, FP es el número de falsos positivos, y FN es el número de falsos negativos.

5.1.5 F1-Score

El F1-Score es la media armónica entre la precisión y el recall. Se utiliza para encontrar un balance entre estos dos aspectos, especialmente útil cuando hay clases desbalanceadas.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}}$$

donde la precisión mide la proporción de predicciones positivas correctas, y el recall mide la proporción de verdaderos positivos sobre todos los casos que realmente son positivos.

5.1.6 Precisión

La precisión mide la proporción de predicciones positivas correctas en relación con todas las predicciones positivas hechas por el modelo:

$$\text{Precisión} = \frac{TP}{TP + FP}$$

donde TP es el número de verdaderos positivos y FP es el número de falsos positivos.

5.1.7 Recall (Sensibilidad)

El recall mide la proporción de verdaderos positivos sobre todos los casos que realmente son positivos:

$$\text{Recall} = \frac{TP}{TP + FN}$$

donde TP es el número de verdaderos positivos y FN es el número de falsos negativos.

5.2 Resultados individuales

En esta sección se presentan los resultados individuales de cada una de las arquitecturas y para todos los casos de dimensionalidad de datos probados.

5.2.1 KNN

El modelo KNN fue evaluado con diferentes configuraciones de número de vecinos y niveles de missingness. Se observaron tendencias consistentes en las métricas MSE, MAE y R2 en función del número de vecinos y el porcentaje de datos faltantes. A continuación, se presentan los resultados obtenidos para cada configuración:

Resultados de Métricas por Número de Vecinos y Missingness

Los resultados muestran cómo varían las métricas de MSE, MAE y R2 con el aumento del número de vecinos y con diferentes niveles de missingness en los datos. Se observa que el aumento en el número de vecinos generalmente mejora las métricas, indicando una mejor imputación con más vecinos. Además, a medida que aumenta el porcentaje de missingness, las métricas tienden a deteriorarse, lo que es esperado dado que hay menos información disponible para la imputación (ver tabla 5.1).

| Número de Vecinos | Missingness | MSE | MAE | R2 |
|-------------------|-------------|--------|--------|--------|
| 3 | 10% | 0.0154 | 0.0266 | 0.9384 |
| 3 | 20% | 0.0310 | 0.0538 | 0.8758 |
| 3 | 30% | 0.0478 | 0.0823 | 0.8087 |
| 3 | 40% | 0.0648 | 0.1117 | 0.7410 |
| 5 | 10% | 0.0136 | 0.0275 | 0.9454 |
| 5 | 20% | 0.0277 | 0.0558 | 0.8893 |
| 5 | 30% | 0.0425 | 0.0851 | 0.8301 |
| 5 | 40% | 0.0575 | 0.1156 | 0.7698 |
| 7 | 10% | 0.0130 | 0.0282 | 0.9480 |
| 7 | 20% | 0.0264 | 0.0572 | 0.8943 |
| 7 | 30% | 0.0405 | 0.0872 | 0.8379 |
| 7 | 40% | 0.0550 | 0.1184 | 0.7799 |
| 10 | 10% | 0.0127 | 0.0290 | 0.9493 |
| 10 | 20% | 0.0257 | 0.0588 | 0.8972 |
| 10 | 30% | 0.0394 | 0.0897 | 0.8423 |
| 10 | 40% | 0.0536 | 0.1219 | 0.7856 |

Tabla 5.1: Tabla de métricas MSE, MAE y R2 para diferentes configuraciones de número de vecinos y niveles de missingness en el modelo KNN

Visualización de Resultados

Se incluyen gráficos que ilustran las tendencias de las métricas MSE, MAE y R2 con respecto al número de vecinos y los niveles de missingness. Estos gráficos (ver ??) facilitan la visualización del impacto del número de vecinos en la precisión de la imputación.??.

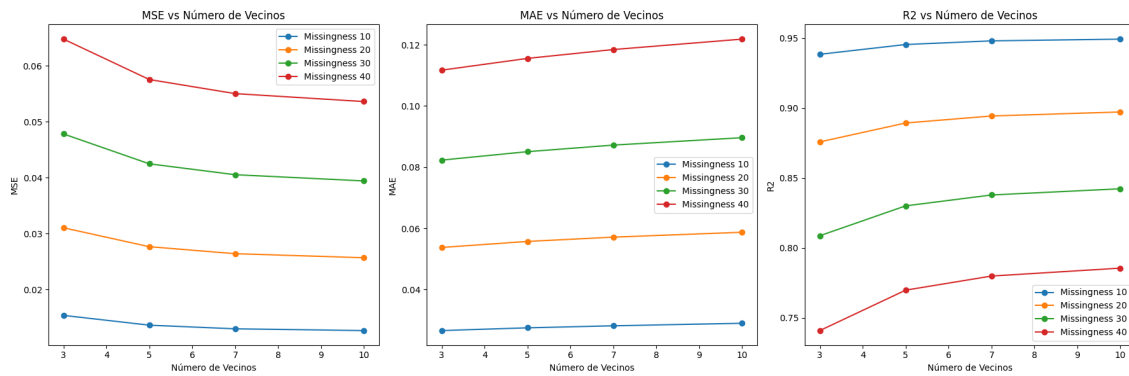


Figura 5.1: Gráficos de MSE, MAE y R2 frente al número de vecinos para diferentes niveles de missingness.(Fuente: Elaboración Propia)

5.2.2 RF

El modelo RF se evalúa bajo dos configuraciones principales: un modelo único para todas las columnas y un modelo separado para cada columna. A continuación, se presentan los resultados obtenidos en ambas configuraciones.

RF con modelo único

En esta configuración, se entrenó un único modelo de RF para predecir los valores faltantes en todas las columnas del conjunto de datos. Los resultados de las métricas de rendimiento se muestran a continuación:

| Missingness | Accuracy | F1-Score | Precision | Recall |
|-------------|----------|----------|-----------|--------|
| 10% | 0.5364 | 0.5364 | 0.5364 | 0.5364 |
| 20% | 0.5420 | 0.5415 | 0.5424 | 0.5422 |
| 30% | 0.5298 | 0.5277 | 0.5304 | 0.5299 |
| 40% | 0.5156 | 0.5095 | 0.5172 | 0.5163 |

Tabla 5.2: Métricas de rendimiento del modelo de RF entrenado como un solo modelo para todas las columnas.

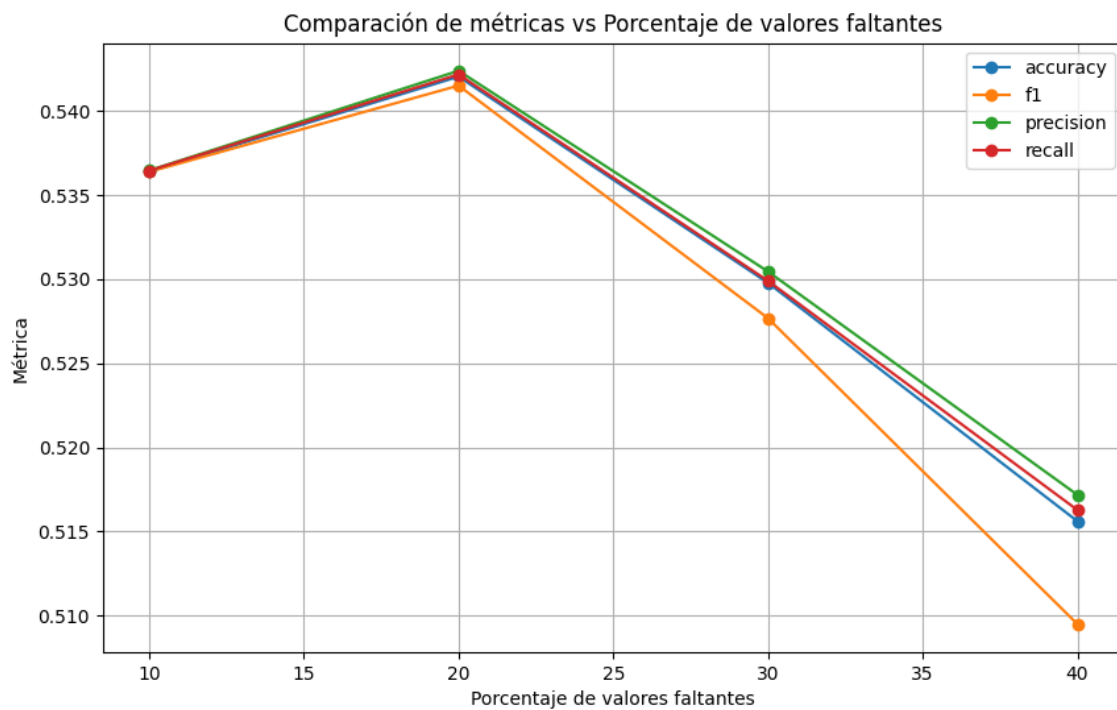


Figura 5.2: Gráficos comparativos de las métricas Accuracy, F1-Score, Precision y Recall frente al porcentaje de valores faltantes para el modelo de RF único (Fuente: Elaboración Propia).

RF con modelo por columna

En esta configuración, se entrenó un modelo de RF específico para cada columna, lo que permite que cada modelo se especialice en la imputación de una característica utilizando las demás como predictores. Los resultados de las métricas por columna se incluyen en la tabla y se grafican en la imagen 5.3:

| Missingness | Accuracy | F1-Score | Precision | Recall |
|-------------|----------|----------|-----------|--------|
| 10% | 0.9758 | 0.9758 | 0.9758 | 0.9758 |
| 20% | 0.9729 | 0.9729 | 0.9729 | 0.9729 |
| 30% | 0.9697 | 0.9697 | 0.9697 | 0.9697 |
| 40% | 0.9668 | 0.9668 | 0.9668 | 0.9668 |

Tabla 5.3: Métricas de rendimiento de los modelos de RF entrenados por columna.

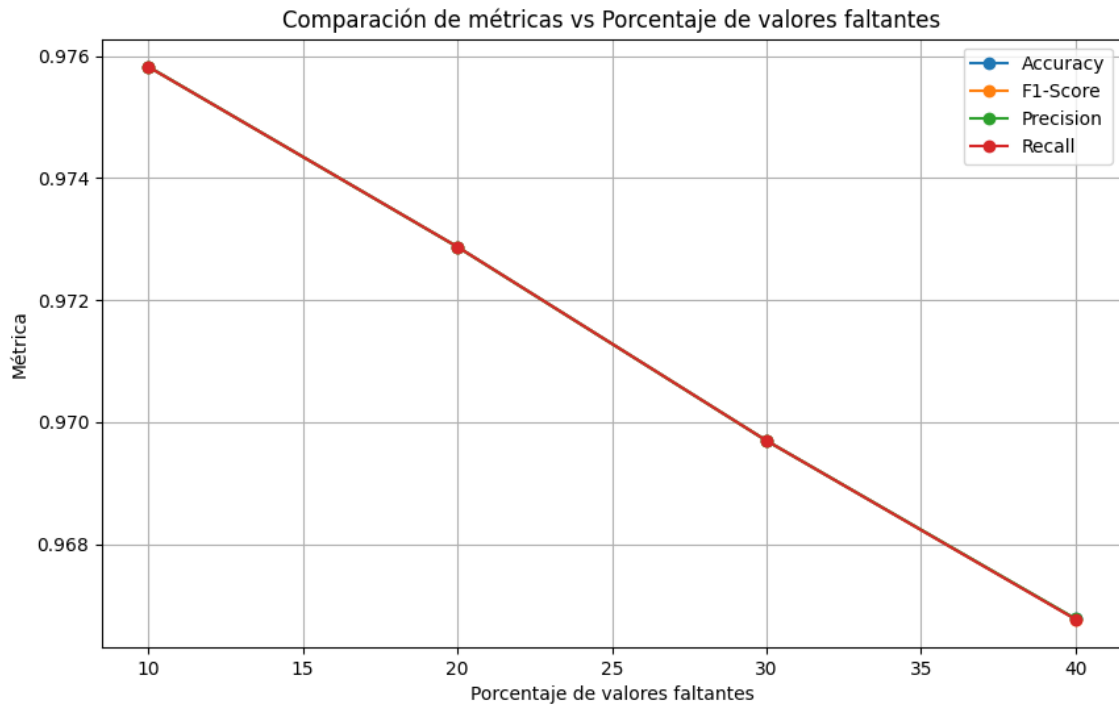


Figura 5.3: Gráficos comparativos de las métricas Accuracy, F1-Score, Precision y Recall frente al porcentaje de valores faltantes para el modelo de RF por columna (Fuente: Elaboración Propia).

5.2.3 Análisis de los resultados de RF

El modelo único muestra un rendimiento inferior debido a la falta de especialización. Al imputar todas las columnas a la vez, el modelo no puede capturar adecuadamente las relaciones entre los SNPs, lo que resulta en una disminución de las métricas (Accuracy, F1-Score, Precision y Recall) a medida que aumenta el porcentaje de valores faltantes. La variabilidad entre las columnas hace que este enfoque sea menos efectivo

El modelo por columna muestra métricas altas debido a la especialización, pero este enfoque no capta las relaciones entre columnas, ya que cada modelo se enfoca en imputar una única columna de manera aislada. Aunque este tipo de imputación puede no captar interacciones genéticas más complejas como lo hacen los métodos basados en haplotipos o modelos mixtos, el enfoque por columna puede ser útil en sistemas donde se procesan repetidamente los mismos genes, a pesar de estar sobreajustado. Algunos ejemplos específicos donde el enfoque por columna puede ser útil incluyen:

- **Estudios de genotipado rutinario:** En proyectos donde se imputan valores faltantes repetidamente en un conjunto fijo de genes o SNPs, y donde los patrones de pérdida de datos son consistentes, un modelo por columna puede ofrecer una imputación precisa ya que, al especializarse en una única columna, el modelo aprende a reconstruir con precisión los valores faltantes basándose en los patrones observados repetidamente en las mismas posiciones genómicas.

- **Diagnóstico genético:** En situaciones donde se analizan los mismos paneles genéticos para identificar mutaciones específicas en genes conocidos (por ejemplo, en el diagnóstico de enfermedades genéticas), un modelo por columna puede ajustarse bien y ofrecer imputaciones fiables, dado que el espacio de búsqueda es más reducido y está controlado (Consortium, 2005).

- **Estudios de cohortes homogéneas:** Si los datos genotípicos provienen de una cohorte genéticamente homogénea, como en estudios de poblaciones cerradas, los patrones de variación genética pueden ser similares entre individuos. En este caso, un modelo por columna puede aprovechar las dependencias dentro de una población específica para generar imputaciones precisas sin la necesidad de captar interacciones más complejas entre SNPs (Browning and Browning, 2009; Jiang and Reif, 2015).

5.2.4 XGBoost

El modelo XGBoost se evalúa para determinar su eficacia en la imputación de datos faltantes. A continuación, se presentan los resultados obtenidos con este modelo.

Resultados de XGBoost

En este análisis, se utilizó un modelo de XGBoost para predecir los valores faltantes en todas las columnas del conjunto de datos en cuatro subsets con distinto porcentaje de pérdida. Los resultados de las métricas de rendimiento se muestran en la tabla siguiente:

| Missingness | Accuracy | F1-Score | Precision | Recall |
|-------------|----------|----------|-----------|--------|
| 10% | 0.5275 | 0.5264 | 0.5281 | 0.5278 |
| 20% | 0.5260 | 0.5215 | 0.5275 | 0.5264 |
| 30% | 0.5197 | 0.5087 | 0.5220 | 0.5200 |
| 40% | 0.5141 | 0.4943 | 0.5182 | 0.5152 |

Tabla 5.4: Métricas de rendimiento del modelo de XGBoost para la imputación de valores faltantes en sets de datos con distinto porcentaje de pérdida.

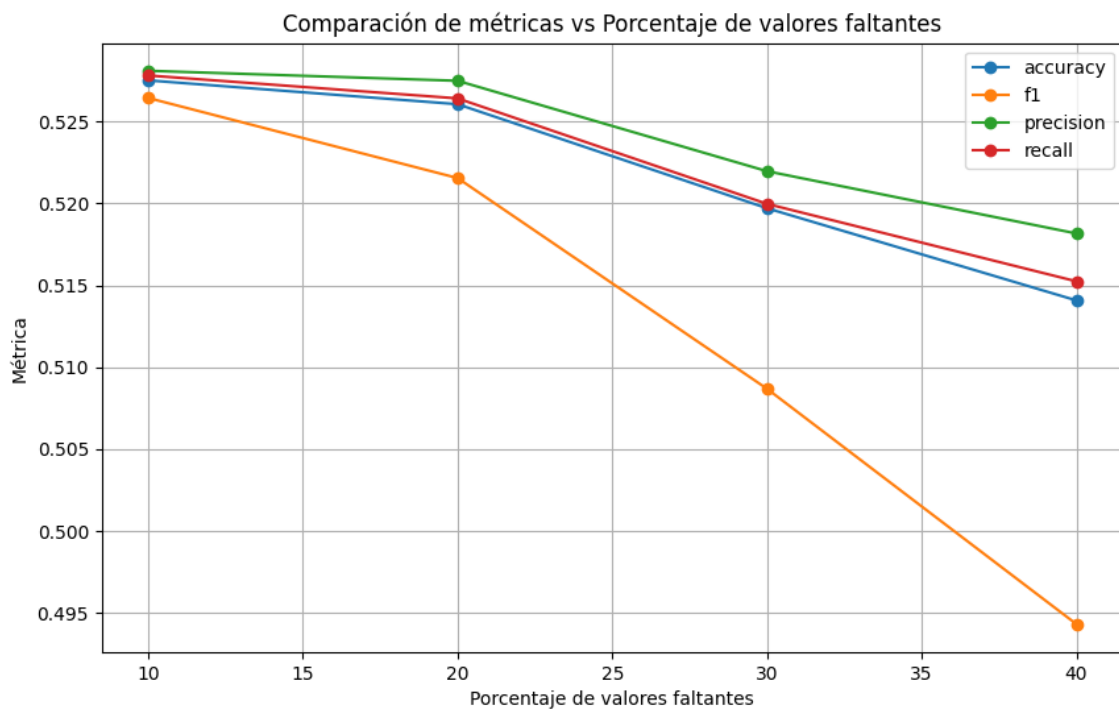


Figura 5.4: Gráficos comparativos de las métricas Accuracy, F1-Score, Precision y Recall frente al porcentaje de valores faltantes para el modelo de XGBoost (Fuente: Elaboración Propia)

5.2.5 Análisis de los resultados de XGBoost

XGBoost muestra unas métricas alrededor del 0.5, lo cual se considera bajo además una caída en el rendimiento de las métricas a medida que aumenta el porcentaje de valores faltantes. Esto se debe a las siguientes limitaciones arquitectónicas:

- **Divisiones binarias:** XGBoost realiza divisiones rígidas en cada nodo, lo que di-

ficulta manejar valores faltantes de forma efectiva, ya que las decisiones sobre estos son heurísticas y no modelan correctamente la ausencia de datos.

- **Falta de relaciones enter columnas:** Los árboles procesan características individualmente, lo que impide capturar relaciones entre columnas que podrían mejorar la imputación.
- **Segmentación local:** Al dividir los datos en subconjuntos, XGBoost pierde una visión global, lo que afecta la precisión de la imputación al no considerar todo el conjunto de datos.

5.2.6 SCDA

El método SCDA se evaluó mediante la comparación del desempeño del modelo entrenado con el conjunto de datos completo y con subconjuntos de datos. A continuación se presentan los resultados obtenidos en ambas configuraciones, demostrando cómo la precisión promedio se ve afectada por el porcentaje de pérdida de datos.

Entrenado con conjunto de datos completo

Se reproduce la ejecución del modelo original con los datos originales obteniendo los mismos resultados. El modelo SCDA, entrenado con el conjunto de datos completo, muestra una excelente capacidad de generalización, como se evidencia por las bajas pérdidas y las altas precisiones alcanzadas tanto en el entrenamiento como en la validación (ver tabla 5.5).

| Métrica | Valor |
|----------------------------------|--------|
| Pérdida de Entrenamiento Final | 0.0164 |
| Pérdida de Validación Final | 0.0136 |
| Precisión de Entrenamiento Final | 0.9985 |
| Precisión de Validación Final | 0.9996 |

Tabla 5.5: Métricas finales de entrenamiento y validación para SCDA con el conjunto completo de datos.

La gráfica de pérdida muestra una rápida disminución tanto en entrenamiento como en validación durante las primeras épocas, seguida de una estabilización a niveles muy bajos (ver figura 5.5). Esto indica que el modelo rápidamente alcanza una buena adaptación

a los datos sin signos evidentes de sobreajuste, ya que la pérdida de validación sigue de cerca a la pérdida de entrenamiento y se mantiene a la par en términos de magnitud.

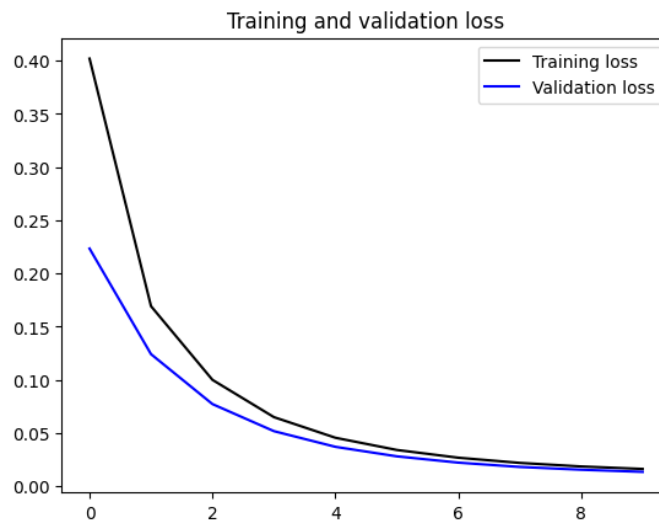


Figura 5.5: Pérdida de entrenamiento y validación para el conjunto completo en SCDA (Fuente: Elaboración Propia).

En cuanto a la precisión, la gráfica revela que el modelo alcanza y mantiene niveles excepcionalmente altos de precisión tanto en entrenamiento como en validación (ver figura 5.6) . La precisión en entrenamiento es casi perfecta, alcanzando valores cercanos al 100%, y la precisión de validación es comparativamente alta, lo que sugiere que el modelo no solo aprende eficazmente los patrones de los datos de entrenamiento, sino que también generaliza bien a nuevos datos.

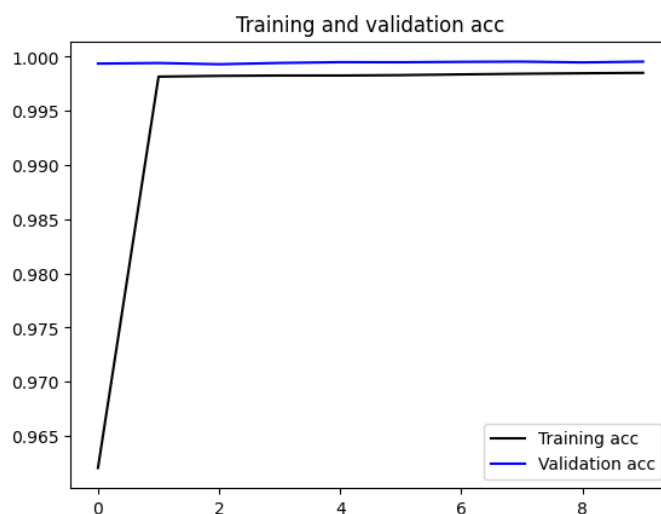


Figura 5.6: Precisión de entrenamiento y validación para el conjunto completo en SCDA.(Fuente: Elaboración Propia)

El modelo muestra una gran capacidad para mantener altos niveles de precisión incluso cuando se enfrenta a una pérdida creciente de datos. Esto se ilustra claramente en la tabla 5.6 y figura 5.10, que presentan la precisión promedio obtenida a diferentes niveles de pérdida de datos durante la fase de prueba, alcanzando valores muy cercanos al 1.

| Porcentaje de pérdida | Precisión promedio |
|-----------------------|--------------------|
| 10.0% | 0.9979 |
| 20.0% | 0.9978 |
| 30.0% | 0.9976 |
| 40.0% | 0.9974 |

Tabla 5.6: Precisión promedio por porcentaje de pérdida de datos entrenado con el conjunto completo.

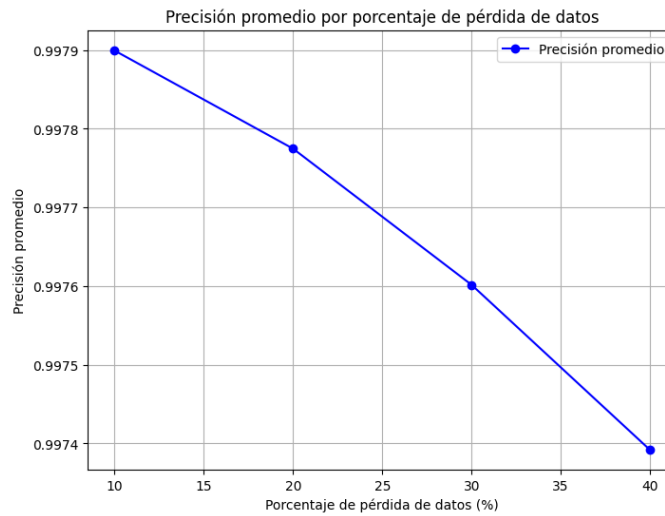


Figura 5.7: Precisión promedio por porcentaje de pérdida de datos entrenado con conjunto completo de datos (Fuente: Elaboración Propia)

Entrenado con subconjunto de datos

El entrenamiento de SCADA con subconjuntos de datos, al contar con menos ejemplos para aprender, muestra un patrón distinto en comparación con el conjunto completo. Los resultados indican una convergencia más gradual y una variabilidad más notoria sobre todo en la precisión y no tanto en la pérdida (ver tabla 5.7).

| Métrica | Valor |
|----------------------------------|--------|
| Pérdida de Entrenamiento Final | 0.1956 |
| Pérdida de Validación Final | 0.1710 |
| Precisión de Entrenamiento Final | 0.9326 |
| Precisión de Validación Final | 0.9467 |

Tabla 5.7: Métricas finales de entrenamiento y validación para SCDA estranado con subconjunto de datos.

Aunque la pérdida disminuye notablemente desde valores iniciales altos, se estabiliza a un nivel ligeramente superior al observado en el conjunto completo, aunque siguiendo un patrón muy parecido en cuanto a convergencia (ver figura 5.8).

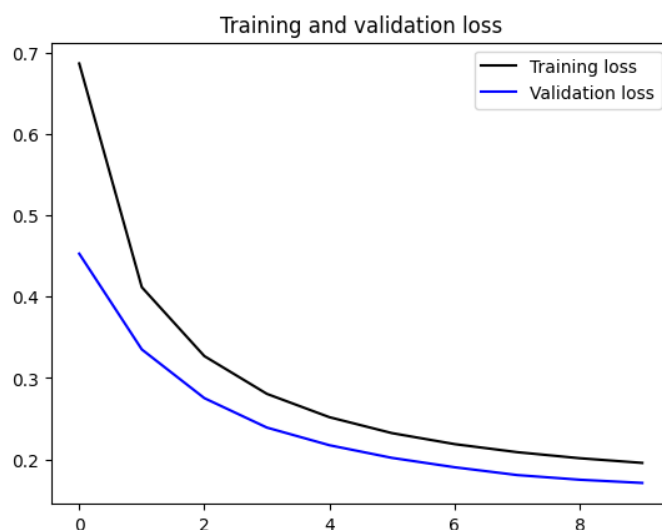


Figura 5.8: Pérdida de entrenamiento y validación en SCDA (Fuente: Elaboración propia)

En este caso, la precisión también aumenta rápidamente, pero a un nivel más bajo, alcanzando y manteniendo valores por encima del 90%, además de ser menos estable y presentar mayor diferencia entre entrenamiento y validación, presentando más sobreajuste que el modelo entrenado con el set completo de datos, lo cual refleja el impacto de una menor cantidad de datos (ver figura 5.9).

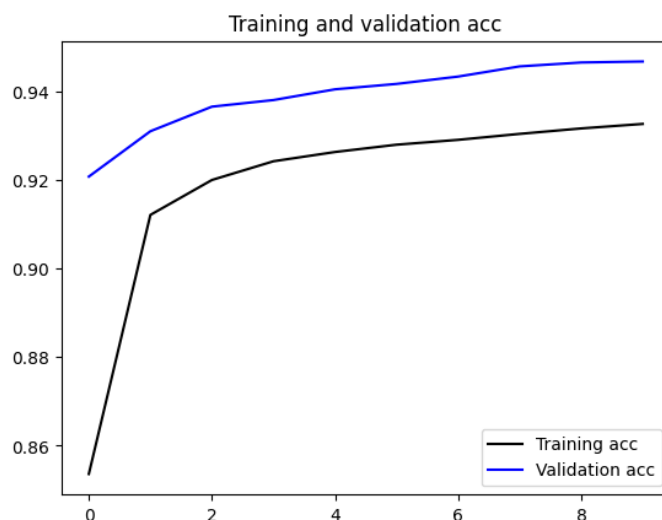


Figura 5.9: Precisión de entrenamiento y validación para subconjuntos en SCDA, reflejando una alta capacidad de clasificación a pesar de la reducción de datos (Fuente: Elaboración Propia).

Así, vemos reflejada la calidad del entrenamiento en el test. Aunque vemos resultados

buenos, por encima del 0.8 (ver tabla 5.8 y figura 5.10) , disminuyen notablemente respecto a los del test en el modelo entrenado con el set completo de datos.

| Porcentaje de pérdida | Precisión promedio |
|-----------------------|--------------------|
| 10.0% | 0.8407 |
| 20.0% | 0.8374 |
| 30.0% | 0.8336 |
| 40.0% | 0.8286 |

Tabla 5.8: Precisión promedio por porcentaje de pérdida de datos entrenado con subconjuntos de datos.

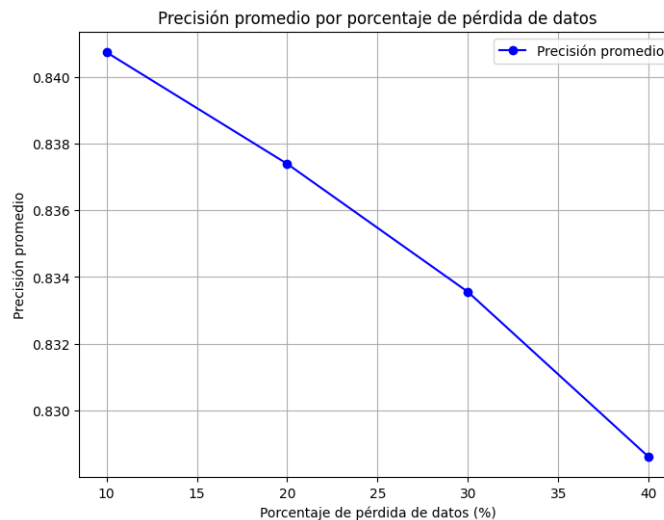


Figura 5.10: Precisión promedio por porcentaje de pérdida de datos entrenado con subconjunto de datos(Fuente: Elaboración Propia).

Estos resultados destacan cómo la reducción en el tamaño del conjunto de datos afecta de manera significativa la capacidad del modelo para mantener su precisión, especialmente a medida que aumenta el porcentaje de pérdida de datos.

5.2.7 Análisis de los Resultados de SCDA

Los resultados obtenidos del modelo SCDA, un autoencoder, muestran un rendimiento notablemente alto cuando se entrena con el conjunto completo de datos. La precisión promedio es cercana a 1, incluso con hasta un 40% de pérdida de datos, lo que indica la capacidad del modelo para manejar la falta de datos de manera eficiente. Esta resiliencia

puede explicarse por las características inherentes de los autoencoders, que, a diferencia de los modelos tradicionales de ML, están diseñados para aprender representaciones latentes compactas de los datos y son menos dependientes de cada punto de entrada individual.

Rendimiento con el Conjunto de Datos Completo

El modelo SCDA entrenado con el conjunto completo de datos alcanza precisiones superiores al 99.74% incluso con un 40% de pérdida. Los autoencoders pueden inferir patrones más profundos y estructuras ocultas en los datos, lo que les permite completar o imputar valores faltantes de manera más efectiva que los modelos de aprendizaje supervisado tradicionales, que dependen de la correlación explícita entre características. Esta capacidad de aprendizaje no supervisado le permite al modelo hacer predicciones precisas incluso con una cantidad significativa de valores ausentes.

Rendimiento con Subconjuntos de Datos

Sin embargo, cuando el modelo es entrenado con subconjuntos de datos más pequeños, la precisión promedio disminuye drásticamente, cayendo por debajo del 83% cuando hay un 40% de pérdida de datos. Esta caída en el rendimiento puede atribuirse a varios factores clave:

- **Reducción de la información disponible:** Aunque los autoencoders están diseñados para manejar bien datos incompletos, su capacidad de generalización depende de la cantidad de datos representativos disponibles para entrenar. Los subconjuntos reducidos proveen menos información para aprender las representaciones latentes de manera efectiva.
- **Pérdida de patrones clave:** En los subconjuntos más pequeños, es probable que falten algunas de las relaciones subyacentes importantes en los datos completos, afectando a la capacidad del autoencoder para capturar las dependencias necesarias para imputar valores faltantes con precisión.
- **Mayor riesgo de sobreajuste:** En subconjuntos reducidos, el autoencoder puede tender a sobreajustarse a los pocos datos disponibles, lo que afecta su capacidad de generalización al predecir sobre datos no vistos.

5.2.8 SCADA

SCADA ha sido evaluado en términos de su capacidad de imputación bajo dos configuraciones de entrenamiento: utilizando el conjunto de datos completo y utilizando subconjuntos de datos. Se presentan los resultados obtenidos para cada configuración.

Entrenamiento con el Conjunto de Datos Completo

Durante el entrenamiento con el conjunto completo de datos, las gráficas de revelan una rápida convergencia tanto en la pérdida como en la precisión, reflejada en las métricas finales de pérdida y precisión tanto para el entrenamiento como la validación (ver tabla 5.9).

| Métrica | Valor |
|----------------------------------|--------|
| Pérdida de Entrenamiento Final | 0.0155 |
| Pérdida de Validación Final | 0.0138 |
| Precisión de Entrenamiento Final | 0.9987 |
| Precisión de Validación Final | 0.9996 |

Tabla 5.9: Métricas finales de entrenamiento y validación para SCADA con el conjunto completo de datos.

La pérdida de entrenamiento muestra una reducción significativa en las etapas iniciales y se estabiliza rápidamente, lo que sugiere que el modelo es capaz de captar las características esenciales de los datos sin sobreajustarse. La pérdida de validación sigue de cerca a la pérdida de entrenamiento, manteniéndose en niveles bajos, lo que indica una buena generalización del modelo a nuevos datos (ver figura 5.11).

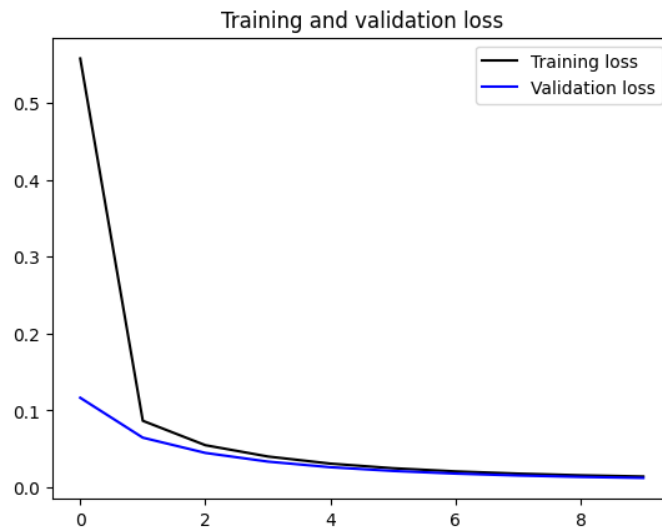


Figura 5.11: Pérdida de entrenamiento y validación para el conjunto completo en SCADA
(Fuente: Elaboración Propia)

En cuanto a la precisión, tanto la precisión de entrenamiento como la de validación alcanzan rápidamente valores cercanos al 100%, y se mantienen estables a lo largo del proceso de entrenamiento. Este alto nivel de precisión refleja la capacidad del modelo para clasificar correctamente los datos sin dificultades significativas(ver figura ??).

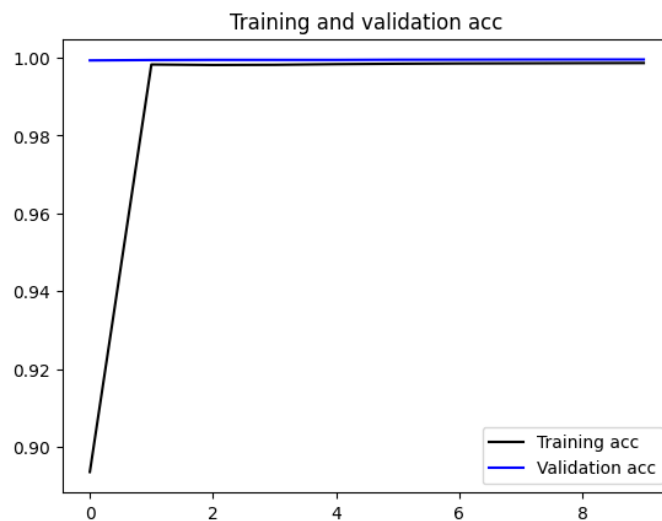


Figura 5.12: Precisión de entrenamiento y validación para el conjunto completo en SCADA
(Fuente: Elaboración Propia).

Aunque se observa una disminución en la precisión promedio en test con el incremento en el porcentaje de pérdida de datos, las métricas se mantienen notablemente altas (ver

tabla 5.10 y figura 5.13). Este fenómeno indica que, incluso con una reducción significativa en la cantidad de datos disponibles, el modelo conserva una considerable capacidad predictiva. Esta característica subraya la robustez del modelo SCADA, especialmente en su habilidad para manejar condiciones de datos incompletos sin una degradación significativa del rendimiento.

| Porcentaje de Pérdida | Precisión Promedio |
|-----------------------|--------------------|
| 10% | 0.9979 |
| 20% | 0.9978 |
| 30% | 0.9976 |
| 40% | 0.9974 |

Tabla 5.10: Precisión promedio observada a medida que aumenta el porcentaje de pérdida de datos en el conjunto completo en SCADA.

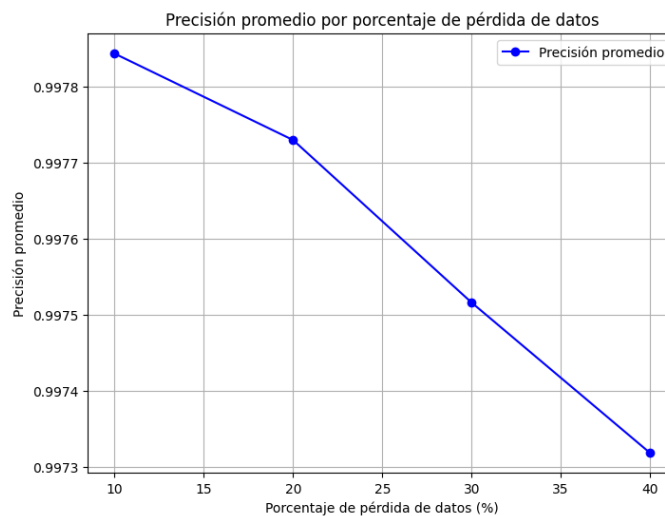


Figura 5.13: Gráfico de la precisión promedio en función del porcentaje de pérdida de datos para con el conjunto completo (Fuente: Elaboración Propia).

Entrenamiento con Subconjunto de Datos

El entrenamiento de SCADA con subconjuntos de datos, al contar con menos ejemplos para aprender, muestra un patrón distinto en comparación con el conjunto completo. Los resultados indican una convergencia más gradual y una variabilidad más notoria tanto en la pérdida como en la precisión (ver tabla 5.11).

| Métrica | Valor |
|----------------------------------|--------|
| Pérdida de Entrenamiento Final | 0.2244 |
| Pérdida de Validación Final | 0.2071 |
| Precisión de Entrenamiento Final | 0.9212 |
| Precisión de Validación Final | 0.9307 |

Tabla 5.11: Métricas finales en entrenamiento y validación para SCADA con subconjunto de datos.

La pérdida de entrenamiento y validación disminuye de manera menos abrupta que en el conjunto completo. Esta disminución más lenta y la variabilidad en la pérdida de validación pueden ser indicativas de desafíos en el aprendizaje a partir de datos reducidos y probablemente menos representativos.

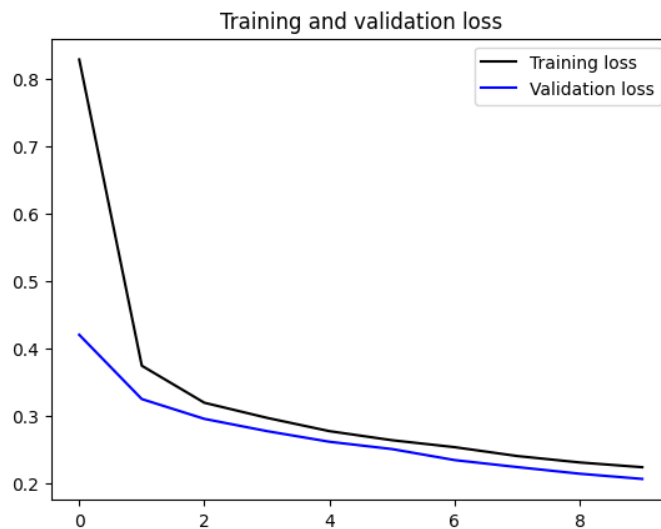


Figura 5.14: Gráfico de la evolución de la pérdida de entrenamiento y validación durante el entrenamiento con subconjunto de datos en SCADA (Fuente: Elaboración Propia)

En términos de precisión, se observa una mejora progresiva en la precisión de entrenamiento, mientras que la precisión de validación no alcanza los mismos niveles altos observados en el conjunto completo. Esto sugiere que el modelo enfrenta limitaciones en la generalización a partir del subconjunto de datos.

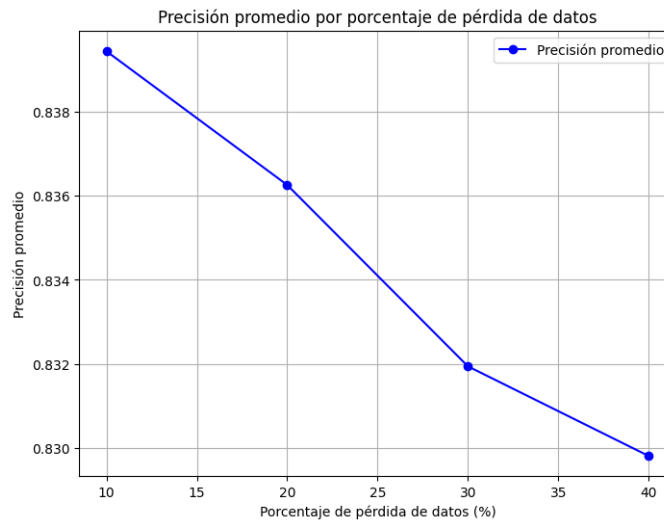


Figura 5.15: Gráfico de la evolución del accuracy de entrenamiento y validación durante el entrenamiento con subconjunto de datos en SCADA (Fuente: Elaboración Propia).

Aunque se observa una disminución progresiva en la precisión promedio en test en los subconjuntos de datos con valores faltantes (ver tabla 5.12 y figura 5.16) con el aumento del porcentaje de pérdida de datos, las métricas se mantienen dentro de un rango aceptable, superando el 80%, lo que demuestra la capacidad del modelo para adaptarse a unos datos con una disminución de características de más de 28 veces respecto al set de datos completo.

| Porcentaje de Pérdida | Precisión Promedio |
|-----------------------|--------------------|
| 10% | 0.8394 |
| 20% | 0.8363 |
| 30% | 0.8319 |
| 40% | 0.8298 |

Tabla 5.12: Precisión promedio en test observada a medida que aumenta el porcentaje de pérdida de datos en SCADA.

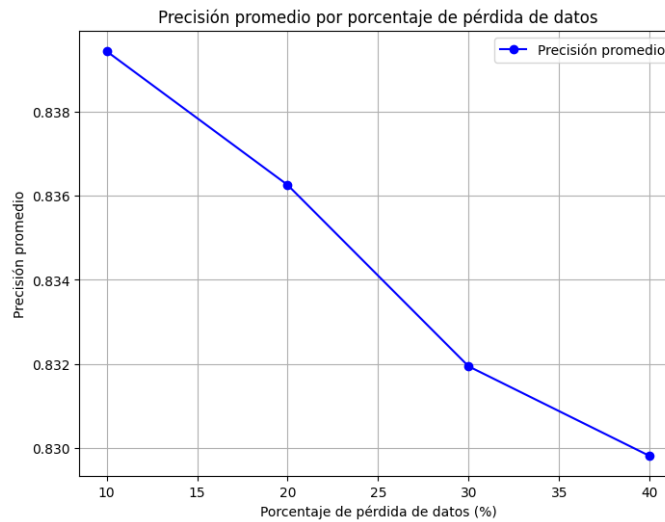


Figura 5.16: Gráfico de la precisión promedio en test observada a medida que aumenta el porcentaje de pérdida de datos en SCADA (Fuente: Elaboración Propia).

Este fenómeno subraya que, a pesar de la reducción significativa en la cantidad de datos disponibles, el modelo aún conserva una capacidad predictiva notable. Sin embargo, es importante destacar que la disminución en las métricas, aunque moderada, refleja los desafíos inherentes al trabajar con menos datos, siendo menos representativos y potencialmente más sesgados. La robustez de SCADA, en este contexto, se demuestra por su habilidad para mantener niveles de precisión razonables incluso bajo condiciones de incompletitud de datos, aunque no alcanza los niveles excepcionalmente altos observados con el conjunto completo de datos.

5.2.9 Comparación del Modelo con Conjunto Completo y Subconjunto de Datos

Al comparar los resultados obtenidos con el conjunto completo y los subconjuntos de datos, se observan diferencias significativas en la capacidad del modelo para generalizar y mantener su rendimiento en la fase de validación.

En el caso del entrenamiento con el **conjunto completo de datos**, SCADA muestra métricas prácticamente perfectas, con una precisión final cercana al 100% tanto en el entrenamiento como en la validación, y una pérdida mínima. Estos resultados reflejan la capacidad del modelo para aprender y generalizar de manera efectiva a partir de un conjunto de datos diverso y representativo. La mínima diferencia entre las métricas de

entrenamiento y validación indica que el modelo no ha sufrido sobreajuste y puede manejar con éxito datos nuevos o faltantes.

Por el contrario, al entrenar SCADA con el **subconjunto de datos reducido**, aunque la precisión final sigue siendo alta (por encima del 92% en el entrenamiento y validación), las pérdidas son significativamente mayores que las observadas en el conjunto completo. Esto sugiere que el modelo ha enfrentado dificultades para generalizar tan bien como lo hizo con los datos completos, posiblemente debido a la falta de diversidad y representatividad en los subconjuntos de datos. El aumento de la pérdida sugiere que el modelo puede estar ajustándose a características específicas de los datos reducidos, afectando negativamente su rendimiento en la imputación.

5.3 Comparativa General

5.3.1 Comparativa de Modelos de ML Tradicionales

Modelos tradicionales, en este caso KNN, RF y XGBoost, presentan limitaciones significativas en el manejo de datos faltantes. La comparación entre estos modelos se enfoca en métricas como Accuracy, F1-Score, Precision y otros indicadores que evidencian la capacidad escasa de estos modelos para lidiar con datos genéticos incompletos.

Resultados de KNN

El modelo KNN muestra un comportamiento predecible: a medida que aumenta el número de vecinos, las métricas de rendimiento mejoran, pero este rendimiento se ve afectado conforme aumenta el porcentaje de datos faltantes. Como se observa, el aumento del número de vecinos mejora el rendimiento del modelo, pero el porcentaje de datos faltantes afecta negativamente las métricas, como se evidencia por el aumento en MSE y MAE, y la disminución de R2 conforme se incrementa el missingness (ver tabla 5.1).

Resultados de RF (modelo único) y XGBoost

El modelo de RF con un único modelo para todas las columnas se enfrenta a dificultades a medida que aumenta el porcentaje de datos faltantes. Este enfoque limita la capacidad del modelo para especializarse en la imputación de valores faltantes por cada característica. Comparado con XGBoost, RF con modelo único muestra un rendimiento ligeramente mejor, pero ambos modelos presentan una métricas muy pobres, teniendo ambos todas las

métricas cercanas a 0.5 (5.13). Ambos métodos basados en árboles no se adaptan bien a la tarea y tienen unos resultados muy mejorables.

| Modelo | Missingness | Accuracy | F1-Score | Precision |
|-----------------------|-------------|----------|----------|-----------|
| Random Forest (Único) | 10% | 0.5364 | 0.5364 | 0.5364 |
| Random Forest (Único) | 20% | 0.5420 | 0.5415 | 0.5424 |
| Random Forest (Único) | 30% | 0.5298 | 0.5277 | 0.5304 |
| Random Forest (Único) | 40% | 0.5156 | 0.5095 | 0.5172 |
| XGBoost | 10% | 0.5275 | 0.5264 | 0.5281 |
| XGBoost | 20% | 0.5260 | 0.5215 | 0.5275 |
| XGBoost | 30% | 0.5197 | 0.5087 | 0.5220 |
| XGBoost | 40% | 0.5141 | 0.4943 | 0.5182 |

Tabla 5.13: Comparativa de métricas de rendimiento para los modelos RF y XGBoost con diferentes niveles de missingness.

5.3.2 Comparativa de Modelos de DL: SCDA y SCADA

Resultados de SCDA y SCADA

Los modelos SCDA y SCADA, ambos basados en autoencoders, muestran un rendimiento muy superior al manejar datos faltantes. SCADA, que incorpora mecanismos de atención, iguala e incluso supera a SCDA en las métricas, especialmente cuando se trata de datos reducidos. Se puede ver en la tabla 5.14, que para el dataset completo, tanto SCDA como SCADA tienen un rendimiento prácticamente similar, con la misma precisión para validación (0.9996) y una precisión muy similar en entrenamiento aunque muy ligeramente superior la de SCADA. Presentan algo más de diferencia en el caso del dataset reducido donde el rendimiento de SCDA es ligeramente superior.

| Modelo | Conjunto de Datos | Precisión de Entrenamiento | Precisión de Validación |
|--------|-------------------|----------------------------|-------------------------|
| SCDA | Completo | 0.9985 | 0.9996 |
| SCDA | Reducido | 0.9326 | 0.9467 |
| SCADA | Completo | 0.9987 | 0.9996 |
| SCADA | Reducido | 0.9212 | 0.9307 |

Tabla 5.14: Comparativa de métricas de precisiones de entrenamiento y validación para SCDA y SCADA con conjuntos completos y reducidos.

Tanto SCDA como SCADA mantienen una precisión promedio casi perfecta en test entrenados con el conjunto completo de datos, incluso con un 40% de pérdida de datos. Este resultado subraya la capacidad de los autoencoders en general para capturar patrones y relaciones profundas en los datos y realizar imputaciones con alta precisión.

| % de Pérdida | Precisión Promedio SCDA | Precisión Promedio SCADA |
|--------------|-------------------------|--------------------------|
| 10.0% | 0.9979 | 0.9979 |
| 20.0% | 0.9978 | 0.9978 |
| 30.0% | 0.9976 | 0.9976 |
| 40.0% | 0.9974 | 0.9974 |

Tabla 5.15: Comparativa de métricas de precisión promedio por porcentaje de pérdida de datos para SCDA y SCADA entrenados con el conjunto completo.

Cuando se utiliza el subconjunto de datos para entrenamiento, SCADA demuestra ser más eficaz que SCDA, aunque con unos resultados similares, se puede ver que sus métricas de precisión descienden proporcionalmente menos a medida que aumenta el porcentaje de pérdida de datos (ver tabla ??). El mecanismo de atención integrado en SCADA le permite manejar mejor los datos incompletos y extraer patrones más útiles, sobre todo cuando los datos de entrenamiento son más reducidos y con ello la dificultad de captar relaciones, mayor.

| % de Pérdida | Precisión Promedio SCDA | Precisión Promedio SCADA |
|---------------------|--------------------------------|---------------------------------|
| 10.0% | 0.8407 | 0.8394 |
| 20.0% | 0.8374 | 0.8363 |
| 30.0% | 0.8336 | 0.8319 |
| 40.0% | 0.8286 | 0.8298 |

Tabla 5.16: Comparativa de métricas de precisión promedio por porcentaje de pérdida de datos para SCDA y SCADA entrenados con subconjunto de datos.

Al comparar los modelos tradicionales (KNN, RF, XGBoost) con los autoencoders (SCDA, SCADA), queda claro que los modelos avanzados tienen un rendimiento mucho mejor al manejar datos faltantes, manteniendo una precisión mucho más alta incluso con niveles elevados de missingness. Los autoencoders son más efectivos al aprender representaciones latentes, lo que permite imputar los valores faltantes con mayor precisión.

6. Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones generales del proyecto, así como una visión sobre las posibles líneas de desarrollo futuro. En la Sección 6.1, se detalla el proceso de obtención de los datos genéticos, destacando los principales retos enfrentados. La Sección 6.2 ofrece un resumen de las conclusiones clave del estudio, y finalmente, en la Sección 6.3, se exploran las perspectivas de evolución en este campo, señalando oportunidades para investigaciones futuras.

6.1 Proceso de Adquisición y Procesamiento de Datos Genéticos: Retos y Limitaciones superadas

En el desarrollo de este trabajo, se gestionó la obtención de datos genéticos de diabetes a través del NIDDK para la aplicación de diversos algoritmos de aprendizaje automático. Este proceso requirió la presentación de una solicitud ante el comité ético de la universidad, que emitió un certificado de idoneidad ética temporal, seguido por la versión definitiva tras la aprobación del comité ético de la universidad, CEI. Este documento se encuentra disponible en el Anexo 2.

Después de varios meses de trámites y envíos de documentos, se recibió en junio uno de los conjuntos de datos solicitados. Estos datos genéticos presentaban una estructura compleja y estaban escasamente documentados, lo que supuso una dificultad para su procesamiento. La institución proporcionó una herramienta específica para su manipulación, aunque dicha herramienta no resultaba compatible con flujos de trabajo modernos, lo que limitó su flexibilidad.

Tras un preprocesamiento inicial, los datos fueron sometidos a distintos algoritmos del estado del arte. Sin embargo, los experimentos presentaron baja reproducibilidad y los resultados obtenidos no fueron consistentes debido a la disparidad en la estructura de los datos, lo que afectaba la integridad de los modelos aplicados.

En respuesta a estas limitaciones, se optó por utilizar un conjunto de datos genéticos alternativo, disponible públicamente en el repositorio del proyecto SCDA¹ (Sparse Convolutional Denoising Autoencoders for Genotype Imputation), que incluye datos de levaduras bien documentados y sin datos faltantes. Estos datos se emplearon para realizar compar-

¹<https://github.com/work-hard-play-harder/SCDA>

ativas con variaciones del modelo de autoencoder original, así como con otros algoritmos de aprendizaje automático, proporcionando un análisis más exhaustivo sobre el comportamiento de diferentes arquitecturas frente a este tipo de datos.

6.2 Conclusiones del Proyecto

El proyecto ha cumplido con éxito el objetivo de explorar y evaluar diferentes algoritmos para la imputación de datos faltantes en un contexto genético. A través de experimentos realizados con el conjunto de datos de levaduras, ha sido posible obtener resultados consistentes y comparables que constituyen un análisis claro del rendimiento de los modelos aplicados.

Se lograron superar las limitaciones iniciales relacionadas con la adquisición de datos y la falta de documentación adecuada. Aunque los datos originales de diabetes no han podido ser utilizados debido a estas limitaciones, el uso de un conjunto de datos de levaduras alternativo permitió llevar a cabo la evaluación de diversas técnicas y arrojar conclusiones valiosas sobre la imputación de datos genéticos incompletos.

Uno de los logros clave del proyecto fue la capacidad de comparar diferentes arquitecturas y métodos de imputación de datos, estableciendo una base sólida para futuras decisiones en estudios genéticos y aplicaciones clínicas. Además, se demostró la viabilidad de usar técnicas avanzadas de aprendizaje automático, como los autoencoders, para mejorar la precisión y la eficiencia en la imputación de datos.

Un avance significativo fue el desarrollo y mejora del modelo base **SCDA**, logrando un progreso notable con la implementación de **SCADA**, una versión mejorada con mecanismos de atención. Esta evolución del modelo base no solo superó a SCDA en la imputación de datos faltantes, sino que también ofreció un rendimiento superior en situaciones con datos más reducidos o con una mayor proporción de missingness. SCADA demostró ser más robusto y eficaz al captar las características más relevantes de los datos, lo que permitió mantener altos niveles de precisión incluso en condiciones adversas. Este avance pone de relieve el potencial de mejorar las arquitecturas existentes mediante la incorporación de mecanismos de atención y otras técnicas de optimización.

6.3 Trabajo Futuro

El trabajo futuro podría enfocarse en varios aspectos clave para mejorar los resultados obtenidos en este proyecto:

6.3.1 Integración con Datos Reales y Heterogéneos

Una de las principales áreas de mejora será la aplicación de estos métodos a datos genéticos humanos reales, con estructuras más complejas y con mayores niveles de heterogeneidad. Esto incluiría la integración de conjuntos de datos más amplios, como aquellos provenientes de estudios clínicos, para evaluar cómo los modelos pueden generalizar a contextos más variados y realistas.

6.3.2 Mejora en la Interpretabilidad de Resultados

Si bien los resultados obtenidos proporcionan una buena base para el análisis de imputación de datos faltantes, la interpretabilidad de estos modelos avanzados sigue siendo un desafío. En futuros trabajos, sería recomendable investigar técnicas que permitan una mejor interpretación de las decisiones tomadas por los modelos, especialmente en contextos clínicos, donde la transparencia es clave para la adopción de estas tecnologías.

6.3.3 Aplicaciones Clínicas Potenciales

El proyecto también abre la puerta para que estos enfoques se utilicen en aplicaciones clínicas. En el futuro, los métodos desarrollados podrían ser aplicados a la imputación de datos genéticos en estudios de diagnóstico genético, lo que podría tener implicaciones significativas en la medicina personalizada. La precisión en la imputación de datos faltantes es crucial en el diagnóstico de enfermedades genéticas y el desarrollo de tratamientos basados en el perfil genético del paciente.

6.3.4 Exploración de Nuevas Tecnologías: Transformers y GAN

Además de las técnicas actuales, sería interesante explorar nuevas tecnologías que han demostrado un gran potencial en el campo de la imputación de datos. En particular:

- **Transformers:** Los modelos basados en Transformers, que han revolucionado el procesamiento de lenguaje natural, podrían ser aplicados en la imputación de datos

faltantes. Su capacidad para capturar dependencias a largo plazo y aprender relaciones complejas entre datos podría mejorar significativamente la imputación de datos genéticos.

- **Generative Adversarial Networks (GAN):** Las GAN, que son populares en la generación de datos sintéticos, podrían ser utilizados para crear datos imputados realistas que completen los valores faltantes en los conjuntos de datos genéticos. La combinación de GAN con otras técnicas de imputación podría mejorar tanto la calidad como la robustez de los datos imputados.

La incorporación de estas tecnologías emergentes no solo abriría nuevas posibilidades para la imputación de datos, sino que también permitiría aprovechar las últimas innovaciones en aprendizaje profundo para abordar uno de los principales desafíos en el análisis de datos genéticos: la falta de información completa y precisa.

6.3.5 Mejora de la Interpretabilidad del Modelo

A medida que los modelos avanzados se aplican a datos genéticos, la interpretabilidad de los resultados se vuelve esencial. Futuras investigaciones podrían enfocarse en mejorar la transparencia de los modelos, desarrollando técnicas que expliquen cómo los autoencoders, transformers o GAN imputan datos faltantes y qué características son las más importantes en el proceso de imputación.

Bibliography

- Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2014). *Molecular biology of the cell*. Garland Science.
- Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular systems biology*, 12(7):878.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- Browning, B. L. and Browning, S. R. (2009). A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. *The American Journal of Human Genetics*, 84(2):210–223.
- Chen, J. and Shi, X. (2019). Sparse convolutional denoising autoencoders for genotype imputation. *Genes*, 10(9):652.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.
- Consortium, H. (2005). A haplotype map of the human genome. *Nature*, 437(7063):1299–1320.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014b). Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680.
- Goodwin, S., McPherson, J. D., and McCombie, W. R. (2016). Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17(6):333–351.
- Howie, B. N., Donnelly, P., and Marchini, J. (2009). A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. *PLoS genetics*, 5(6):e1000529.
- Jiang, Y. and Reif, J. C. (2015). Modeling epistasis in genomic selection. *Genetics*, 201(2):759–768.

- Kojima, K., Tadaka, S., Katsuoka, F., Tamiya, G., Yamamoto, M., and Kinoshita, K. (2020). A genotype imputation method for de-identified haplotype reference information by using recurrent neural network. *PLOS Computational Biology*, 16(10):e1008207.
- Li, M., Li, Y., and Weeks, D. E. (2011). Imputation of missing snps in african americans using the european hapmap reference panel. *Genetic epidemiology*, 35(8):847–856.
- Li, Y., Willer, C. J., Ding, J., Scheet, P., and Abecasis, G. R. (2009). Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genetic epidemiology*, 34(8):816–834.
- Libbrecht, M. W. and Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332.
- Manolio, T. A., Collins, F. S., Cox, N. J., Goldstein, D. B., Hindorff, L. A., Hunter, D. J., McCarthy, M. I., Ramos, E. M., Cardon, L. R., Chakravarti, A., et al. (2009). Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753.
- Mardis, E. R. (2008). Next-generation dna sequencing methods. *Annual review of genomics and human genetics*, 9:387–402.
- McKay, S., Schnabel, R. D., Murdoch, B. M., Matukumalli, L. K., Aerts, J., Coppieters, W., Crews, D. H., Dias-Neto, E., Gill, C. A., Gao, C., Mannen, H., Stothard, P., Wang, Z., Van Tassell, C. P., Williams, J. L., Taylor, J. F., and Moore, S. S. (2007). Whole genome linkage disequilibrium maps in cattle. *BMC Genomic Data*, 8(1).
- Meienberg, J., Bruggmann, R., Oexle, K., and Matyas, G. (2016). Clinical sequencing: is wgs the better wes? *Human genetics*, 135(3):359–362.
- Min, S., Lee, B., and Yoon, S. (2017). Predicting the secondary structure of small rna using deep learning. *Nucleic acids research*, 45(11):3493–3503.
- Poplin, R., Chang, P., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., Newburger, D., Dijamco, J., Nguyen, N., Afshar, P. T., et al. (2018). Prediction of cardiovascular events with genetic variation. *bioRxiv*, page 202178.
- Quang, D. and Xie, X. (2016). Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic Acids Research*, 44:e107–e107.

- Sanger, F., Nicklen, S., and Coulson, A. R. (1977). Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467.
- Shendure, J. and Ji, H. P. (2008). Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145.
- Soares, P., Rocha, L., Pinho, A. J., Bastos, H., Pereira, L., and Carneiro, J. (2012). Snpools: exact and efficient snp selection for association studies. *Bioinformatics*, 28(5):618–619.
- Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001a). Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525.
- Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001b). Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. *arXiv preprint arXiv:1710.10903*.
- Xing, E. P. and Jordan, M. I. (2016). Deep learning for genome-wide genetic prediction. *Nature Reviews Genetics*, 17(7):475–486.
- Zhou, J. and Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 12:931–934.

7. Anexo 1. Repositorio del proyecto

El código fuente de este trabajo, incluyendo los datos en crudo, preprocesados, los modelos generados y las implementaciones de los modelos de imputación de datos genéticos (KNN, RF, XGBoost, SCDA, SCADA) y los experimentos realizados, está disponible en el siguiente repositorio público:

- **Repositorio GitHub:** https://github.com/celiac1/ML_yeast_dna_imputation.git

El repositorio está documentado y preparado para su ejecución, incluyendo un archivo README que detalla los pasos para la instalación, ejecución de los modelos y replicación de los experimentos presentados en este trabajo. Además contiene todos los notebooks ejecutados donde se puede ver el proceso de entrenamiento y test de los modelos y los resultados obtenidos.

8. Anexo 2. Certificado de idoneidad ética de la investigación

En este anexo se incluye el certificado expedido por el comite de ética de la universidad informando de la evaluación positiva de idoneidad ética de la investigación. Se inserta a continuación:

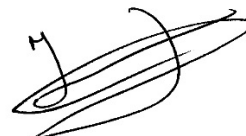
COMITÉ DE ÉTICA DE LA INVESTIGACIÓN

EVALUACIÓN POSITIVA DE IDONEIDAD ÉTICA DE INVESTIGACIÓN

El Comité de Ética de la Investigación de UNIR, en su reunión celebrada el 26 de julio de 2024, ha examinado la solicitud de evaluación de idoneidad ética de la investigación *“Imputación de Datos Genómicos Faltantes con técnicas de Aprendizaje Automático”* cuya Investigadora Principal es Celia Cabello Collado.

A la vista de la documentación presentada, el Comité ha decidido informar favorablemente de la investigación con código PI: 073/2024

Y para que así conste, a petición del interesado, se firma el 31 de julio de 2024



Fdo. Jesús Díaz del Campo,
secretario del CEI