CARRERA DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL

MEMORIA DEL TRABAJO FINAL

Construcción de un modelo para predecir la mortalidad en pacientes en diálisis renal

Autor:
Lic. Ezequiel Scordamaglia

Director:
Esp. Ing. Trinidad Monreal (FIUBA)

Jurados:
Esp. Ing. Fernando Emir Garade (FIUBA)
Dr. Lic. Tobías Canavesi (UNLP)
Ing. José Álamos (HAW Hamburg)

Este trabajo fue realizado en la ciudad de Lanús, Buenos Aires, entre diciembre de 2023 y junio de 2024.

Resumen

En esta memoria se describe el diseño y la implementación de un modelo de inteligencia artificial y la arquitectura necesaria para su utilización, desarrollado para una empresa médica que opera centros de atención en todo el país. El algoritmo utiliza datos clínicos de pacientes en tratamiento de diálisis renal con el propósito de predecir el riesgo de mortalidad. Como resultado, este trabajo permite al personal médico definir estrategias de tratamiento personalizadas para mejorar la salud de los pacientes en riesgo.

Se utilizaron técnicas de estadística y análisis de datos junto con modelos de aprendizaje automático y aprendizaje profundo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general

En este capítulo se presentan los conceptos básicos de la diálisis renal. Además, se mencionan las motivaciones que impulsan este trabajo de investigación, se establecen los objetivos, el alcance y los requerimientos, y se revisa el estado del arte en el campo de estudio.

1.1. Conceptos básicos de la diálisis renal

En esta sección se abordan las definiciones de diálisis renal y sus principales tipos de tratamiento.

1.1.1. Diálisis renal y tipos de tratamiento

La diálisis renal es un tratamiento en el que se extraen las toxinas y el exceso de agua de la sangre. Se utiliza como terapia renal sustitutiva cuando los riñones no funcionan correctamente debido a su deterioro. Los riñones desempeñan un papel crucial al eliminar las toxinas y el líquido de la sangre, lo que evita que los productos de desecho se acumulen en el cuerpo. Cuando los riñones no pueden realizar esta función, la diálisis se convierte en una herramienta vital.

Existen dos tipos principales de tratamientos de diálisis renal [1]:

- Hemodiálisis (HD): en este tratamiento se utiliza una membrana artificial. La purificación de la sangre se lleva a cabo mediante un riñón artificial, que elimina el exceso de agua, residuos y toxinas antes de devolverla al cuerpo. Cada sesión de hemodiálisis puede durar aproximadamente 4 horas y debe realizarse unas 3 veces por semana.
- Diálisis Peritoneal (DP): en este método de tratamiento, la filtración de la sangre se realiza en la cavidad peritoneal del paciente. Se utiliza un catéter permanente que se coloca en el abdomen, a través del cual se introduce una solución especial llamada líquido de diálisis en la cavidad peritoneal. Esta solución absorbe los desechos y el exceso de líquido del cuerpo a través de la membrana peritoneal, que actúa como una barrera semipermeable. Luego de un período de tiempo especificado (generalmente varias horas), el líquido de diálisis se drena del abdomen y se lleva consigo los desechos y el exceso de líquido. Hay dos variantes de diálisis peritoneal:
    - Diálisis Peritoneal Continua Ambulatoria (DPCA): el paciente realiza los intercambios de líquido de diálisis manualmente varias veces al día, mientras sigue con sus actividades diarias. No se requiere una

máquina para realizar los intercambios y el proceso es llevado a cabo por el paciente o su cuidador.

- Diálisis Peritoneal Automática (DPA): en este método, se utiliza una máquina cicladora para realizar los intercambios de líquido de diálisis durante la noche mientras el paciente duerme. La máquina administra automáticamente el líquido de diálisis, lo retira y lo reemplaza según un programa preestablecido. Esto permite una mayor flexibilidad en el tratamiento y puede ser más conveniente para algunos pacientes.

En la figura 1.1 se muestra la diferencia entre ambos tratamientos.

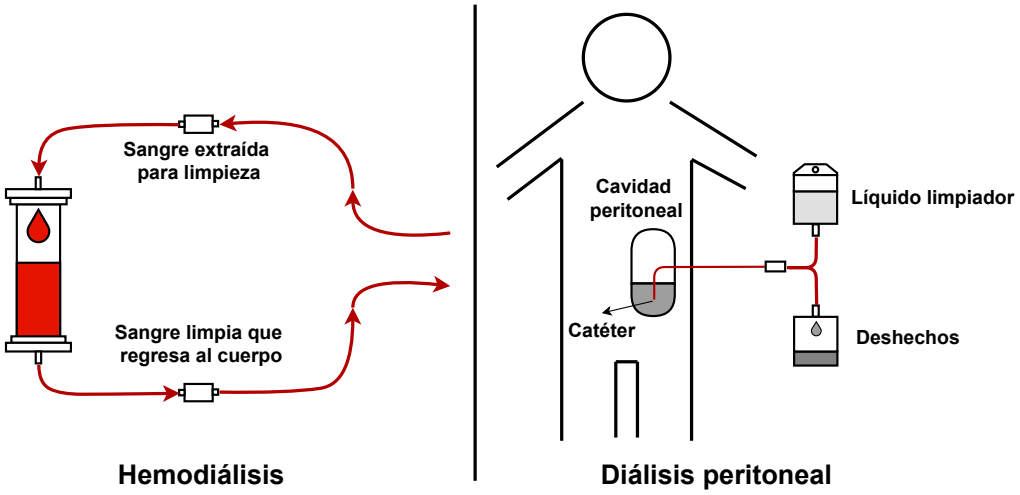

FIGURA 1.1. Diferencia entre hemodiálisis y diálisis peritoneal.

1.2. Contexto y motivación

La motivación principal de este trabajo es incorporar técnicas de inteligencia artificial (IA) en el campo de la medicina, dado que han demostrado una notable capacidad para anticipar eventos futuros basándose en datos históricos [2]. Sin embargo, uno de los desafíos recurrentes a la hora de entrenar modelos de IA es la obtención de datos representativos. En el ámbito de la medicina este desafío también se hace presente ya que se suele contar con pocos datos médicos de una población muy reducida. Para el desarrollo de este trabajo se contó con datos de unos 13 000 pacientes que estuvieron bajo tratamiento de diálisis renal. Tanto el equipo de sistemas como el equipo médico de esta empresa de diálisis renal, si bien carecen de experiencia en herramientas de IA para predicción de eventos, conocen el potencial de estos modelos para identificar patrones, por lo que colaboraron en el desarrollo de este trabajo para lograr el cumplimiento del objetivo.

1.3. Objetivos, alcance y requerimientos

1.3.1. Objetivos

El propósito de este trabajo fue el desarrollo de un modelo de IA que permite predecir el riesgo de mortalidad en pacientes en diálisis renal, junto con la configuración de una plataforma de administración de modelos, una interfaz de comunicación con el modelo y un proceso que solicite las predicciones continuamente.

Este conjunto de herramientas provee una predicción actualizada del riesgo de mortalidad de los pacientes, lo que permite al personal médico adaptar el tratamiento y la medicación prescrita para mejorarles su calidad de vida y, en última instancia, salvar vidas. En la figura 1.2 se muestra la arquitectura de la solución propuesta.

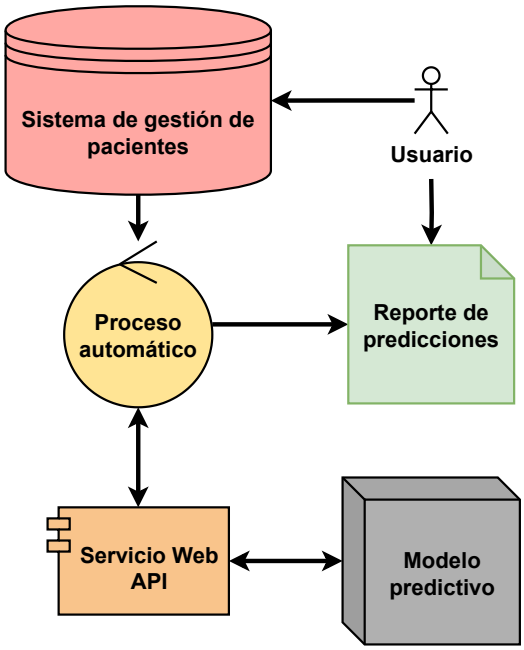

FIGURA 1.2. Arquitectura de la solución.

1.3.2. Alcance

Se encuentra dentro del alcance del trabajo el desarrollo de un modelo de predicción de mortalidad, la instalación y configuración de una plataforma que permita desplegarlo en distintos ambientes, y la construcción de una interfaz y un proceso automático que soliciten predicciones y devuelvan la información al usuario. Asimismo, se incluye en el alcance la obtención y el preprocesamiento del conjunto de datos, que deben estar en cumplimiento con la ley 25.326 para garantizar el derecho al honor y a la intimidad de los pacientes, como así también el análisis de las distintas métricas para evaluar el correcto desempeño del modelo.

Sin embargo, no se encuentra dentro del alcance del proyecto el desarrollo de una plataforma de gestión (sino que se eligió una existente que cumple con los requerimientos del trabajo) ni el desarrollo de una interfaz web orientada al usuario final. Tampoco se encuentra dentro del alcance la instalación del modelo predictivo en el entorno productivo de la empresa médica. Lo único que se instala será el proceso que recupera datos de los pacientes y llama al servicio web cada cierto período de tiempo para recuperar las predicciones.

1.3.3. Requerimientos

A continuación, se listan los requerimientos principales del trabajo agrupados por afinidad:

1. Requerimientos funcionales

   a) La plataforma de gestión de modelos deberá permitir desplegar modelos en diversos ambientes.

   b) La interfaz por servicio web deberá recibir datos médicos de uno o varios pacientes y devolver las predicciones asociadas a ellos.

   c) El modelo predictivo deberá tener una precisión de al menos un 75 %.

   d) El proceso que solicita predicciones y genera el reporte al usuario deberá poder ejecutarse automáticamente cada cierto período de tiempo.

   e) El reporte de predicciones que le llegue al usuario final deberá tener un formato claro y comprensible.

   f) Se utilizará GIT como repositorio para el control de versión de código.

2. Requerimientos de datos a utilizar

   a) Durante el entrenamiento del modelo se deberá resguardar la confidencialidad de los datos de los pacientes.

3. Requerimientos de documentación

   a) Se redactará una memoria técnica con la información del proyecto.

   b) La documentación de la interfaz por servicio web deberá incluir la lista de métodos disponibles con su detalle.

   c) La documentación del modelo predictivo incluirá información sobre el origen de los datos utilizados para el entrenamiento, las características que se usaron, el detalle del modelo seleccionado y la información que haya sobre la explicabilidad del modelo.

1.4. Estado del arte

Se llevó a cabo una exhaustiva revisión de la literatura relacionada con la predicción de la mortalidad de pacientes en diálisis renal y se ha encontrado principalmente una tesis doctoral [3] muy relevante que plantea un objetivo similar pero cuenta con muchos menos datos para el entrenamiento de los modelos. Si bien las variables médicas seleccionadas para realizar las predicciones en dicha tesis son muy similares a las que se seleccionaron en este trabajo, allí se plantea la discriminación de los casos según el tiempo que los pacientes llevan en diálisis, ya sea 3 meses, 6 meses, 1 año o más. Los modelos utilizados en dicha tesis incluyen *Random forest* y regresión logística, algunos de los cuales también fueron utilizados en este trabajo. En cuanto a las conclusiones, para evaluar el desempeño de los modelos se utilizó la métrica de área bajo la curva ROC (AUC), que llega a valores entre 70 % y 73 %. En esta tesis también se muestra qué variables tienen más influencia al realizar la predicción de mortalidad del paciente, lo que resulta sumamente importante para el personal médico. También se ha encontrado una investigación [4] donde se utilizan técnicas de IA para predecir la mortalidad en pacientes con enfermedad renal crónica. Esta también cuenta con muy pocos datos de pacientes y, aunque no se da mucho detalle sobre el entrenamiento de los modelos, concluye indicando que con un modelo de red neuronal se obtiene una predicción superior al 90 %, mejor que con una regresión logística.

Por otro lado, se ha encontrado otra tesis de grado [5] orientada comprobar si el índice neutrófilo/linfocito es un predictor de mortalidad en pacientes que inician hemodiálisis. Si bien no se entrena ningún modelo de IA en la investigación, se concluye en que dicho índice no es un predictor de la mortalidad pero que la edad mayor a 60 años sí representa un factor de riesgo. Existen muchos otros trabajos relacionados con la temática de la predicción de mortalidad en pacientes con enfermedades renales, donde algunos abordan el beneficio que aporta la IA en la detección y predicción de eventos en medicina [6][7][8], otros la predicción de mortalidad teniendo enfermedad renal junto con enfermedad coronaria [9], y otros la predicción de contraer algún cáncer renal [10].

Si bien existe literatura académica que aborda el tema de la predicción de mortalidad de pacientes en diálisis renal, la mayoría de los trabajos se centra en investigaciones de carácter teórico y experimental. Además, parten de conjuntos de datos muy chicos, lo que no permite a los modelos generalizar el conocimiento para poder realizar predicciones correctas.

Este trabajo se destaca por su enfoque práctico, ya que se desarrolla una herramienta concreta que pueda ser implementada en una empresa médica dedicada a la diálisis renal. Se espera obtener predicciones en tiempo real para los pacientes en diálisis renal y en base a su riesgo de mortalidad adecuar el tratamiento y la medicación prescrita. Esta iniciativa ofrece una solución práctica y viable en el ámbito clínico, lo que contribuye a mejorar la atención y seguimiento de los pacientes.

Capítulo 2

Introducción específica

El objetivo de este capítulo es proporcionar una base teórica para comprender las herramientas y métodos utilizados en el desarrollo de este trabajo. En particular, se mencionarán las técnicas para el preprocesamiento de los datos, estrategias para el balanceo de clases, un repaso por los distintos modelos de inteligencia artificial y las métricas utilizadas para su evaluación. También se mencionarán conceptos básicos sobre plataformas de gestión de modelos de IA y servicios web.

2.1. Tratamiento de los datos

Una vez que se obtienen los datos para el entrenamiento de modelos de inteligencia artificial, es fundamental realizar un proceso de tratamiento de los datos para asegurar su calidad y adecuación para el análisis. En primer lugar, la división de los datos en conjuntos de entrenamiento y prueba es una práctica común. Esto se hace para evaluar el rendimiento del modelo de manera efectiva, al emplear un conjunto de datos separado para el entrenamiento y otro para la evaluación, lo que ayuda a evitar el sobreajuste y a evaluar la capacidad de generalización del modelo. Tanto al conjunto de entrenamiento como al conjunto de prueba se le aplican todos los pasos del preprocesamiento que se describen a continuación.

2.1.1. Valores faltantes

Un punto clave en el tratamiento de datos es la corrección de valores inconsistentes o nulos. Esto implica identificar y corregir valores atípicos, faltantes o errores de entrada que podrían distorsionar el análisis y afectar la precisión del modelo. El valor faltante puede darse por diversas razones, como errores durante el ingreso manual de datos, mediciones incorrectas, fallas en el experimento, entre otras [11]. Esta situación suele darse en cualquier conjunto de datos y puede tener un impacto significativo en el análisis y la interpretación de los resultados. Por lo tanto, es crucial entender las causas subyacentes y abordarlas adecuadamente durante el proceso de tratamiento de datos. Usualmente se clasifican en los siguientes grupos:

- Falta completamente al azar (*Missing completely at random* o MCAR): la probabilidad de que un registro tenga un valor faltante para un atributo no depende ni de los datos observados ni de los datos faltantes. Por ejemplo, una muestra de laboratorio que se pierde.
- Falta al azar (*Missing at random* o MAR): indica que la probabilidad de que un registro tenga un valor faltante para un atributo podría depender de los datos observados, pero no del valor del dato faltante en sí mismo. Por

ejemplo, las personas con ingresos más altos pueden ser menos propensas a revelarlos en una encuesta, pero si la falta de respuesta es aleatoria dentro de las clases de ingresos, los datos de ingresos son faltantes al azar.

- Falta no al azar (*Missing not at random* o MNAR): implica que la probabilidad de que un registro tenga un valor faltante para un atributo podría depender del valor del atributo mismo. Por ejemplo, un sensor que no detecta temperaturas por debajo de cierto umbral o personas que no completan los ingresos anuales en encuestas si superan cierto valor.

2.1.2. Técnicas de imputación

Para resolver el problema de los valores faltantes, existen diversos métodos de imputación. Estos son algunos de los más utilizados:

- Imputación por la media o mediana: se remplazan los valores faltantes con la media o la mediana de la variable correspondiente. Esto es simple y rápido, pero puede introducir sesgos si la distribución de los datos es asimétrica o tiene valores atípicos.
- Imputación por el valor más frecuente: se remplazan los valores faltantes con el valor más común de la variable. Es útil para variables categóricas o variables con una distribución de frecuencia clara.
- Imputación por regresión: se utilizan modelos de regresión para predecir los valores faltantes a partir de las variables restantes. Esto puede ser más preciso que los métodos anteriores, pero puede ser computacionalmente intensivo y requiere asumir una relación lineal entre las variables.
- Imputación por vecinos más cercanos (*K nearest neighbor* o KNN): los valores faltantes se estiman a partir de los valores de observaciones similares en el espacio de características. Este método puede ser efectivo en conjuntos de datos con estructuras de vecindario claras, pero puede ser sensible a la elección de la métrica de distancia y número de vecinos.
- Imputación múltiple por ecuaciones encadenadas (*multiple imputation by chained equations* o MICE): se imputan los valores faltantes mediante la estimación secuencial de modelos predictivos y se utilizan las variables restantes como predictores. Este método captura la incertidumbre asociada con la imputación de valores faltantes y puede proporcionar estimaciones más precisas.

Si los datos faltantes se consideran completamente aleatorios (MCAR), los métodos simples como la imputación por la media o mediana pueden ser apropiados. Cuando los datos faltantes son aleatorios (MAR), se pueden utilizar métodos más sofisticados como la imputación por regresión que utilizan información de otras variables observadas. Para datos no aleatorios (MNAR), la elección del método de imputación es más compleja y puede requerir técnicas específicas que modelen la relación entre los datos faltantes y los observados. Siempre es importante tener en cuenta las limitaciones y el contexto específico del conjunto de datos al tomar decisiones sobre el tratamiento de valores faltantes.

2.1.3. Discretización

Otra de las técnicas que se puede aplicar a los datos es la discretización, que es la conversión de variables continuas en variables discretas o categóricas. Esta técnica ayuda a que los modelos generen reglas más breves y comprensibles, lo que reduce la complejidad y también es útil para aumentar la generalización y precisión del conocimiento [12].

2.1.4. Codificación

Para las variables categóricas, se suele realizar una codificación, como *one-hot*, que convierte las variables categóricas en representaciones numéricas y asigna un valor binario a cada categoría. Además, existen otras técnicas de codificación, como la de etiquetas (*label encoding*) y la de frecuencia (*frequency encoding*), que también se utilizan para transformar variables categóricas en datos numéricos compatibles con algoritmos de aprendizaje automático [13].

2.1.5. Valores atípicos

Los valores atípicos, también conocidos como *outliers*, son puntos de datos que se desvían significativamente del resto de la distribución de los datos. Estos pueden ser causados por errores en la recopilación de datos, eventos raros o simplemente representar variaciones genuinas en el fenómeno estudiado. En el contexto del entrenamiento de modelos, los valores atípicos pueden distorsionar los resultados y reducir la precisión del modelo al introducir ruido en los datos. Esto puede llevar a que el modelo aprenda patrones incorrectos o sesgados, lo que afecta negativamente su capacidad para generalizar en datos nuevos.

Para mitigar el impacto de los valores atípicos en el entrenamiento del modelo, existen varias técnicas que se pueden aplicar:

- Eliminación basada en percentiles: consiste en eliminar los valores que se encuentran por encima o por debajo de ciertos percentiles predefinidos en la distribución de los datos. Por ejemplo, se pueden eliminar los puntos por encima del percentil 95 o por debajo del percentil 5.
- *Capping* o truncamiento: esta técnica implica establecer un límite superior e inferior para los valores en el conjunto de datos y luego reemplazar cualquier valor que esté por encima o por debajo de estos límites con el valor del límite correspondiente. Esto ayuda a limitar el impacto de los *outliers* sin eliminar completamente los datos.

Además, algunos algoritmos de modelado son inherentemente robustos a los valores atípicos, como los árboles de decisión y los modelos basados en reglas, lo que los hace preferibles en situaciones donde los *outliers* son comunes o difíciles de eliminar.

2.1.6. Transformaciones de variables

Las transformaciones de variables son técnicas comunes utilizadas para modificar la distribución de los datos y hacer que se ajusten mejor a ciertos supuestos estadísticos, como la normalidad. Dos transformaciones ampliamente utilizadas son:

- Box-Cox: esta transformación busca estabilizar la varianza y hacer que la distribución de los datos se aproxime más a una distribución normal. Es aplicable solo a datos estrictamente positivos y requiere que los datos no contengan ceros. La fórmula de la transformación de Box-Cox se expresa en la ecuación 2.1:

$$y^{(\lambda)} = \begin{cases} \frac{y^{\lambda}-1}{\lambda} & \text{si } \lambda \neq 0 \\ \ln(y) & \text{si } \lambda = 0 \end{cases} \tag{2.1}$$

  donde y es la variable a transformar y λ es el parámetro de transformación que maximiza la normalidad de los datos.

- Yeo-Johnson: esta es una extensión de la transformación de Box-Cox que permite trabajar con datos que incluyen ceros y valores negativos. Esta técnica ajusta la fórmula de Box-Cox para manejar una gama más amplia de datos. La fórmula de la transformación de Yeo-Johnson se expresa en la ecuación 2.2:

$$y^{(\lambda)} = \begin{cases} \frac{(y+1)^{\lambda}-1}{\lambda} & \text{si } \lambda \neq 0, y \geq 0 \\ \ln(y+1) & \text{si } \lambda = 0, y \geq 0 \\ -\frac{(-y+1)^{2-\lambda}-1}{(2-\lambda)} & \text{si } \lambda \neq 2, y < 0 \\ -\ln(-y+1) & \text{si } \lambda = 2, y < 0 \end{cases} \tag{2.2}$$

2.1.7. Normalización

Como último punto se encuentra la normalización de datos, que implica ajustar los valores de las variables de entrada para que tengan una escala similar. Algunas razones para normalizar los datos incluyen:

- Evitar problemas de escala: si las características tienen escalas muy diferentes, algunas características pueden dominar el proceso de entrenamiento, lo que puede llevar a un rendimiento deficiente del modelo.
- Facilitar el proceso de entrenamiento: la normalización puede ayudar al optimizador a converger más rápido durante el entrenamiento, lo que puede resultar en tiempos de entrenamiento más cortos.
- Mejorar la generalización: la normalización también puede contribuir a que el modelo sea más capaz de generalizar correctamente a nuevos datos, al prevenir que se sobreajuste a ciertos rangos específicos de valores de las características.

Existen varias técnicas de normalización [14]:

- Min-max: esta técnica escala los datos a un rango específico, típicamente entre 0 y 1. Cada valor se transforma de acuerdo a la fórmula 2.3:

$$X_{\text{esc}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{2.3}$$

  donde X es el valor original, X_{min} es el valor mínimo en el conjunto de datos y X_{max} es el valor máximo.

- *Z score*: esta técnica conocida como estandarización transforma los datos para que tengan una media de 0 y una desviación estándar de 1. Se utiliza la formula 2.4 para estandarizar.

$$Z = \frac{X - \mu}{\sigma} \tag{2.4}$$

  donde X es el valor original, μ es la media y σ es la desviación estándar del conjunto de datos.

- Normalización por rangos: escala los datos a un rango específico, pero no necesariamente entre 0 y 1. Se puede definir un rango personalizado, como (-1, 1) o (0,5, 1,5), según sea necesario. Se utiliza la formula general 2.5 para normalizar.

$$X_{\text{esc}} = \frac{(X - X_{\text{min}}) \cdot (b - a)}{X_{\text{max}} - X_{\text{min}}} + a \tag{2.5}$$

  donde X es el valor original, X_{min} y X_{max} son el valor mínimo y máximo respectivamente, y a y b son los límites del rango deseado.

La elección de la técnica de normalización dependerá del tipo de datos y del problema en cuestión.

Todas las técnicas anteriormente mencionadas proveen soluciones a los problemas que suelen encontrarse en los conjuntos de datos y ayudan a garantizar la calidad y la adecuación para su posterior análisis y entrenamiento de modelos.

2.2. Desbalance de clases

El desbalance de clases es un problema común en el aprendizaje automático, donde una o más clases están subrepresentadas en comparación con otras en el conjunto de datos [15]. Esto puede ser problemático porque los algoritmos de aprendizaje automático tienden a favorecer a las clases mayoritarias y pueden tener dificultades para aprender patrones en las clases minoritarias. El desbalance de clases puede llevar a modelos sesgados y poco precisos, especialmente en problemas de clasificación donde la precisión de las clases minoritarias es de particular interés, como la detección de fraudes, enfermedades o anomalías.

Algunas técnicas comunes para abordar el desbalance de clases incluyen:

- Submuestreo: se reduce el número de muestras de las clases mayoritarias para equilibrar la proporción de clases. Esto puede ayudar a prevenir el sesgo hacia las clases mayoritarias, pero también puede resultar en la pérdida de información valiosa.

- Sobremuestreo: implica aumentar el número de muestras de las clases minoritarias mediante técnicas como la replicación de instancias o la generación de nuevas muestras sintéticas. Esto puede ayudar a mejorar la representación de las clases minoritarias y a evitar el sesgo hacia las clases mayoritarias.

- Ponderación de clases: algunos algoritmos de aprendizaje automático permiten asignar pesos diferentes a las clases para tener en cuenta el desbalance de clases durante el entrenamiento del modelo. Esto puede ayudar a compensar la falta de representación de las clases minoritarias.

En la figura 2.1 se muestra la diferencia entre las técnicas de submuestreo y sobremuestreo.

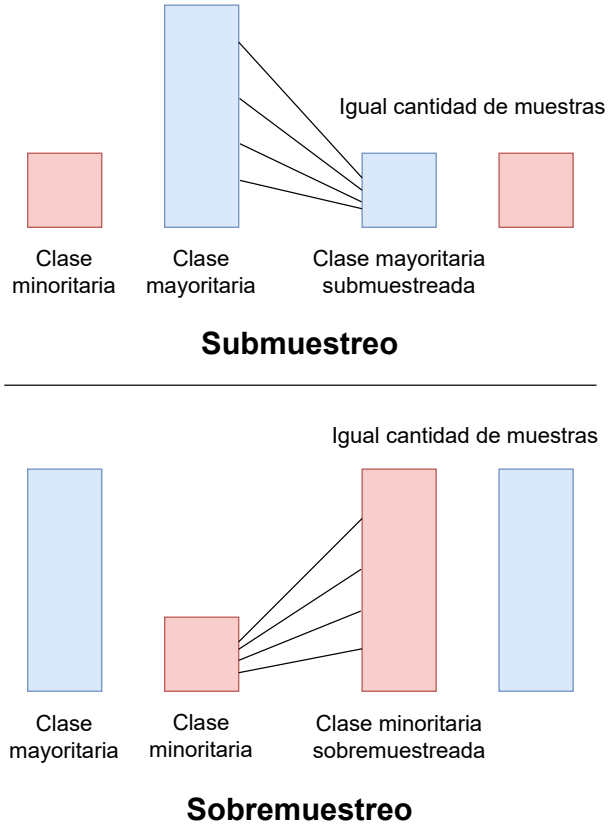

FIGURA 2.1. Submuestreo y sobremuestreo.

Para el caso de submuestreo, existen varias técnicas utilizadas para equilibrar el desbalance de clases. Una de ellas es el submuestreo aleatorio (*Random Undersampling*), donde se elimina aleatoriamente un subconjunto de muestras de la clase mayoritaria para acercarse al número de muestras de la clase minoritaria. Otra técnica es la de muestras cercanas (*NearMiss*), que elimina muestras de la clase mayoritaria que están próximas a las muestras de la clase minoritaria en el espacio de características, y preserva así la información relevante.

En cuanto al sobremuestreo, se puede aplicar la técnica de duplicación de muestras aleatorias (*Random Oversampling*), que consiste en duplicar las muestras de la clase minoritaria para aumentar su representación en el conjunto de datos. Pero también existen técnicas para la generación de muestras sintéticas, como SMOTE (*Synthetic Minority Over-sampling Technique*), que crea nuevas muestras interpoladas entre las muestras existentes de la clase minoritaria, lo que ayuda a mejorar su representación sin duplicar directamente las muestras existentes.

Y por último, también se pueden ajustar ciertas configuraciones de los modelos para que las predicciones incorrectas en las clases minoritarias sean penalizadas durante el entrenamiento. Una técnica ampliamente utilizada para esto es la *focal*

loss (pérdida focal) [16], que aborda el problema del desbalance de clases mediante la modulación de la pérdida de los ejemplos difíciles durante el entrenamiento del modelo. Esta se basa en dos conceptos principales: la reducción de la importancia de los ejemplos fáciles (alpha o parámetro de modulación) y el enfoque en los ejemplos difíciles (gamma o parámetro de focalidad). La función de pérdida *focal loss* utiliza la fórmula 2.6.

$$FL(p_t) = -\alpha_t \cdot (1 - p_t)^{\gamma} \cdot \log(p_t) \tag{2.6}$$

En esta fórmula,

- p_t es la probabilidad predicha por el modelo para la clase objetivo.
- α_t es el peso asociado a la clase objetivo, que puede depender de la probabilidad p_t.
- γ es el parámetro de focalidad que modula la magnitud de la corrección de la focalidad de la pérdida.

Cada técnica tiene sus ventajas y desventajas y muchas veces se usa una combinación de ellas. La elección adecuada depende del conjunto de datos específico y del problema en cuestión. Es importante experimentar con diferentes enfoques y evaluar su rendimiento para determinar la estrategia más efectiva.

2.3. Modelos de inteligencia artificial

Los modelos de inteligencia artificial son comúnmente utilizados para reconocer patrones en grandes conjuntos de datos y obtener predicciones. Se dice que los modelos aprenden cuando logran mejorar sus resultados en una tarea específica luego de procesar muchos datos y sin obtener instrucciones explícitas de un programador [2]. Los tipos de aprendizaje se dividen en:

- Aprendizaje supervisado: para entrenar al modelo se utiliza un conjunto de datos etiquetados. Esto quiere decir que se le provee tanto las características como el valor objetivo esperado. El modelo aprende a hacer predicciones basadas en estos ejemplos y se ajusta para minimizar los errores entre las predicciones y las etiquetas conocidas.
- Aprendizaje no supervisado: no se utilizan etiquetas en los datos de entrenamiento. El modelo explora patrones y estructuras en los datos sin una guía explícita. Este enfoque es útil cuando no se conocen las categorías de antemano y se quiere descubrir patrones ocultos.
- Aprendizaje por refuerzo: el modelo aprende a tomar decisiones a través de la interacción con un entorno. Para cada acción recibirá recompensas o castigos según su desempeño. El objetivo es maximizar las recompensas a lo largo del tiempo.

Teniendo los datos y el tipo de aprendizaje que se desea implementar, se debe buscar también una arquitectura de modelo que tenga la capacidad de aprender de los datos y devolver predicciones. Dentro de la inteligencia artificial, existe un universo que se conoce como *Machine Learning* (ML) o aprendizaje automático, que refiere a aquellos algoritmos que utilizan métodos estadísticos para analizar

datos, aprender de ellos y elaborar predicciones o sugerencias. Entre los modelos de ML más conocidos se encuentran los siguientes:

- Regresión logística: se utiliza para clasificación binaria. Estima la probabilidad de que una instancia pertenezca a una determinada clase.
- Árboles de decisión: organizan las características de los datos en una estructura similar a un árbol. Cada nodo en este árbol representa una pregunta sobre una característica específica de los datos. Por ejemplo, podría ser ¿Tiene diabetes? o ¿Es menor de 30 años?. Las ramas del árbol representan las posibles respuestas a estas preguntas, como sí o no. Siguiendo las ramas del árbol, eventualmente se llega a una hoja que representa la decisión o predicción final.
- Bosque aleatorio (*Random forest*): es una técnica de ensamble que combina múltiples árboles de decisión. En su proceso de entrenamiento, cada árbol se construye utilizando una técnica llamada *bootstrapping*. El *bootstrapping* implica muestrear aleatoriamente el conjunto de datos original con reemplazo, lo que significa que cada muestra puede seleccionarse más de una vez y que algunas muestras pueden no ser seleccionadas en absoluto. Este proceso crea múltiples conjuntos de datos de entrenamiento, cada uno ligeramente diferente del original. Luego, se entrena un árbol de decisión con cada uno de estos conjuntos de datos de entrenamiento, lo que resulta en múltiples árboles de decisión que capturan diferentes aspectos y variaciones del conjunto de datos original. Durante la etapa de predicción, las predicciones de cada árbol se promedian para obtener la salida final del bosque aleatorio. Este enfoque de combinación de múltiples modelos ayuda a reducir el sobreajuste y mejorar la generalización del modelo, ya que los árboles individuales tienden a especializarse en diferentes regiones o aspectos del espacio de características.
- *Support vector machine* (SVM): es un algoritmo de clasificación que encuentra el hiperplano que mejor separa las clases en un espacio de características de alta dimensión. Puede ser usado tanto para clasificación como para regresión.

Dentro del universo de ML, hay otro grupo más chico que se denomina *Deep Learning* (DL) o aprendizaje profundo, en donde los algoritmos utilizan una arquitectura de redes neuronales que simulan el comportamiento del cerebro humano, por lo que suelen ser mucho más grandes y complejos. Estos últimos suelen usarse para tareas de visión por computadora o procesamiento de lenguaje natural, y requieren mucha potencia de cómputo y grandes cantidades de datos. En la figura 2.2 se puede visualizar la diferencia entre utilizar modelos de *Machine Learning* y modelos de *Deep Learning* para obtener predicciones.

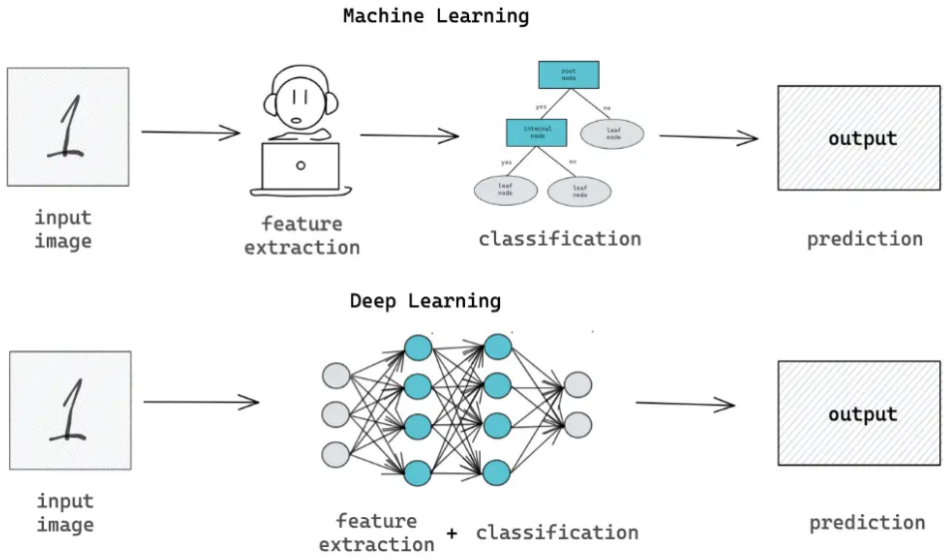

FIGURA 2.2. *Machine Learning* vs *Deep Learning*. [1].

Los modelos también se diferencian por el tipo de predicción que realizan. A un modelo que predice un valor continuo, como el precio de una vivienda, se lo conoce como modelo de regresión. Por otra parte, a un modelo que predice una etiqueta o clase, ya sea binaria o multi-clase, se lo conoce como modelo de clasificación. Este último tipo de modelo es muy utilizado en medicina, ya que sirve para detectar la presencia o ausencia de cierta enfermedad o para predecir su tipo específico. En este trabajo en particular se utilizaron modelos de clasificación con aprendizaje supervisado y se entrenaron modelos de *Machine Learning* y *Deep Learning*.

2.4. Evaluación de modelos de clasificación

Para evaluar el comportamiento de un modelo de clasificación binaria comúnmente se recurre a una herramienta llamada matriz de confusión. Esta matriz, como se muestra en la figura 2.3, presenta las clases predichas en las columnas y las clases reales en las filas. A partir de esta tabla, se derivan cuatro métricas clave:

- Verdaderos positivos (*True Positive* o TP): representan las predicciones correctas de una condición positiva.
- Verdaderos negativos (*True Negative* o TN): son las predicciones correctas de una condición negativa.
- Falsos positivos (*False Positive* o FP): se dan cuando el modelo predice incorrectamente una condición positiva que en realidad no lo es.
- Falsos negativos (*False Negative* o FN): se dan cuando el modelo predice incorrectamente la ausencia de una condición positiva que en realidad está presente.

[1]Imagen tomada de https://www.stratascratch.com/blog/data-science-vs-machine-learning-vs-deep-learning-the-difference/

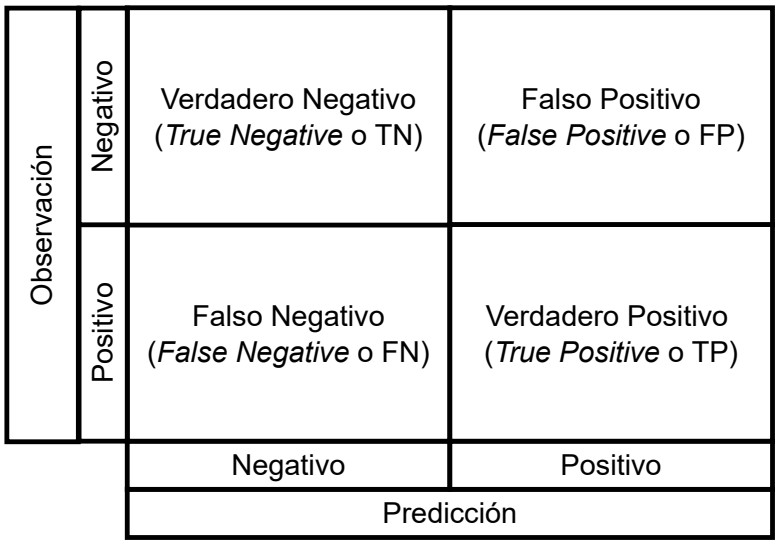

FIGURA 2.3. Matriz de confusión.

En contextos médicos, los falsos negativos pueden tener consecuencias significativas para la salud del paciente, ya que se está prediciendo que un paciente no tiene cierta condición cuando en realidad la tiene.

Particularmente en este trabajo, los falsos negativos podrían implicar no tomar medidas necesarias para un tratamiento adecuado, lo que aumentaría el riesgo de mortalidad del paciente. Por lo tanto, es fundamental minimizar la tasa de falsos negativos tanto como sea posible, incluso si esto conlleva un aumento en los falsos positivos.

Derivados de estos 4 valores de la matriz de confusión se definen las siguientes métricas para evaluar el rendimiento de un modelo de clasificación:

- *Precision*: es la proporción de ejemplos positivos que fueron correctamente clasificados como positivos respecto al total de ejemplos clasificados como positivos. Es decir, mide la calidad de las predicciones positivas del modelo. Se calcula como TP / (TP + FP).
- *Recall*: es la proporción de ejemplos positivos que fueron correctamente clasificados como positivos respecto al total de ejemplos que son realmente positivos. Es decir, mide la capacidad del modelo para encontrar todos los ejemplos positivos. Se calcula como TP / (TP + FN).
- *F1-Score*: es la media armónica de la *precision* y el *recall*. Proporciona un equilibrio entre ambas métricas y es útil cuando hay un desequilibrio entre las clases. Se calcula como 2 * (*precision* * *recall*) / (*precision* + *recall*).
- *Accuracy*: es la proporción de ejemplos clasificados correctamente (tanto positivos como negativos) respecto al total de ejemplos. Es una métrica general de la calidad del modelo en todas las clases. Se calcula como (TP + TN) / (TP + TN + FP + FN).
- AUC (*Area Under the Curve*): el AUC es el área bajo la curva ROC (*Receiver Operating Characteristic*). La curva ROC es una representación gráfica de la sensibilidad frente a la tasa de falsos positivos para diferentes umbrales de clasificación. El AUC mide la capacidad del modelo para distinguir entre

clases positivas y negativas. Un valor de AUC cercano a 1 indica un buen rendimiento del modelo, mientras que un valor cercano a 0,5 indica un rendimiento aleatorio.

En este trabajo se utilizaron las métricas antes mencionadas para la evaluación de los modelos entrenados, y se prestó especial atención a las métricas de *F1-Score* y *Recall*, ya que existe un desbalance entre las clases y es importante detectar correctamente a la mayoría de casos positivos.

2.4.1. Técnicas utilizadas en el entrenamiento de modelos

Durante el entrenamiento de modelos de *Deep Learning* se pueden aplicar distintas técnicas o herramientas que ayuden al modelo a aprender correctamente de los datos y a mantenerse en buena forma. Una de ellas es la técnica de *early stopping* [17], que se utiliza para prevenir el sobreajuste deteniendo el entrenamiento cuando el rendimiento deja de mejorar, lo que ayuda a obtener mejores resultados y reducir los tiempos de procesamiento. Otra técnica es la planificación de tasa de aprendizaje, que ajusta dinámicamente el valor de este hiperparámetro después de un número predefinido de épocas o pasos de entrenamiento para mejorar la convergencia del modelo. Por último, se menciona también dos técnicas de regularización: la L2 (*Ridge Regularization*), que penaliza los pesos grandes en el modelo para evitar el sobreajuste y mejorar la estabilidad, y *dropout*, que apaga aleatoriamente un porcentaje de unidades en una red neuronal para mejorar la generalización y prevenir la coadaptación entre las neuronas. Estas técnicas son esenciales para optimizar el entrenamiento y mejorar la capacidad predictiva del modelo, que será más robusto y preciso.

2.5. Plataforma de gestión de modelos

Una plataforma de gestión de modelos de IA es una herramienta diseñada para ayudar a los equipos de desarrollo y científicos de datos a gestionar, monitorear y desplegar modelos de manera eficiente. Estas plataformas ofrecen las siguientes ventajas:

- Centralización: permiten centralizar todos los modelos en un único lugar para facilitar su gestión y acceso.
- Implementación automatizada: facilitan la implementación de modelos en entornos de producción y proporcionan herramientas para la integración y la implementación continua (*Continuous integration and delivery/deployment* o CI/CD).
- Versionado: ofrecen capacidades de versionado para los modelos, lo que facilita el seguimiento de cambios y la colaboración entre equipos.

Existen múltiples plataformas para gestión de modelos *Open Source*, tales como MLFlow [18], Apache Airflow [19] o Jenkins [20], que se conectan directamente con repositorios en la nube, como GitHub, y descargan versiones de modelos para aplicar en distintos ambientes. También cuentan con la posibilidad de realizar rentrenamientos automáticos de los modelos cuando se disponga de nuevos conjuntos de datos, aunque este punto queda fuera del alcance de este trabajo. Se decide optar por Jenkins como plataforma de gestión de modelos, ya que se

utiliza actualmente para administrar otros recursos de la organización y puede configurarse para desplegar modelos sin mayores inconvenientes.

2.6. Servicios web

Una vez que el modelo de IA fue entrenado y desplegado en un algún ambiente, el siguiente paso es exponerlo a través de un servicio web. Generalmente se desarrolla utilizando una API (*Application Programming Interface*), que es una interfaz que define cómo comunicarse con el servicio web. Comúnmente se utilizan APIs para la interacción entre diferentes sistemas informáticos, lo que facilita el intercambio de datos y la ejecución de funciones remotas. Se suelen emplear para solicitar datos de alguna fuente de información desde aplicaciones web y móviles. Los servicios web se caracterizan por estar estructurados con métodos que requieren parámetros específicos y proporcionan respuestas predefinidas, lo que simplifica la interacción entre sistemas. Esta estandarización les otorga independencia de plataforma y lenguaje, lo que los convierte en herramientas altamente adaptables y versátiles.

En esta caso, la API actúa como un intermediario entre la aplicación cliente y el modelo de IA, lo que facilita la comunicación y la interacción. Como primer paso, recibe los datos del cliente en un formato específico y realiza su preprocesamiento, tal como se hizo durante el entrenamiento del modelo. Una vez procesados los datos, se envían al modelo de IA para obtener una predicción como respuesta. Dicha predicción es la que se devuelve al cliente como respuesta del llamado a la API.

Capítulo 3

Diseño e implementación

En este capítulo se presentan las tareas realizadas para el cumplimento de los objetivos. Se abordan temas como la adquisición de los datos, su preprocesamiento, las técnicas empleadas para balancear las clases, los modelos de IA utilizados y las herramientas para el acceso al modelo.

3.1. Arquitectura propuesta

Para cumplir con los objetivos planteados en este trabajo se pensó en una solución integral, donde fue necesario desarrollar y configurar una serie de herramientas que interactúen entre si para lograr que se generen predicciones de mortalidad de forma automática.

En la figura 3.1 se muestra la arquitectura desarrollada.

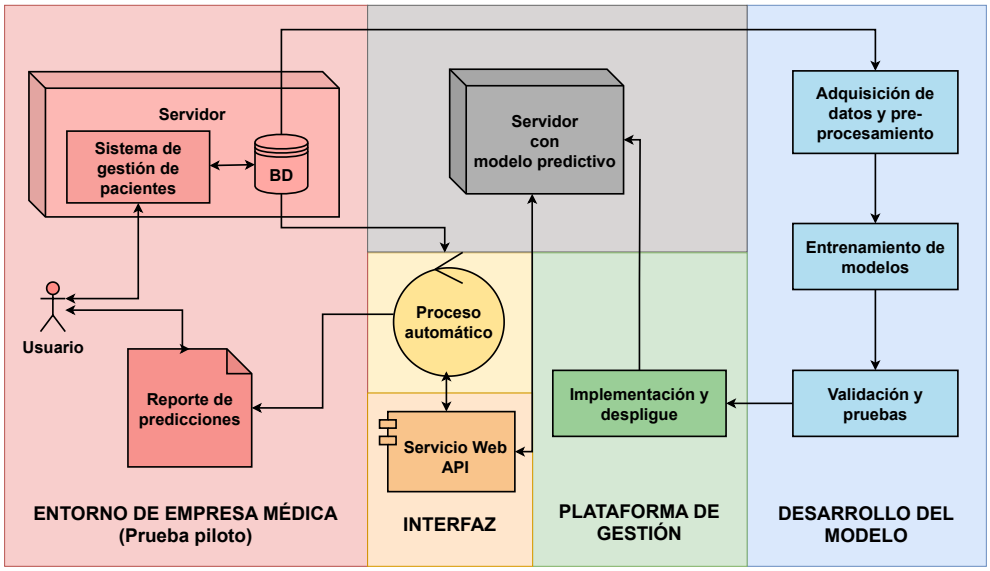

FIGURA 3.1. Arquitectura del sistema.

La solución implementada comienza con la adquisición de datos y entrenamiento de modelos, seguida por la selección e implementación del mejor modelo en un servidor. Luego se desarrolla un servicio web para el acceso al modelo y, por último, una aplicación automática que solicite las predicciones y se las devuelva al usuario.

En las próximas secciones se describen con más detalle cada una de estas etapas.

3.2. Adquisición de datos

En el sistema de gestión de pacientes de la empresa médica se registran datos clínicos de pacientes en tratamiento de diálisis renal. Entre los datos que se registran, se encuentran:

- Datos personales de los pacientes, como nombre, apellido, documento, sexo y fecha de nacimiento.
- Datos del tratamiento de diálisis, como turno, grupo de días, motivo de ingreso, centro de atención, primera diálisis en la vida, etiología de la enfermedad renal primaria y diuresis residual.
- Evaluación clínica, donde se registra el peso, la altura, la presión arterial y el tiempo en diálisis.
- Exámenes de laboratorio, en los que algunas variables se analizan todos los meses, otras cada 3 meses y otras cada 6 meses.
- Accesos vasculares, donde se registra el tipo, la localización, la fecha de confección, la fecha de primer uso, los eventos y los procedimientos realizados.
- Prescripción de medicamentos, como por ejemplo la administración de eritropoyetina y hierro.
- Condiciones médicas, como diabetes, en espera para trasplante y otras.

Todos los centros de atención utilizan el mismo sistema de gestión de pacientes para registrar los datos, por lo que se encuentran consolidados a nivel nacional.

Como primer paso en la adquisición de los datos, se seleccionaron las variables médicas que tienen más influencia en la mortalidad de los pacientes. Esto se realizó en conjunto con un médico de la empresa de diálisis renal, quien aportó información muy valiosa sobre el impacto que tiene cada una. Se propusieron más de 80 variables que se registran en el sistema, de las cuales solo se seleccionaron 37. Las variables seleccionadas son las siguientes:

- Motivo de ingreso: indica si el motivo de ingreso fue programado o no programado.
- Edad: representa la edad en años del paciente.
- Sexo: representa el sexo del paciente (masculino/femenino).
- Altura: representa la altura del paciente en el último mes en centímetros.
- Peso: representa el peso del paciente en el último mes en kilogramos.
- Tiempo en diálisis: indica el tiempo que el paciente se dializa expresado en minutos.
- *Mean Arterial Pressure* (MAP): representa la presión arterial media y se calcula con la fórmula siguiente: tensión diastólica + ((tensión sistólica - tensión diastólica) / 3).
- Hemoglobina: representa el valor de hemoglobina del último mes en g/dl.
- Urea pre: representa el valor de urea pre diálisis del último mes en mg/dl.
- Urea post: representa el valor de urea post diálisis del último mes en mg/dl.

- Potasio: representa el valor de potasio del último mes en mEq/l.
- Glucemia: representa el valor de glucemia del último mes en mg/dl.
- Calcio: representa el valor de calcio del último mes en mg/dl.
- Fósforo: representa el valor de fósforo del último mes en mg/dl.
- Albúmina: representa el valor de albúmina del último mes en g/dl.
- Bicarbonato (HCO3): representa el valor de bicarbonato del último mes en mEq/l .
- Saturación transferrina: representa el porcentaje de saturación transferrina. Se toma el último valor de los últimos 3 meses.
- Ferritina plasmática: representa el valor de ferritina plasmática en ng/ml. Se toma el último valor de los últimos 3 meses.
- Pth (immulite): representa el valor de pth en pg/ml. Se toma el último valor de los últimos 3 meses.
- Anti-HIV: representa el valor de anticuerpos contra el virus de la inmunodeficiencia humana por el método ELISA. Se toma el último valor de los últimos 6 meses. Los valores posibles son (N = no reactivo / R = reactivo).
- Anti-HCV: representa el valor de anticuerpos contra el virus de la hepatitis C por el método ELISA. Se toma el último valor de los últimos 6 meses. Los valores posibles son (N = no reactivo / R = reactivo).
- HBsAg: representa el valor de antígeno de superficie del virus de la hepatitis B por el método ELISA. Se toma el último valor de los últimos 6 meses. Los valores posibles son (N = no reactivo / R = reactivo).
- Anti-HBs: representa el valor de anticuerpos contra el antígeno de superficie del virus de la hepatitis B por el método ELISA en ul/l. Se toma el último valor de los últimos 6 meses.
- HCV PCR: representa el valor de anticuerpos contra el virus de la inmunodeficiencia humana por el método PCR. Se toma el último valor de los últimos 6 meses. Los valores posibles son (N = negativo / P = positivo).
- HBsAg PCR: representa el valor de antígeno de superficie del virus de la hepatitis B por el método PCR. Se toma el último valor de los últimos 6 meses. Los valores posibles son (N = negativo / P = positivo).
- Hemoglobina glicosilada: representa el porcentaje de hemoglobina glicosilada del último mes.
- Vitamina D 25oh: representa el valor de vitamina D 25oh del último mes en ng/ml.
- Kt/V: medida de la eficacia del tratamiento de diálisis. Se calcula en base a la urea pre y post, el balance, el peso post, el tiempo en diálisis, el sexo, la edad y la altura.
- Diabético: indica si el paciente es diabético.
- Etiología de la enfermedad renal primaria: indica la causa subyacente que conduce al desarrollo de la enfermedad renal.

- Tipo de acceso vascular (AV): indica el tipo del primer acceso vascular: catéter liso, tunelizado, prótesis o fístula arteriovenosa.
- Días en diálisis en la vida: representa la cantidad de días desde la primera diálisis en la vida hasta el día de la fecha.
- Grupo sanguíneo: indica el tipo de sangre y factor.
- Lista de espera para trasplante: representa la situación en relación a la lista de espera para el trasplante.
- EPO (dosis mensual): representa la dosis mensual de eritropoyetina prescrita en el último mes.
- Hierro (dosis mensual): representa la dosis mensual de hierro prescrita en el último mes.

Se definió también que solo se usarían datos de pacientes en tratamiento de hemodiálisis y que tengan más de 90 días en tratamiento, ya que antes de ese lapso de tiempo la mortalidad es mucho mayor.

Una vez definidas las variables y las condiciones de los pacientes a utilizar, se prepararon las consultas a la base de datos para obtener dicha información. Para esto se empleó una base de datos que contiene información actual e histórica, ya que se guarda al final de cada mes una foto del estado de salud de los pacientes en ese momento, que incluye a los activos y a los que fallecieron en dicho mes.

Se definieron dos estrategias para el armado de los conjuntos de datos de entrenamiento y pruebas:

1. Todos los pacientes del padrón que tuvieron tratamiento HD y que hayan superado los 90 días de tratamiento. Cada paciente cuenta con datos del último examen de laboratorio. Son 11 961 registros de los cuales el 63,7 % falleció.
2. Todos los pacientes que existan en todas las fotos mensuales de todos los meses entre los años 2016 y 2023, que lleven más de 90 días en tratamiento HD al momento de tomar la foto. Son 284 466 registros, donde solo el 1,5 % falleció. En esta estrategia un mismo paciente puede aparecer varias veces con estados de salud distintos.

Las dos estrategias contienen las mismas 36 variables por cada registro. Cada conjunto de datos de las distintas estrategias se trabajó en proyectos separados, aunque compartieron gran parte del preprocesamiento de los datos.

También se armó un conjunto de datos para realizar la validación, que incluyó a los pacientes que llevan más de 90 días en tratamiento HD con su estado al final del mes de enero de 2024, tanto fallecidos como no fallecidos. Son unos 2807 registros, donde el 1 % falleció. Esta es la situación más parecida a la realidad, ya que el modelo estará constantemente evaluando a pacientes activos para detectar a ese 1 % con alto riesgo de mortalidad.

3.3. Preprocesamiento de datos

Como primer paso, cada conjunto de datos se dividió en dos partes: 75 % para realizar el entrenamiento y el 25 % restante para probar el desempeño del modelo. Esto permite evaluar qué tan bien generaliza el modelo con datos nuevos y ayuda medir su eficacia en situaciones reales.

Es importante destacar que en el caso de la estrategia 2 un mismo paciente puede repetirse varias veces con distintos valores para sus características. Por esto se tuvo la precaución al hacer la división entre entrenamiento y pruebas de que un mismo paciente no se encuentre en ambos conjuntos. También se detectó que los pacientes que se encontraban muy mal de salud y terminaban egresando por trasplante, aportaban información confusa al modelo. Estos casos conceptualmente tiene un alto riesgo de mortalidad y estaban clasificados con la etiqueta de "no fallecido". Por este motivo fueron eliminados de los conjuntos de datos de ambas estrategias.

Durante el análisis exploratorio de los datos recibidos, se identificaron ciertas inconsistencias que requerían corrección, como edades superiores a 120 años, alturas de 0 cm o pesos inferiores a 10 kg. Estos datos inconsistentes se marcaron como faltantes para su posterior completitud.

Algunas de las variables seleccionadas, si bien contenían información sumamente importante, contaban con gran cantidad de valores faltantes. Esto se resolvió de forma distinta para las variables numéricas y categóricas.

3.3.1. Preprocesamiento de variables numéricas

Como primer punto, se analizaron los valores atípicos (*outliers*) en las variables numéricas. En las figuras 3.2 y 3.3 se observan los diagramas de caja o *box plot* de ambas estrategias de datos, donde se observa la mediana, los cuartiles y los *outliers* de cada variable numérica.

Para disminuir el ruido generado por estos valores atípicos, se aplicó la técnica de *capping* o truncamiento. Se definieron como límites inferior y superior la distancia de 4 desvíos estándar desde la media y se reemplazó todo valor que estuviera por fuera del intervalo con el límite correspondiente.

En relación al análisis de datos faltantes en las variables numéricas, se dividió a éstas en dos grupos según su situación:

- Variables con gran cantidad de datos faltantes: si bien en las primeras pruebas se aplicó la técnica de discretización y codificación *one hot* para intentar mantener esta información, en posteriores pruebas se decidió eliminarlas, ya que en el conjunto de validación estas variables no tenían datos. Las variables que se eliminaron son: hemoglobina glicosilada, vitamina D 25oh y bicarbonato. En la tabla 3.1 se muestra el porcentaje de valores faltantes de estas variables eliminadas en ambas estrategias.

  Para el caso particular de las variables EPO y hierro, que también contenían gran cantidad de valores faltantes, se decidió completar con ceros, ya que se comprobó en los sistemas de información que solo se registran valores cuando se prescriben dichos fármacos.

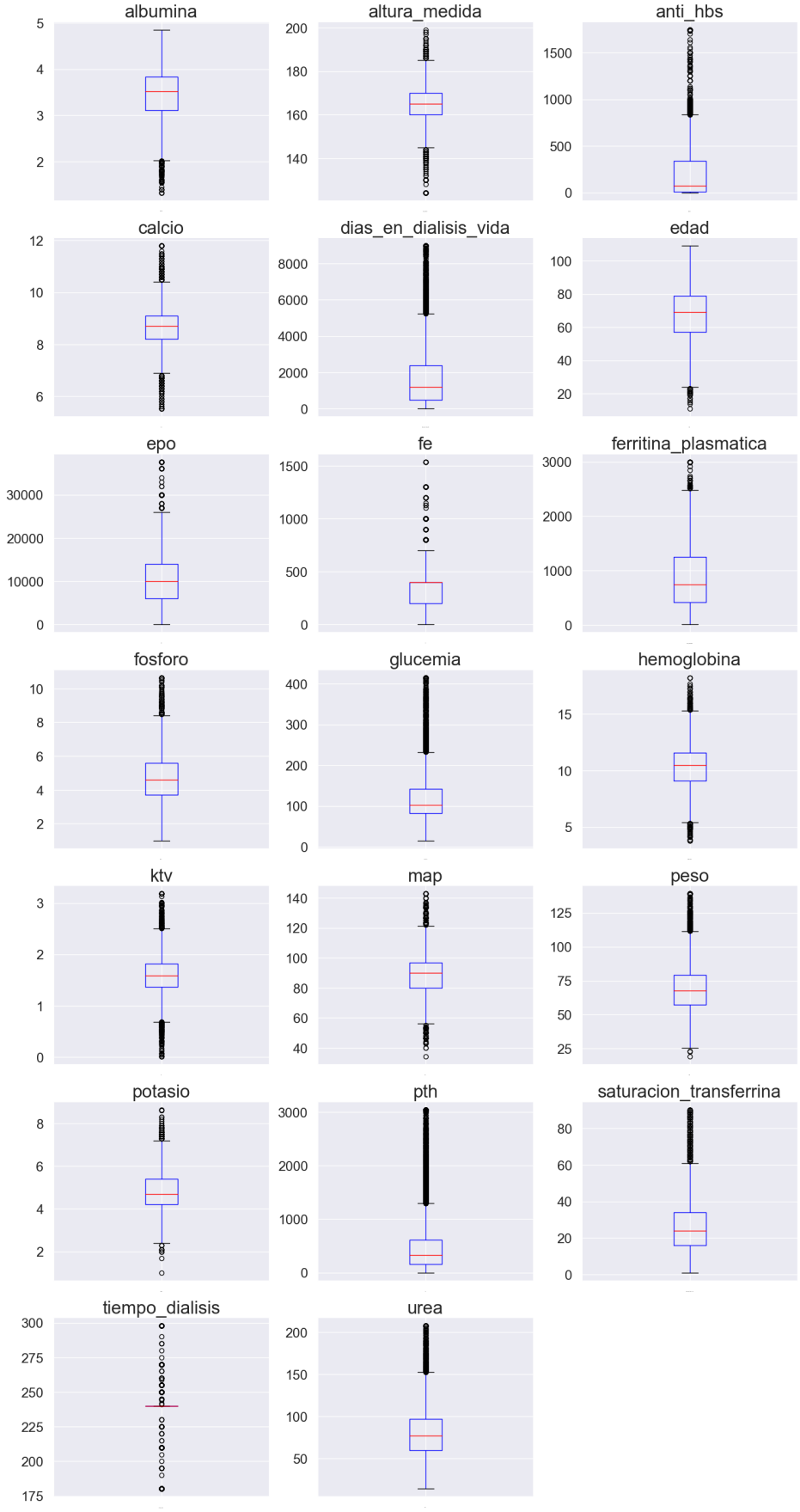

FIGURA 3.2. *Outliers* en variables numéricas de estrategia 1.

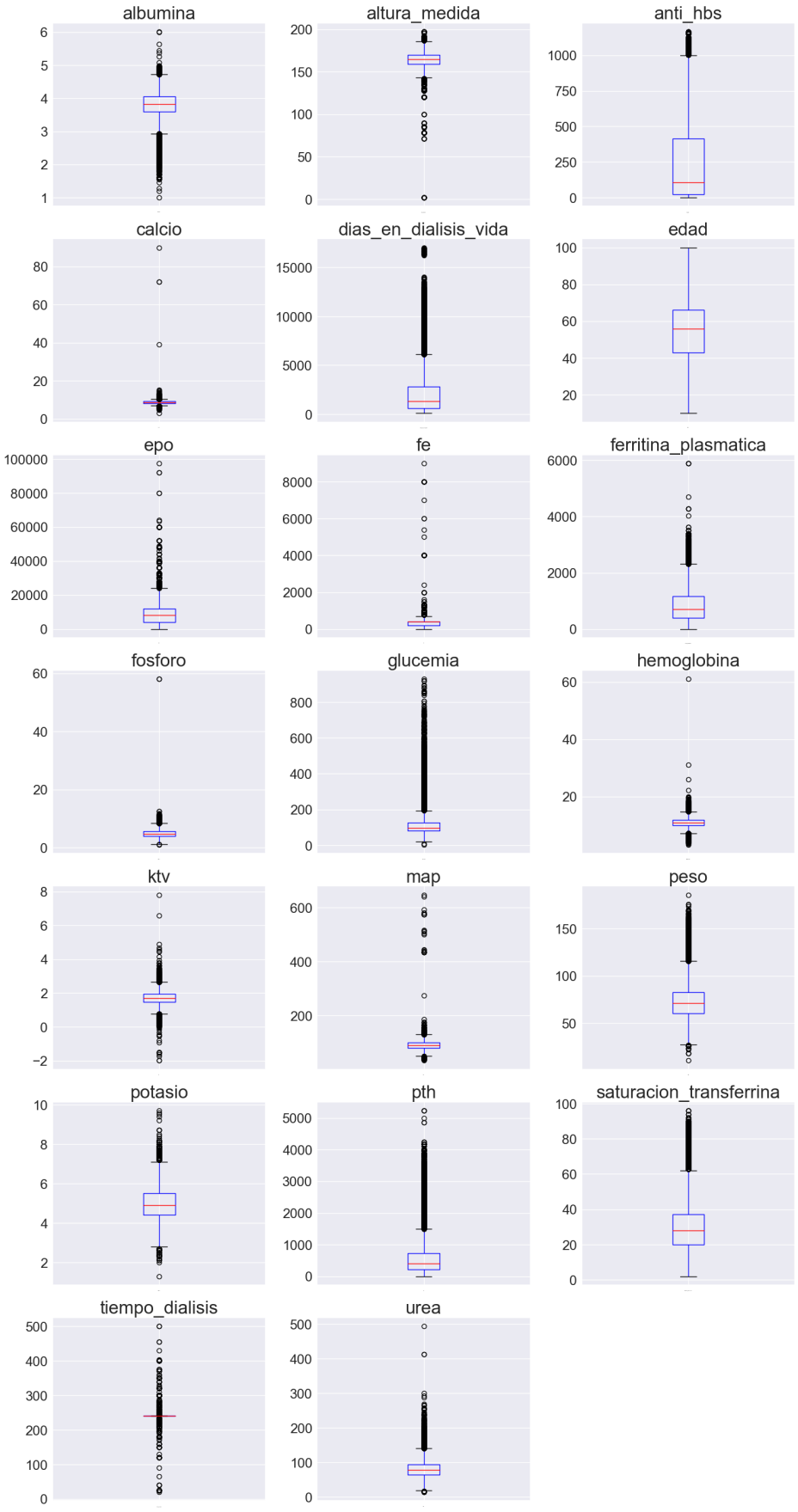

FIGURA 3.3. *Outliers* en variables numéricas de estrategia 2.

TABLA 3.1. Porcentaje de valores faltantes en variables numéricas eliminadas.

Variable	Estrategia 1	Estrategia 2
Hemoglobina glicosilada	94,8893 %	90,0596 %
Vitamina D 25oh	98,7345 %	98,5357 %
Bicarbonato	68,2404 %	28,3737 %

- Variables con poca cantidad de datos faltantes: puntualmente para las variables urea pre y urea post, se aplicó un promedio entre ellas para dejar un único valor(en el caso de que alguna de las dos no existiera se completó con el valor de la otra). Para el resto se completaron los valores faltantes con la técnica de imputación KNN. En la tabla 3.2 se muestran los porcentajes de valores faltantes en cada variable para las dos estrategias.

TABLA 3.2. Porcentaje de valores faltantes en variables numéricas.

Variable	Estrategia 1	Estrategia 2
Edad	0,0365 %	0,0052 %
Altura	0,0608 %	0,0000 %
Peso	0,1825 %	0,0026 %
Tiempo de Diálisis	0,1339 %	0,0155 %
MAP	0,0852 %	0,0013 %
Hemoglobina	0,1095 %	0,0155 %
Potasio	6,1207 %	0,0116 %
Glucemia	9,9416 %	0,0142 %
Calcio	0,0365 %	0,0013 %
Fósforo	0,0487 %	0,0000 %
Albumina	0,1339 %	0,0039 %
Saturación transferrina	3,1638 %	0,7534 %
Ferritina plasmática	0,7544 %	0,1187 %
Pth	2,8961 %	2,3596 %
Anti-HBs	2,4337 %	0,7096 %
Kt/V	0,3164 %	0,0606 %

En cuanto a las distribuciones de las variables numéricas, si bien se realizaron pruebas aplicando transformaciones de Yeo Johnson, no se observaron mejoras en los resultados. Por tal motivo, no se incluyó ninguna transformación en la implementación final. En las figuras 3.4 y 3.5 se muestran las distribuciones en ambas estrategias.

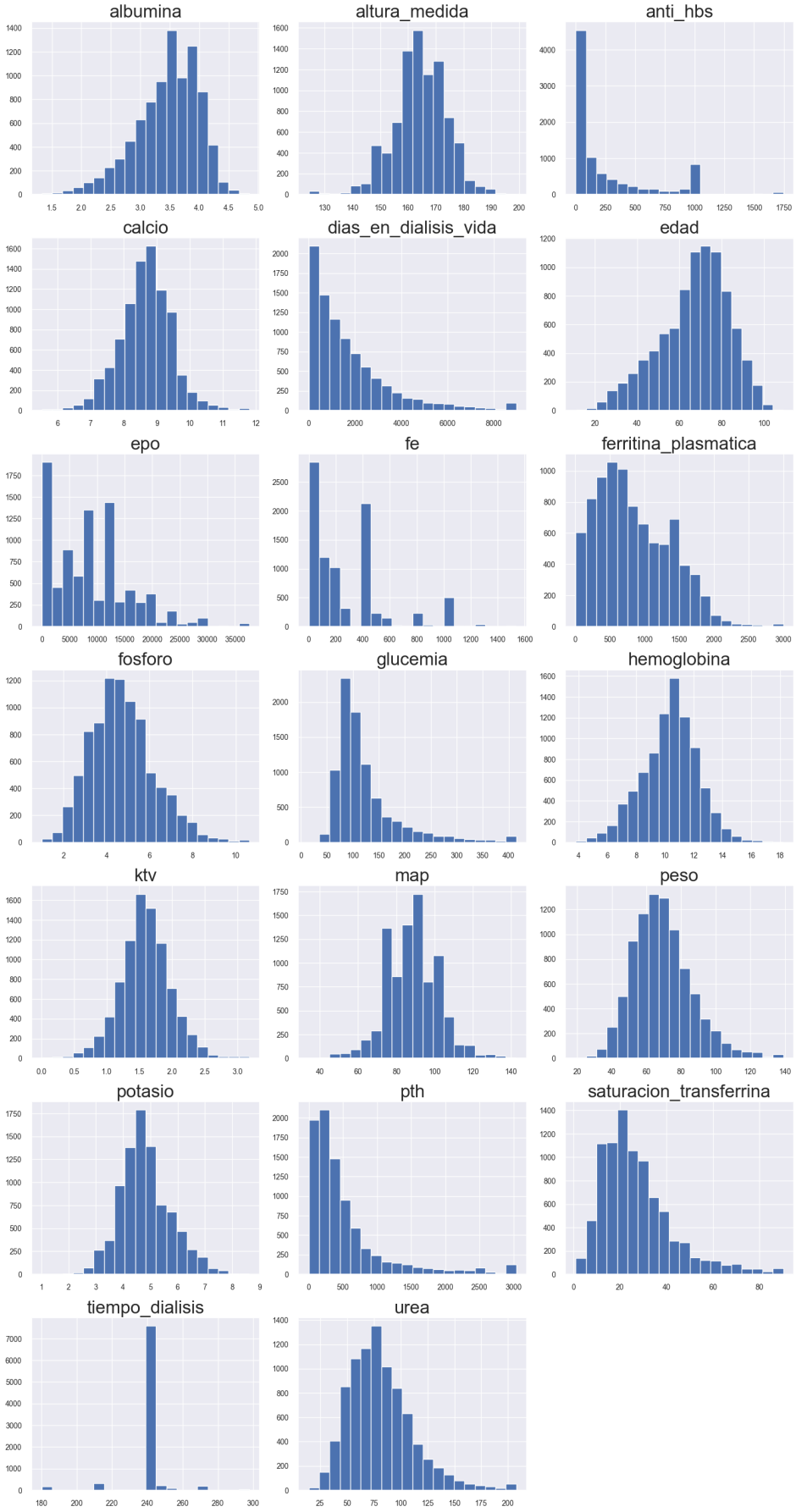

FIGURA 3.4. Distribuciones de las variables numéricas en la estrategia 1.

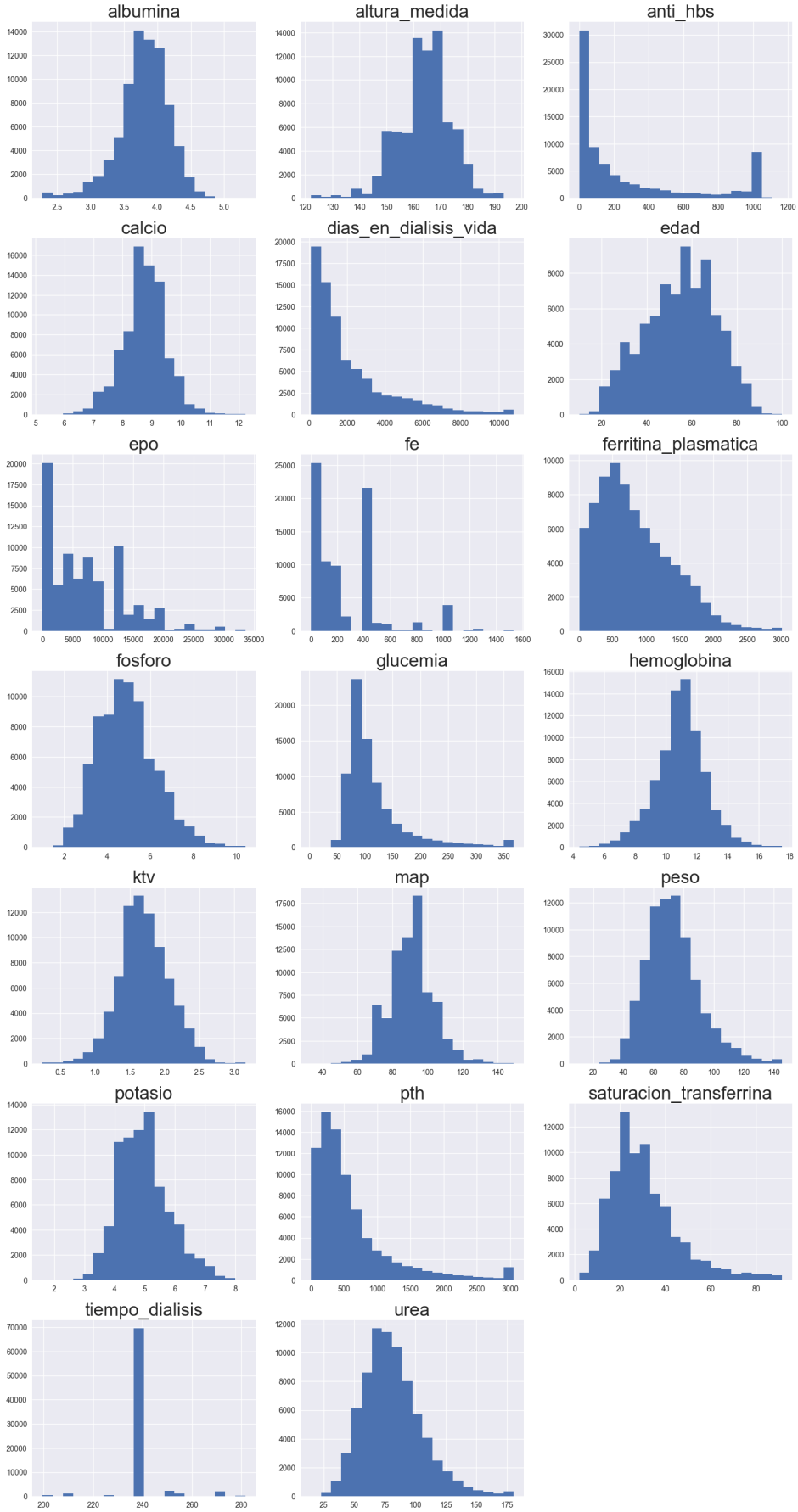

FIGURA 3.5. Distribuciones de las variables numéricas en la estrategia 2.

3.3.2. Preprocesamiento de variables categóricas

Para las variables categóricas se utilizó la técnica de codificación *one-hot*, que crea nuevas variables binarias para cada categoría. Se decidió solo crear nuevas variables para las categorías que representaban más del 10 % del total de datos. Esta técnica también resuelve el problema de los datos faltantes en estas variables, ya que en dicho caso se colocará el valor 0 (o *false*) en todas las variables binarias del grupo para esa observación. En la tabla 3.3 se muestra el porcentaje de valores faltantes en las variable categórica en cada estrategia.

TABLA 3.3. Porcentaje de valores faltantes en variables categóricas.

Variable	Estrategia 1	Estrategia 2
Anti-HIV	0,4016 %	0,0426 %
Anti-HCV	0,6814 %	0,1780 %
HBSAg	0,4016 %	0,0542 %
HCV PCR	98,2599 %	98,7060 %
HBSAg PCR	99,0995 %	99,3420 %
Acceso vascular	0,3164 %	0,0000 %
Grupo sanguíneo	56,3154 %	63,6521 %
Lista de espera para trasplante	34,5461 %	27,0758 %

3.3.3. Normalización de los datos

Para optimizar los tiempos de entrenamiento del modelo, se aplica una normalización a todas las variables del conjunto de datos. Luego de probar distintas técnicas, se elije la de *z-score*, que usa la media y el desvío estándar. Este proceso transforma todas las variables del conjunto de datos para que tengan un rango similar.

Una vez aplicados todos los puntos anteriores, el conjunto de datos se encuentra completo y posee únicamente valores numéricos, lo que ayuda a los modelos a poder aprender de los datos y generalizar el conocimiento. Las cantidades de variables numéricas que reciben los modelos para realizar el entrenamiento usando las distintas estrategias son:

- Estrategia 1: 53 variables.
- Estrategia 2: 56 variables.

Las 3 variables que solo existen en la estrategia 2 refieren a clases dentro de variables categóricas que en la estrategia 1 no tenían suficientes datos (más del 10 %) y se unificaron con un tipo llamado “otros” durante la codificación *one hot*.

Todos los puntos del preprocesamiento mencionados anteriormente se aplicaron tanto al conjunto de entrenamiento, como al conjunto de pruebas y de validación. Se utilizaron los mismos estimadores del entrenamiento (*capping*, imputador de valores faltantes y normalización) para transformar los conjuntos de pruebas y validación.

3.4. Estrategias aplicadas para balancear las clases

En los conjuntos de datos de las dos estrategias se cuenta con un gran desbalance entre las clases. En la tabla 3.4 se muestra la proporción (N_clase 0 / N_clase 1) de las clases en cada conjunto de datos.

TABLA 3.4. Desbalance de pacientes por estrategia.

Estrategia	Total de Pacientes	No Fallecidos	Fallecidos	Proporción
Estrategia 1	11 961	4340	7621	0,56
Estrategia 2	284 155	279 686	4469	62,58

Los modelos de *Machine Learning* y *Deep Learning* suelen tener problemas al intentar aprender de conjuntos de datos desbalanceados, por lo que resulta obligatorio realizar alguna acción para corregir esta situación. Para este trabajo se realizaron pruebas de entrenamientos utilizando submuestreo, con la técnica de submuestreo aleatorio y *Near Miss*, y sobremuestreo, con la técnica de SMOTE. Solo en la estrategia 1 se pudo aplicar SMOTE, lo que mejoró levemente la predicción en modelos de *Machine Learning*. En el caso de la estrategia 2, no se aplicó ninguna de las técnicas anteriores, ya que el desbalance era muy pronunciado. Para este caso se utilizó la técnica de *focal loss* aplicada sobre los modelos de *Deep Learning*, que implica utilizar una función de pérdida durante el entrenamiento del modelo que penalice más las predicciones incorrectas para los ejemplos difíciles. Esto significa que el modelo presta más atención a los casos de la clase minoritaria.

3.5. Diseño y desarrollo de modelos

Para este trabajo se probaron arquitecturas de modelos de *Machine Learning* y de *Deep Learning*. A continuación se presentan en detalle las arquitecturas utilizadas y las métricas que evaluaron su desempeño.

3.5.1. Modelo de *Machine Learning*

Se entrenaron varios tipos de modelos de *Machine Learning* para ambas estrategias utilizando una librería llamada PyCaret [21], que automatiza los entrenamientos de distintas arquitecturas. Los modelos entrenados fueron los siguientes: *Random forest*, regresión logística, árbol de decisión y SVM. Los resultados se describen en el capitulo 4.

Con el conjunto de datos de la estrategia 1 se lograron resultados aceptables, pero con los datos de la estrategia 2 se obtuvieron resultados muy por debajo de lo esperado. Por ese motivo se decidió continuar entrenando modelos de ML solo con el conjunto de datos de la estrategia 1.

Se eligió el modelo *Random forest* principalmente por haber obtenido los mejores resultados. Pero también se lo elije por tener un hiperparámetro llamado *class_weight* que permite darle más peso a la clase minoritaria durante el entrenamiento, lo que resulta de mucha utilidad al trabajar con un conjunto de datos desbalanceado.

3.5.2. Modelo de *Deep Learning*

Como modelo de *Deep Learning* se diseñó una red neuronal multicapa totalmente conectada (*fully connected*) que devuelva un único valor binario. Al no contar con conocimientos previos sobre el diseño de una red neuronal óptima para resolver un problema de este tipo, se probaron distintas arquitecturas para determinar cuál proporcionaba los mejores resultados.

En la tabla 3.5 se muestran las diferentes combinaciones de modelos que se probaron en esta exploración. Los valores indican la cantidad de neuronas en cada capa.

TABLA 3.5. Comparación de modelos con diferentes configuraciones de capas ocultas.

Modelo	**Capa 1**	**Capa 2**	**Capa 3**
1	200		
2	500		
3	1000		
4	200	100	
5	500	200	
6	1000	500	
7	500	200	100
8	1000	500	100

En el capitulo 4 se describen los resultados de las pruebas realizadas con las distintas arquitecturas de redes neuronales.

A diferencia de los ocurrido con los modelos de ML, los resultados obtenidos con el conjunto de datos de la estrategia 1 fueron inferiores a los obtenidos con el conjunto de datos de la estrategia 2. Por tal motivo, se continuó utilizando el segundo conjunto únicamente para entrenar modelos de DL.

Se decidió seleccionar el modelo 6 que está conformado por 2 capas ocultas densamente conectadas. La capa de entrada recibe 56 características y se conectan con la primer capa oculta que tiene 1000 neuronas con función de activación ReLU (*Rectified Linear Unit*). Posteriormente, hay una segunda capa oculta con 500 neuronas que también utilizan la función de activación ReLU. La capa de salida consta de una sola neurona, dado que el modelo devuelve un resultado binario. En la figura 3.6 se muestra la estructura de la red neuronal.

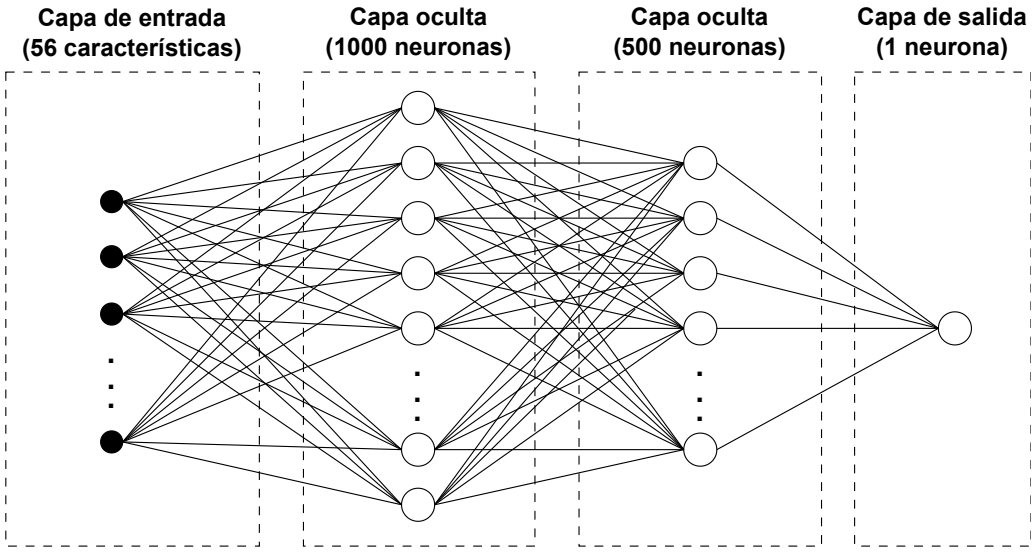

FIGURA 3.6. Estructura de la red neuronal.

3.5.3. Selección de funciones de activación, función de pérdida y algoritmo de optimización

Se seleccionó la función de activación ReLU para cada una de las capas ocultas de la red neuronal. Esta función es comúnmente utilizada por ser muy simple, ya que devuelve cero para valores negativos y el mismo valor para valores positivos. Esto significa que no requiere cálculos complejos como las funciones sigmoidales o tangenciales hiperbólicas, lo que puede hacer que el entrenamiento de la red neuronal sea más rápido y eficiente computacionalmente. Como función de pérdida, se decidió utilizar *focal loss*, ya que fue diseñada específicamente para abordar el problema del desequilibrio de clases en problemas de clasificación. En cuanto al algoritmo de optimización se eligió Adam, que se destaca por su adaptabilidad y eficiencia computacional. Combina conceptos como el *momentum* para acelerar la convergencia y la adaptación de la tasa de aprendizaje para cada parámetro, lo que lo hace especialmente efectivo en problemas con gradientes variables. Además, su regularización implícita y su capacidad para manejar una amplia gama de aplicaciones lo convierten en una opción popular y confiable para el entrenamiento de redes neuronales.

3.5.4. Estrategias implementadas para evitar el *overfitting*

El *overfitting*, o sobreajuste, es un fenómeno en el aprendizaje automático donde un modelo se adapta demasiado bien a los datos de entrenamiento, y captura no solo la relación subyacente entre las características y las etiquetas, sino también el ruido aleatorio o las peculiaridades específicas del conjunto de entrenamiento. En otras palabras, el modelo memoriza los datos de entrenamiento. Esto resulta en un rendimiento deficiente cuando el modelo se enfrenta a datos nuevos, nunca antes vistos, lo que le impide generalizar correctamente. Para minimizar el *overfitting* y mejorar la robustez del modelo, se realizó lo siguiente:

- Mantener el modelo lo más simple posible, sin agregar capas o neuronas innecesarias.

- Utilizar las características mínimas que proporcionen información relevante al modelo para poder aprender y generalizar el conocimiento.
- Agregar regularización *Ridge* (L2) que penaliza los valores grandes de los pesos, para que reduzcan su magnitud durante el entrenamiento.
- Utilizar *early stopping* para detener el proceso de entrenamiento al detectar un aumento del error en validación.
- Utilizar un planificador de tasa de aprendizaje para ir disminuyendo el *learning rate* a medida que pasan las épocas. Esto permite hacer un ajuste más fino en las últimas épocas de entrenamiento.

3.6. Conjunto de herramientas para acceso al modelo

Como parte del objetivo de este trabajo, se configuraron y desarrollaron una serie de herramientas que permiten hacer uso del modelo para obtener predicciones.

En primer lugar, se configuró la plataforma Jenkins para desplegar el modelo en distintos ambientes. Al trabajar con GIT y tener un repositorio en la nube, se puede configurar esta plataforma para que descargue la ultima versión del modelo entrenado y lo despliegue en algún servidor.

También se desarrolló una API para el acceso al modelo que permite a una aplicación cliente solicitar una o varias predicciones juntas al modelo predictivo. La API fue desarrollada con arquitectura RESTful en lenguaje Python mediante la librería FastAPI [22]. Se desarrollaron los siguientes métodos:

- `POST /predict`: este método recibe por parámetro un archivo con extensión CSV (*comma separated values*) con uno o varios registros. El archivo contiene todas las variables necesarias para realizar la predicción más un identificador del paciente, que sirve para asociar la predicción devuelta. El método devuelve un archivo con las predicciones asociadas a cada paciente, que incluye la probabilidad de cada clase.
- `POST /predict_single`: este método recibe por parámetro los valores de cada una de las variables y devuelve una predicción individual, que incluye la probabilidad de cada clase.
- `POST /evaluate`: este método recibe por parámetro un archivo con extensión CSV con uno o varios registros. El archivo contiene todas las variables necesarias para realizar la predicción e incluye una variable que indica si el paciente falleció o no. Con los datos recibidos se generan las predicciones por cada paciente y se comparan con los datos reales. Este método devuelve las métricas *f1 score*, *accuracy*, *precision*, *recall* y los 4 valores de la matriz de confusión. Estos resultados permiten evaluar el rendimiento del modelo sobre nuevos conjuntos de validación.

Los tres métodos cargan un modelo entrenado y realizan la ingeniería de características, esto quiere decir que se aplica todo el preprocesamiento de datos realizado durante el entrenamiento del modelo sobre los nuevos datos recibidos. Una vez que los datos tengan el formato correcto, se computan sobre el modelo para obtener las predicciones como respuesta. La misma plataforma utilizada para desplegar el modelo también se pudo configurar para que despliegue automáticamente la última versión de la API cuando sea necesario.

Por último, se desarrolló una aplicación de consola en lenguaje C#. Esta se encarga de realizar una consulta a la base de datos de la organización médica para recuperar el listado de pacientes activos a ese momento, que hayan estado en tratamiento de hemodiálisis por más de 90 días. Con el resultado obtenido genera un archivo en formato CSV y realiza una llamada a la API utilizando el método `/predict`. Este método devuelve las predicciones de cada paciente y estas se graban en una nueva tabla de la base de datos. Esta aplicación se ejecutaría una vez por día para mantener actualizada la información sobre el riesgo de mortalidad de los pacientes activos. Estas predicciones se muestran al usuario final en formato de reporte dentro de la aplicación de gestión de pacientes, donde pueden exportarla en formato de planilla de cálculos.

Capítulo 4

Ensayos y resultados

En este capítulo se exponen los resultados obtenidos de los entrenamientos de los modelos de IA utilizados. Se comentarán los resultados de los modelos de *Machine Learning* y *Deep Learning* y se indicará cuál fue el modelo elegido.

4.1. Resultados de los modelos

Una vez que se realizó el preprocesamiento tanto a los datos de entrenamiento como a los datos de prueba y validación, se entrenaron múltiples modelos de distintas arquitecturas.

Al tratarse de una predicción de riesgo de mortalidad, la empresa médica solicitó priorizar los resultados que disminuyan los falsos negativos y aumenten la métrica de *recall*, teniendo en cuenta que impactaría negativamente en la métrica de *precision* y *F1 score*.

A continuación se presentan los resultados obtenidos.

4.2. Resultados en modelos de *Machine Learning*

Como se explicó en el capitulo anterior, se entrenaron múltiples modelos de *Machine Learning*. En las tablas 4.1 y 4.2 se muestran los resultados de los entrenamientos con los conjuntos de datos de ambas estrategias y evaluados con el conjunto de validación.

TABLA 4.1. Resultados de modelos de ML entrenados con datos de la estrategia 1.

Modelo	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**
Random forest classifier	0,7987	0,0427	0,8333	0,0813	0,8158
Logistic regression	0,7513	0,0255	0,6000	0,0490	0,6764
Decision tree classifier	0,7078	0,0297	0,8333	0,0574	0,7699
SVM - Linear kernel	0,3633	0,0132	0,8000	0,0261	0,5793

TABLA 4.2. Resultados de modelos de ML entrenados con datos de la estrategia 2.

Modelo	Accuracy	Precision	Recall	F1 Score	AUC
Random forest classifier	0,9889	0,0000	0,0000	0,0000	0,4998
Logistic regression	0,9879	0,0000	0,0000	0,0000	0,4992
Decision tree classifier	0,9634	0,0602	0,1666	0,0884	0,5693
SVM - Linear kernel	0,9733	0,0212	0,0333	0,0259	0,5084

Como se puede observar en las tablas, solo se obtuvieron buenos resultados utilizando el conjunto de datos de la estrategia 1. El modelo seleccionado fue el *Random forest*, y se logró el mejor resultado ajustando los pesos de cada clase junto con la aplicación de SMOTE, para sobremuestrear la clase minoritaria (clase 0, no fallecidos).

Se utilizó el modelo *RandomForestClassifier* de la librería *Sckit-Learn* [23] y se entrenó con los siguientes hiperparámetros:

- Cantidad de Estimadores: 88
- Profundidad máxima de cada árbol: 34
- Pesos de las clases: No fallecidos: 105,29473000000468, Fallecidos: 1

Los hiperparámetros no mencionados se dejaron con su valor por defecto según lo indicado en la documentación del modelo [24]. El resto se obtuvieron realizando una búsqueda aleatoria con distintas configuraciones.

En la figura 4.1 se muestra la matriz de confusión y en la figura 4.2 se muestra la curva ROC.

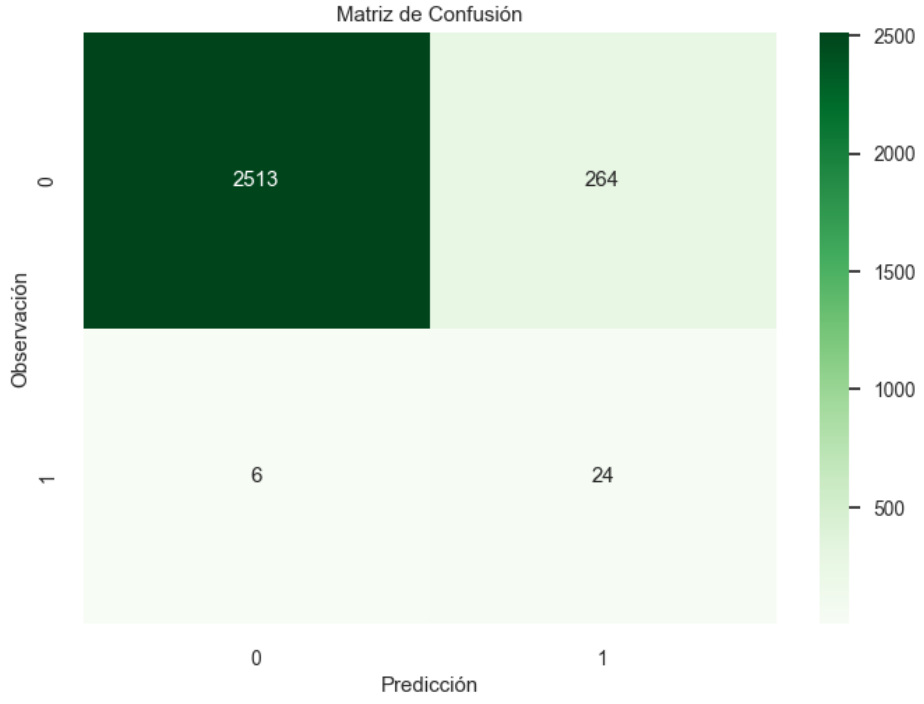

FIGURA 4.1. Matriz de Confusión del modelo *Random forest*.

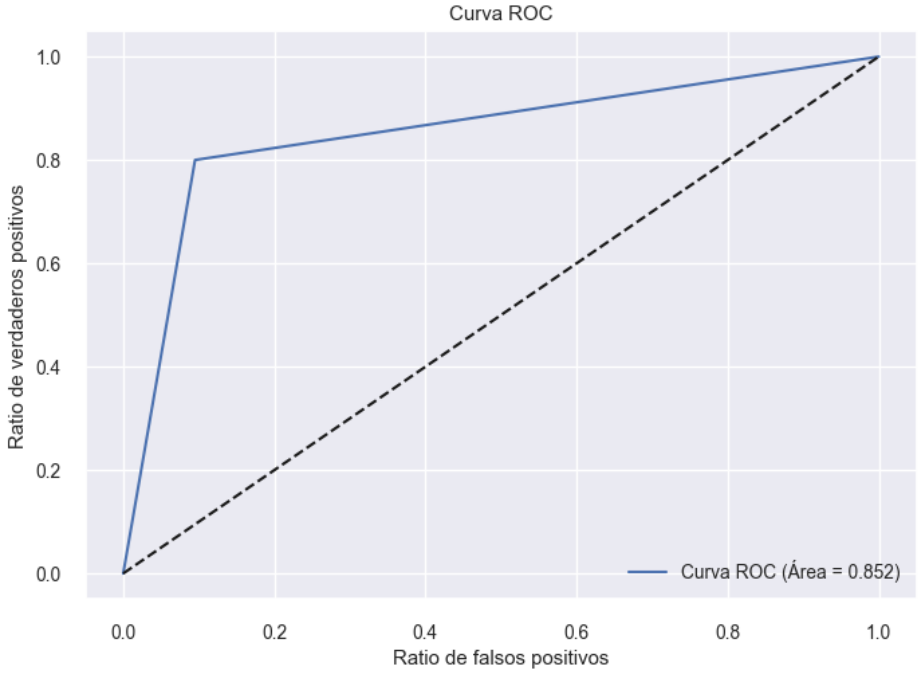

FIGURA 4.2. Curva ROC del modelo *Random forest*.

Las métricas logradas fueron las siguientes:

- *F1 Score* = 0,1509
- *Accuracy*= 0,9038
- *Precision*= 0,0833
- *Recall*= 0,8000
- *AUC*= 0,852

El modelo exhibe una precisión general sólida del 90,38 % (*accuracy*), y destaca su capacidad para realizar la mayoría de las predicciones correctamente. Además, el elevado *recall* del 80 % indica que el modelo identifica correctamente la gran mayoría de los casos de mortalidad. La curva ROC muestra una capacidad notable del modelo para distinguir entre pacientes con y sin riesgo, con un área bajo la curva (AUC) de 0,852, lo que sugiere una habilidad significativa para realizar esta discriminación.

Del entrenamiento de este modelo puede desprenderse información que puede ser sumamente valiosa para el personal médico, como es la importancia de las características a la hora de realizar las predicciones. El modelo *Random Forest* calcula automáticamente la importancia de las características basándose en la disminución media de la impureza (*Mean decrease impurity* o MDI). Esta se calcula como la media y la desviación estándar de la acumulación de la disminución de impurezas dentro de cada árbol. En la figura 4.3 se observa la importancia de las 15 características más relevantes del conjunto de datos de la estrategia 1 utilizando la técnica MDI.

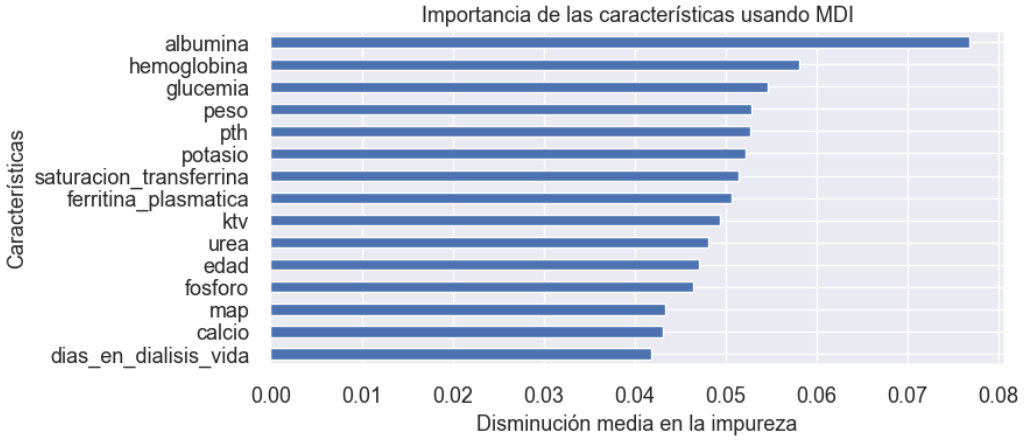

FIGURA 4.3. Importancia de las características más relevantes para el modelo de ML.

4.3. Resultados en modelos de *Deep Learning*

En cuanto a los modelos de *Deep Learning*, en la subsección 3.5.2 se definieron distintas arquitecturas de redes neuronales para identificar cuál logra mejores resultados.

En la tabla 4.3 se muestran los valores de las diferentes métricas obtenidos en los entrenamientos de los modelos con el conjunto de datos de la estrategia 1 y evaluados con el conjunto de validación.

TABLA 4.3. Resultados de diferentes modelos con el conjunto de datos de la estrategia 1.

Modelo	Accuracy	Precision	Recall	F1 Score	AUC	Épocas
1	0,5963	0,0167	0,6333	0,0324	0,6150	100
2	0,6149	0,0183	0,6667	0,0356	0,6400	100
3	0,6270	0,0189	0,6666	0,0367	0,6470	100
4	0,6387	0,0223	0,7666	0,0433	0,7020	100
5	0,6451	0,0217	0,7333	0,0423	0,6890	66
6	0,6579	0,0216	0,7000	0,0419	0,6790	29
7	0,6494	0,0210	0,7000	0,0409	0,6740	47
8	0,7028	0,0236	0,6666	0,0457	0,6850	27

En los entrenamientos con el conjunto de datos de la estrategia 1 se utilizaron los hiperparámetros indicados en la tabla 4.4. Si bien todos los modelos se configuraron para entrenar durante 100 épocas, se utilizó *early stopping*, por lo que algunos se detuvieron antes al notar que ya no disminuía la pérdida.

TABLA 4.4. Hiperparámetros utilizados durante los entrenamientos con el conjunto de datos de la estrategia 1.

Hiperparámetro	**Valor**
Número de épocas de entrenamiento	100
Función de activación de capas ocultas	Relu
Función de pérdida (*focal loss*). Parámetro Gamma	7
Función de pérdida (*focal loss*). Parámetro Alpha	0,58
Optimizador	Adam
Learning rate	0,00005
Tipo de algoritmo de entrenamiento	*Mini-batch*
Tamaño del *batch*	2048
Early stopping. Paciencia	5 épocas

En la tabla 4.5 se muestran los valores de las diferentes métricas obtenidos en los entrenamientos de los modelos con el conjunto de datos de la estrategia 2 y evaluados con el conjunto de validación.

TABLA 4.5. Resultados de diferentes modelos con el conjunto de datos de la estrategia 2.

Modelo	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**	**Épocas**
1	0,6128	0,0225	0,8333	0,0438	0,7220	61
2	0,6295	0,0225	0,8000	0,0440	0,7140	45
3	0,6192	0,0228	0,8333	0,0445	0,7250	45
4	0,6110	0,0233	0,8666	0,0453	0,7370	32
5	0,6366	0,0221	0,7666	0,0430	0,7010	19
6	0,6941	0,0272	0,8000	0,0527	0,7460	28
7	0,6515	0,0240	0,8000	0,0466	0,7250	25
8	0,6969	0,0264	0,7666	0,0511	0,7310	21

En los entrenamientos con el conjunto de datos de la estrategia 2 se utilizaron los hiperparámetros indicados en la tabla 4.6. También se utilizó *early stopping*.

TABLA 4.6. Hiperparámetros utilizados durante los entrenamientos con el conjunto de datos de la estrategia 2.

Hiperparámetro	**Valor**
Número de épocas de entrenamiento	100
Función de activación de capas ocultas	Relu
Función de pérdida (*focal loss*). Parámetro Gamma	7
Función de pérdida (*focal loss*). Parámetro Alpha	0,984
Optimizador	Adam
Learning rate	0,00005
Tipo de algoritmo de entrenamiento	*Mini-batch*
Tamaño del *batch*	8192
Early stopping. Paciencia	5 épocas

Las redes neuronales entrenadas con los datos de la estrategia 1 no llegaron a buenos resultados, por lo que solo se utilizó el conjunto de datos de la estrategia 2 que contenía mayor cantidad de datos, aunque la desproporción entre clases también era mayor. De los modelos comparados, todos llegaron a resultados similares con un *accuracy* entre 61 y 69 % y un *recall* entre 76 y 86 %. Se utilizó el modelo 6, compuesto por dos capas ocultas de 1000 y 500 neuronas respectivamente, y se usó como función de pérdida la *focal loss*. Esto ayudó mucho a compensar el desbalance de las clases y permitió al modelo dar resultados aceptables. Se realizó una búsqueda aleatoria para encontrar los hiperparámetros que logren el mejor rendimiento del modelo. De esta forma se pudo lograr métricas que cumplan con el objetivo del trabajo.

En la figura 4.4 se observa la matriz de confusión y en la figura 4.5 se muestra la curva ROC.

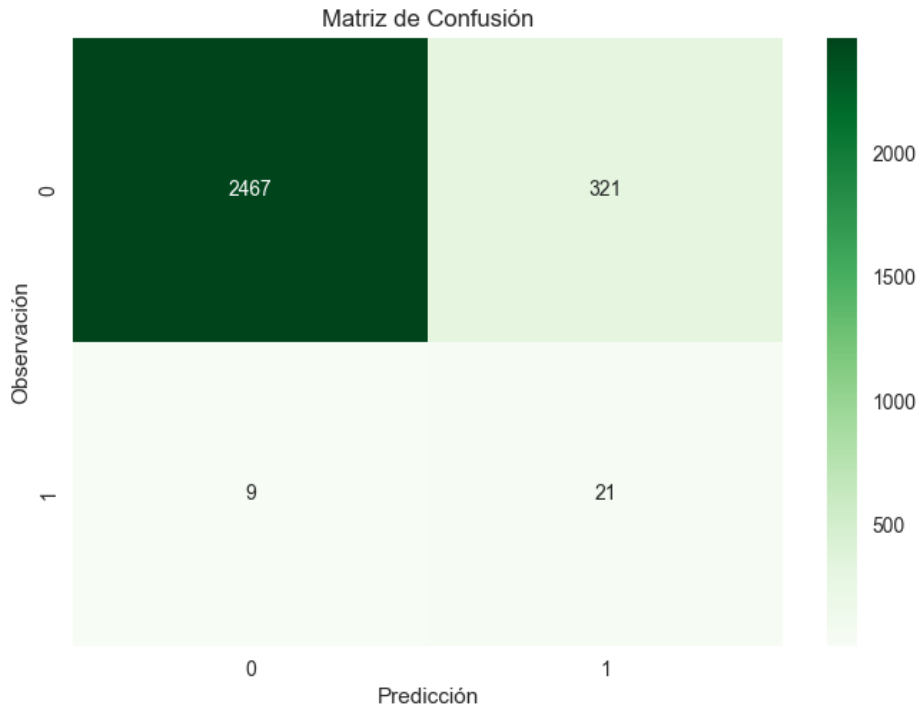

FIGURA 4.4. Matriz de Confusión de la red neuronal.

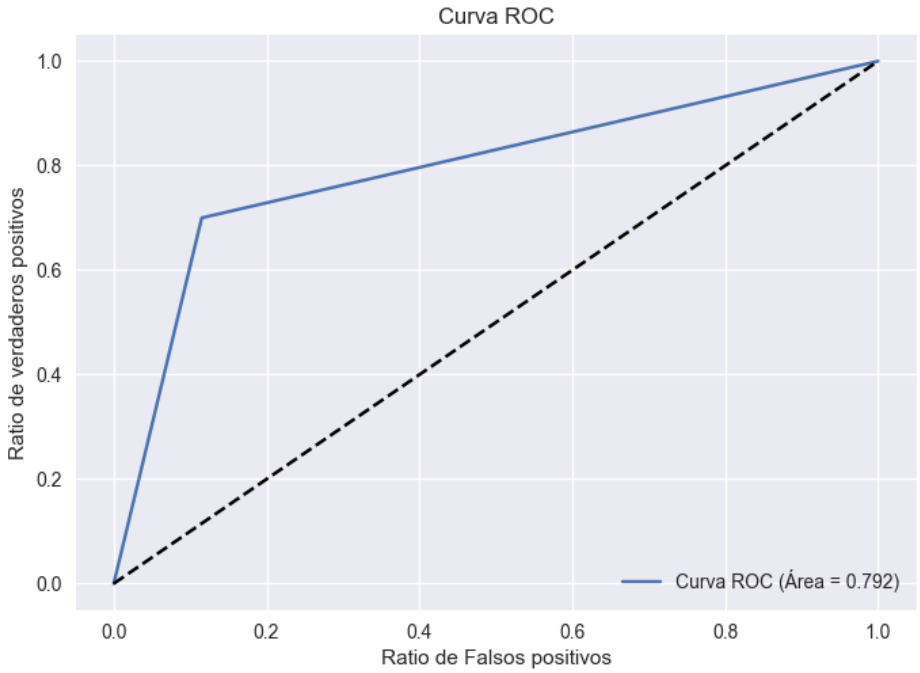

FIGURA 4.5. Curva ROC de la red neuronal.

A continuación, se enumeran las métricas obtenidas con el conjunto de validación sobre el modelo seleccionado descrito en el punto 3.5.2, luego de 32 épocas de entrenamiento:

- *F1 Score*= 0,1129
- *Accuracy*= 0,8828
- *Precision*= 0,0614
- *Recall*= 0,7000
- *AUC*= 0,792

Los hiperparámetros utilizados en el modelo final se muestran en la tabla 4.7.

TABLA 4.7. Hiperparámetros utilizados en el modelo final.

Hiperparámetro	**Valor**
Número de épocas de entrenamiento	32
Función de activación de capas ocultas	Relu
Función de pérdida (*focal loss*). Parámetro gamma	6,834666033630177
Función de pérdida (*focal loss*). Parámetro alpha	0,9442629243520555
Optimizador	Adam
Learning rate inicial	0,000005
Planificador de *Learning rate*. Parámetro épocas	30
Planificador de *Learning rate*. Parámetro gamma	0,8
Lambda de regularización L2	0,001
Tipo de algoritmo de entrenamiento	*Mini-batch*
Tamaño del *batch*	256
Early stopping. Parámetro paciencia	20 épocas

El modelo presenta un *accuracy* del 88,28 %, lo que indica un rendimiento aceptable en la mayoría de las predicciones. Sin embargo, su baja *precision* (6,14 %) sugiere que muchas de las predicciones positivas son falsas. Por otro lado, un *recall* de 70 % indica que el modelo identifica a la mayoría de los casos con alto riesgo. El AUC de 0,792 indica una capacidad moderada para discriminar entre pacientes con riesgo y sin riesgo.

La técnica MDI utilizada en el modelo *Random Forest* para calcular la importancia de las características no se puede aplicar a modelos de redes neuronales. En este caso se utiliza la técnica de permutación de características, donde estas se barajan n veces y el modelo se reajusta para estimar su importancia. En la figura 4.6 se presenta la importancia de las 15 características más relevantes en el valor predicho por el modelo de DL con el conjunto de datos de la estrategia 2, utilizando la técnica de permutación.

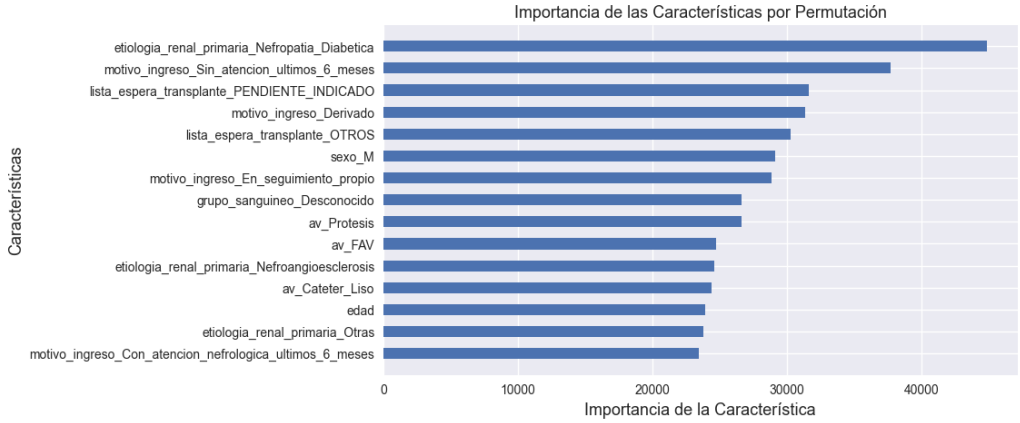

FIGURA 4.6. Importancia de las características más relevantes para el modelo de DL.

4.4. Modelo seleccionado

Si bien ambos modelos superan el requisito de obtener un *accuracy* por encima del 75 %, el modelo de ML logra valores de *recall*, *precision* y *F1 Score* más altos, lo que es fundamental en este tipo de problemas. Por este motivo se selecciona como modelo definitivo el *Random forest*. Este modelo es mucho más simple, requiere menos datos para entrenarse y menor poder computacional, lo que hará más sencillo realizar nuevos entrenamientos en un futuro.

Capítulo 5

Conclusiones

En este capítulo se presentan las conclusiones y los aportes realizados por este trabajo. También, se detallan futuras líneas de investigación para mejorar los resultados.

5.1. Conclusiones generales

A continuación, se resaltan las conclusiones y los objetivos cumplidos con el desarrollo de este trabajo:

- Se logró implementar un modelo de aprendizaje profundo que realiza predicciones de mortalidad en pacientes en tratamiento de diálisis renal.
- El trabajo cumple con todos los requisitos principales definidos en la subsección 1.3.3.
- La desproporción de clases en los conjuntos de datos presentó un problema difícil de resolver que limitó la capacidad predictiva del modelo.
- Se pudo seguir el cronograma establecido sin mayores inconvenientes.

De los riesgos identificados para este proyecto se tuvieron que ejecutar algunas acciones para mitigar dos de ellos:

- Riesgo 1: no disponer de un conjunto de datos adecuado para el entrenamiento del modelo. En este caso las acciones propuestas en el plan de proyecto no fueron suficientes para mitigar el problema, por lo que se tuvo que realizar una investigación más exhaustiva para encontrar mejores soluciones.
- Riesgo 2: no lograr que el modelo entrenado realice predicciones correctas. Para mitigar el riesgo se invirtió mucho más tiempo del planificado en realizar entrenamientos de modelos y en investigar sobre arquitecturas y estrategias que sean útiles para resolver este tipo de problemas.

5.2. Próximos pasos

Para realizar este trabajo se utilizó la información clínica disponible de los pacientes que realizaron tratamientos de diálisis renal. Esto llevó a la recuperación de algunos datos inconsistentes y gran cantidad de datos faltantes. La calidad de los datos utilizada para entrenar los modelos podría mejorarse desde la etapa de registración, incluso definiendo algunos datos como obligatorios para que

sean registrados en todos los pacientes. Esto ayudaría a contar con información valiosa para entrenar nuevos modelos de predicción en un futuro.

Sería interesante explorar nuevas arquitecturas de modelos que sirvan para resolver problemas de este tipo, donde existe tanta desproporción en los datos. También se podrían realizar nuevos conjuntos de datos desde una perspectiva distinta, que capturen la evolución de las distintas variables clínicas y de laboratorio que den nueva información al modelo para realizar la predicción.

Bibliografía

[1] Peñaranda-Florez Devi Geesel
Torrado-Navarro Yoryely Pereira-Rodríguez Javier Boada-Morales Lorena. «Dialisis y hemodialisis. Una revisión actual según la evidencia». En: *Journal* (2017). URL: https://www.nefrologiaargentina.org.ar/numeros/2017/volumen15_2/articulo2.pdf (visitado 11-03-2024).

[2] Mónica Valeria Cortés Badilla Marianella Álvarez Vega Laura María Quirós Mora. «Inteligencia artificial y aprendizaje automático en medicina». En: *Journal* (2020). URL: https://www.revistamedicasinergia.com/index.php/rms/article/view/557/923 (visitado 11-03-2024).

[3] Victoria Eugenia García Montemayor. «Mortalidad en pacientes en diálisis: Importancia y desarrollo de nuevos métodos fiables de predicción». En: *Journal* (2021). URL: https://helvia.uco.es/bitstream/handle/10396/22708/2022000002402.pdf?sequence=1&isAllowed=y (visitado 11-03-2024).

[4] Beatriz Ricardo Paez Sergio Orlando Escalona González Zoraida Caridad González Milán. «Predicción de mortalidad en pacientes con enfermedad renal crónica mediante el uso de la inteligencia artificial». En: *Journal* (2022). URL: https://convencionsalud.sld.cu/index.php/convencionsalud22/2022/paper/viewFile/44/167 (visitado 11-03-2024).

[5] Br. Shirley Aracelly Herrera Arce. «Índice neutrófilo/linfocito como predictor de mortalidad en pacientes que inician hemodiálisis». En: *Journal* (2022). URL:
https://repositorio.upao.edu.pe/bitstream/handle/20.500.12759/9293/REP_SHIRLEY.HERRERA_INDICE.NEUTROFILO.pdf?sequence=1
(visitado 11-03-2024).

[6] Mengqin Zhang Xing Chen Jun Zhang Jiyi Huang
Lu Zhang Lijing Yao Hengyuan Zhang. «Application of artificial intelligence in renal disease». En: *Journal* (2022). URL:
https://www.sciencedirect.com/science/article/pii/S2588914121000083
(visitado 11-03-2024).

[7] Charat Thongprayoon Pattharawin Pattharanitima
Wisit Cheungpasitporn Pajaree Krisanapan Supawit Tangpanithandee. «Revolutionizing Chronic Kidney Disease Management with Machine Learning and Artificial Intelligence». En: *Journal* (2023). URL:
https://www.mdpi.com/2077-0383/12/8/3018 (visitado 11-03-2024).

[8] Yingxue Li Tingyu Chen Xiang Li Zhihong Liu Guotong Xie Tiange Chen. «Artificial Intelligence in Nephrology: How Can Artificial Intelligence Augment Nephrologists' Intelligence?» En: *Journal* (2020). URL:
https://karger.com/kdd/article/6/1/1/186225 (visitado 11-03-2024).

[9] Yanxiang Gao Enmin Xie Xuecheng Zhao Ziyu Guo Yike Li Nan
Shen-Jingyi Ren Jingang Zheng Zixiang Ye Shuoyan An. «The prediction of in-hospital mortality in chronic kidney disease patients with coronary artery disease using machine learning models». En: *Journal* (2023). URL:

https://link.springer.com/article/10.1186/s40001-023-00995-x (visitado 11-03-2024).

[10] Florian Jungmann Christina Glasner Philipp Stenzel Stephanie Strobl Aurélie Fernandez Daniel-Christoph Wagner Axel Haferkamp Peter Mildenberger Wilfried Roth
Sebastian Foersch Stefan Schulz Ann-Christin Woerl. «Multimodal Deep Learning for Prognosis Prediction in Renal Cancer». En: *Journal* (2021). URL: https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2021.788740/full (visitado 11-03-2024).

[11] Jiří Kaiser. «Dealing with Missing Values in Data». En: *Journal* (2014). URL: https://www.researchgate.net/profile/Kamarul-Ariffin-Mansor-2/post/After_I_ran_EFA_should_I_use_Factor_scores_or_Mean_or_Summated_scales_in_a_multiple_regression_analysis/attachment/59d63ee3c49f478072ea9518/AS%3A273773436047361%401442284081197/download/178-697-1-PB+Handling+Missing+Vaues.pdf (visitado 18-03-2024).

[12] K.S.Vijaya Lakshmi B. Hemada. «A Study On Discretization Techniques». En: *Journal* (2013). URL: https://www.ijert.org/a-study-on-discretization-techniques (visitado 18-03-2024).

[13] Adrián Rocha Íñigo. «Codificación de variables categóricas en aprendizaje automático». En: *Journal* (2020). URL: https://idus.us.es/handle/11441/108887 (visitado 18-03-2024).

[14] Jonathan G. Kolo Suleiman O. E. Sadiku Abdullahi M. Orire Taliha A. Folorunso Abiodun M. Aibinu. «Effects of data normalization on water quality model in a recirculatory aquaculture system using artificial neural network». En: *Journal* (2019). URL: http://repository.futminna.edu.ng:8080/jspui/bitstream/123456789/2033/1/IJ9.pdf (visitado 02-04-2024).

[15] Malak Abdullah Roweida Mohammed Jumanah Rawashdeh. «Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results». En: *Journal* (2020). URL: https://www.researchgate.net/profile/Malak-Abdullah/publication/340978368_Machine_Learning_with_Oversampling_and_Undersampling_Techniques_Overview_Study_and_Experimental_Results/links/5ecd39764585152945121352/Machine-Learning-with-Oversampling-and-Undersampling-Techniques-Overview-Study-and-Experimental-Results.pdf (visitado 18-03-2024).

[16] R. Girshick K. He T. Y. Lin P. Goyal y P. Dollár. «Focal Loss for Dense Object Detection». En: *Journal* (2017). URL: https://ieeexplore.ieee.org/document/8237586 (visitado 02-04-2024).

[17] Lee Giles Rich Caruana Steve Lawrence. «Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping». En: *Journal* (2000). URL: https://proceedings.neurips.cc/paper/2000/file/059fdcd96baeb75112f09fa1dcc740cc-Paper.pdf (visitado 03-04-2024).

[18] MlFlow. *MlFlor WebSite*. https://mlflow.org/docs/latest/index.html. (Visitado 24-03-2024).

[19] Apache. *Apache Airflow WebSite*. https://airflow.apache.org/. (Visitado 24-03-2024).

[20] Jenkins. *Jenkins WebSite*. https://www.jenkins.io/. (Visitado 24-03-2024).

[21] Pycaret. *Pycaret WebSite*. https://pycaret.gitbook.io/docs. (Visitado 06-05-2024).

[22] FastAPI. *FastAPI WebSite*. https://fastapi.tiangolo.com/. (Visitado 04-04-2024).

[23] Scikit-learn. *Scikit-learn WebSite*. https://scikit-learn.org/stable/index.html. (Visitado 10-05-2024).

[24] Modelo RandomForestClassifier. *Documentación del modelo RandomForestClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. (Visitado 10-05-2024).