TRABAJO FINAL DE MÁSTER

Centro Europeo de Másteres y Posgrados

Predicción de diabetes de tipo 2 mediante varios algoritmos basados en aprendizaje automático o machine learning

Luis Merino Ulizarna

Índice

1. Resumen

En la actualidad, es una verdad completamente irrefutable el poder transformacional de la inteligencia artificial (IA). No sólo en los campos más populares en el momento de escribir estas líneas, finales del año 2023, como el procesamiento del lenguaje natural y el auge de los modelos generativos, como el indiscutible éxito de herramientas como ChatGPT, o en el campo de la generación de imágenes artificiales, en base a una descripción en lenguaje natural con modelos como DALL·E. Si no que este poder transformacional es totalmente horizontal a todos los sectores y problemas del ser humano, con una promesa específica en el ámbito de la salud.

No de forma casual, en el World Economic Forum de 2016, su fundador, Klaus Schwab, acuñó el término Cuarta revolución industrial. Según el sitio oficial, "*Se trata de un nuevo capítulo en el desarrollo humano, propiciado por extraordinarios avances tecnológicos equiparables a los de la primera, segunda y tercera revoluciones industriales. Estos avances están fusionando los mundos físico, digital y biológico de un modo que crea tanto enormes promesas como peligros potenciales*".[1]

En este contexto, uno de los campos con mayor potencial disruptivo es en el de salud, la atención hospitalaria y la investigación científica. Según la Organización Mundial de la Salud (OMS)[2], algunos de los beneficios que ofrece esta nueva tecnología son: mejorar la velocidad, así como la precisión del diagnóstico y la detección de enfermedades, favorecer la atención sanitaria, mejorar la respuesta a posibles brotes de enfermedades y la gestión de estos. También remarca la posibilidad de ofrecer a los pacientes mayor control sobre su salud, así como posibilitar el acceso a atención médica de calidad en países con menos recursos. Todo ello, no será exitoso sino se establecen estrategias y restricciones éticas en el uso de este tipo de sistemas.

Una de sus aplicaciones y principal motivación de este trabajo, es la detección precoz de enfermedades con el objetivo o bien de minimizar sus riesgos cuando se manifieste o bien predecir su aparición e intentar prevenir la misma. En concreto se presenta un trabajo que versa sobre la predicción de diabetes mellitus de tipo 2, como una forma de luchar contra el infradiagnóstico de dicha enfermedad.

Pues, según la Sociedad Española de Diabetes (SED), se estima que casi la mitad de los casos de diabetes en España no están diagnosticados[3]. Se expondrán varios enfoques basados en Machine Learning (ML) o Aprendizaje Automático (AA) con el objetivo de facilitar el diagnóstico de esta enfermedad. Puesto que España es el segundo país de Europa con mayor número de personas con diabetes[4].

La aplicación de estas nuevas técnicas basadas en tecnologías emergentes como la IA y el ML están justificadas bajo el pretexto de la medicina de precisión. Consistente en aplicar el tratamiento adecuado para el paciente adecuado en el momento adecuado. Según el presidente de la SED, "debemos avanzar en la denominada Medicina de

[1] *Fourth Industrial Revolution*, 2020
[2] *World Health Organization: WHO*, 2021
[3] Sociedad Española de Diabetes [SED], 21 abril 2023
[4] Sociedad Española de Diabetes, 19 abril 2023

Precisión, adaptando las recomendaciones a las características del paciente, a sus circunstancias y preferencias"

2. Introducción y objetivos

2.1 Contexto

La medicina personalizada se está imponiendo ahora mismo como paradigma de la atención hospitalaria. Se basa en pensar en cada individuo como único, lo que conlleva que el tratamiento que se le aplique debe ser igual de único. Esto es adaptarse a las vicisitudes del paciente, condiciones genéticas, ambientales, de estilo de vida e historia clínica concreta. Surge durante la década de 1990 de la mano del avance del Proyecto Genoma Humano, que proporcionó una visión más completa de los genes que componen el genoma. Gracias a esto se pudo identificar las variantes genéticas que pueden influir en la afección de las enfermedades y la respuesta de los pacientes a los tratamientos.

Los principales beneficios que ofrece son, entre otros, tratamientos más precisos. Debido a que se despoja del enfoque de "tratamiento de talla única" se aumenta la efectividad de estos ya que se tienen en cuenta las condiciones individualizadas del paciente que recibe el tratamiento. La reducción de los efectos secundarios es otra de las ventajas que nos ofrece este enfoque más preciso, gracias a tener en cuenta las particularidades genéticas de cada paciente en el tratamiento, disminuye el potencial rechazo del paciente a la solución que se le aplique.

Estos avances no vienen solos. Se busca también cambiar el enfoque de la medicina actual. Pasar de la idea de medicina reactiva, actuar sobre el paciente una vez ha contraído la enfermedad. Sin embargo, esta idea está cambiando en favor de la medicina predictiva, centrándose en anticiparse a que una persona desarrolle una enfermedad en un futuro próximo. Esto no sería posible sin los avances en genómica, biología computacional e inteligencia artificial aplicada a la salud. Pese a las potenciales aplicaciones de todos estos avances a innumerables áreas de la medicina y la asistencia sanitaria, este trabajo versará acerca de la predicción de la diabetes mellitus, así como de la detección precoz de esta enfermedad.

2.2 Problema

La diabetes mellitus engloba un conjunto de afecciones que impactan la manera en que el organismo procesa la glucosa presente en la sangre. Esta glucosa es esencial como fuente de energía para las células que componen los músculos y tejidos, además de ser el principal combustible para el cerebro. Las causas fundamentales de la diabetes varían según su tipo. Sin embargo, sin importar el tipo de diabetes que se padezca, puede resultar en un exceso de glucosa en la sangre, lo cual puede ocasionar serios problemas de salud. Las formas crónicas de diabetes comprenden tanto el tipo 1 como el tipo 2, pero existen otras formas como prediabetes o gestacional.

Se considera prediabetes, a aquellos niveles de glucosa en sangre que son más altos de los normal pero no lo suficiente como para considerarse diabetes. Sin embargo, estos niveles aumentan el riesgo de diabetes de tipo 2, lo que será un indiciar para mejorar los hábitos de alimentación y actividad física con el objetivo de prevenir la diabetes de tipo 2.

En el caso de la diabetes gestacional, se manifiesta en mujeres embarazadas que nunca han contraído diabetes. En este caso, el bebé podría presentar complicaciones de salud, puede aumentar la probabilidad de que tenga obesidad o diabetes tipo 2 a lo largo de su vida. Pese que este tipo de diabetes suele desaparecer después del parto, puede aumentar también el riesgo de que la madre tenga diabetes tipo 2 durante su vida.

Según el Centro para el Control y Prevención de Enfermedades de Estados Unidos (CDC), la diabetes tipo 1 es causada por una reacción autoinmunitaria, donde el cuerpo de ataca a sí mismo por error, lo que impide que el cuerpo genere insulina, afecta entre el 5 y el 10% de los pacientes con diabetes, y estos deben tomar insulina diariamente para sobrevivir[5]. Aunque se manifiesta con síntomas en niños y adolescentes, actualmente, no hay evidencias de prevención de este tipo de diabetes.

En el caso de la diabetes tipo 2, el problema reside en que el cuerpo no administra correctamente la insulina, lo que causa que los niveles de azúcar en sangre no se mantengan a unos niveles normales. A diferencia de la de tipo 1, puede no presentar síntomas, lo que dificulta su diagnóstico y posterior tratamiento. Por esta razón, los análisis de glucosa en sangre cobran una mayor importancia, sobre todo si se pertenece a un grupo de riesgo (mayor de 45 años, prediabetes, sobrepeso, antecedentes genéticos, etc.). Sin embargo, a diferencia del tipo anterior, este si se puede prevenir manteniendo hábitos saludables de alimentación y ejercicio físico.

Sin duda, el aumento del desarrollo y la popularización de dispositivos electrónicos para la monitorización de ciertas constantes vitales del cuerpo ha ayudado a democratizar el acceso a estas mediciones, lo que les ayuda a mantener hábitos saludables, como los dispositivos ponibles o *wearables*. Con el objetivo de mejorar las mediciones de la salud, se han desarrollado otra serie de biosensores que nos permiten leer estos datos directamente desde nuestro cuerpo. Según el National Institutes of Health (NIH)[6], uno de los primeros biosensores usadas en el ámbito de la medicina y el diagnóstico es el termómetro de mercurio con el objetivo de poder detectar cambios en la temperatura del cuerpo, lo que permitía saber si un paciente tenía o no fiebre, señal de alarma de que el paciente en cuestión puede tener una infección.

Desde 2014, está disponible en España el sistema Freestyle Libre o FLASH desarrollado por la farmacéutica Abbott. Un dispositivo con biosensores que permite a los pacientes con diabetes monitorear sus niveles de glucosa en sangre. Evitando así varios pinchazos diarios para conocer estos niveles en un glucómetro tradicional. Lo que supone un claro avance para los pacientes, puesto que, gracias a este dispositivo menos intrusivo, mejora su calidad de vida respecto a la enfermedad. Aunque las primeres versiones no eran muy sofisticadas y se necesitaba un dispositivo externo específico para la lectura de las mediciones. Actualmente, han desarrollado aplicaciones móviles en las cuales pueden visualizar de forma mucho más sencilla e intuitiva los datos de las mediciones mediante la tecnología NFC, la misma utilizada en las tarjetas bancarias. Esto supone un paso más para mejorar el tratamiento de la enfermedad, aportando naturalidad, puesto que se usa un dispositivo móvil que, actualmente, posee la mayor de la población. Esto también

[5] Centro para el Control y Prevención de Enfermedades, 2022
[6] National Institutes of Health [NIH], 2017

democratiza el acceso a estos datos y da un mayor control e independencia a los pacientes con su enfermedad[7].

Recientemente, desde 2020, a causa de la pandemia de COVID-19 (COronaVIrus Disease) una enfermedad infecciosa respiratoria que provoca disnea, tos, fiebre y en los casos más graves neumonía e insuficiencia respiratorio. Fue causada por el virus SARS-COV-2. Según Reuters[8], se calcula que ha habido alrededor de 556.201.00 contagios comunicados y alrededor de 6.776.000 muertes. Paralizó el mundo entero, incluyendo confinamientos enteros de ciudades y desbordamiento de los sistemas de salud de los países. Debido a esta situación de saturación y confinamiento unido a que una de las principales complicaciones era que se podía transmitir el virus siendo asintomático, se popularizaron dispositivos de medición de oxígeno personales, oxímetros, para monitorizar los niveles en sangre, lo que permitía anticiparse a los síntomas y tomar las medidas oportunas de confinamiento y tratamiento.

A nivel mundial, según la SED, 537 millones de adultos viven con diabetes, ha incrementado un 16% sobre las últimas estimaciones de la FID, la Federación Internacional de Diabetes. A nivel nacional, España es el segundo país de Europa con mayor prevalencia de diabetes. Alcanzando el 14,8%, uno de cada siete adultos. Entre 2019 y 2021, el número de personas con diabetes en España ha aumentado un 42%. Además, se estima que el 30,3% de las personas que padecen diabetes no han sido diagnosticadas[9]. Con el grave riesgo que conlleva para la salud, como infarto de miocardio, accidente cerebrovascular, ceguera e incluso amputación de miembros inferiores.

Por lo tanto, queda de manifiesto la importancia de la prevención y el diagnóstico de esta enfermedad para mejorar la calidad de vida de los pacientes. Todo ello impulsado por los grandes avances tecnológicos arriba expuestos.

Por otro lado, existen condiciones relacionadas con la diabetes que potencialmente pueden ser reversibles, como la prediabetes y la diabetes gestacional, como se ha comentado previamente. Sin embargo, todavía no existe cura para los tipos de diabetes.

2.3 Objetivos generales y específicos

Por todo lo expuesto, se define como objetivo general de este trabajo la aplicación de varias estrategias y enfoques basados en machine learning y deep learning para la predicción y su consecuente diagnóstico temprano de la diabetes mellitus, en el horizonte de la medicina de precisión y la mejora de la vida de los pacientes gracias a la inteligencia artificial. También se definen una serie de objetivos específicos:

- Realizar un análisis exploratorio exhaustivo de datos.

- Concienciación acerca de la diabetes mellitus, divulgar los factores de riesgo, así como los síntomas a observar en el caso de sospecha de padecer esta enfermedad.

[7] R. M. D. Fuentes, 2017
[8] Bhatia et al., 2022
[9] *España es el segundo país con mayor prevalencia de diabetes de Europa*, 2021

- Divulgar la aplicación de la inteligencia artificial a la atención sanitaria, las especificaciones técnicas, así como explicar de forma clara y concisa los conceptos que algunas veces pueden parecer abstractos o confusos.

- Exponer de forma tranquilizadora cómo estas herramientas de inteligencia artificial tienen el potencial transformador tanto para los profesionales sanitarios como para mejorar la calidad asistencial que se ofrece a los pacientes.

3. Materiales y métodos

3.1 Machine learning

Como se ha mencionado anteriormente, la estrategia escogida para lograr el diagnóstico temprano esperado es el machine learning, principalmente. Aunque más adelante se expondrán otros enfoques.

El machine learning (ML) o aprendizaje automático, trata de dotar a las máquinas de la capacidad de aprendizaje. Los modelos basados en ML aprenden a resolver distintas tareas en base a los datos que se les proporciona como ejemplos representativos de la rarea a resolver. A partir de aquí, buscan patrones en esos datos de entrada que, más tarde, puedan abstraerse a nuevos datos y por tanto hacer predicciones. En definitiva,

En definitiva, estos sistemas basados en ML aportan un nuevo enfoque de programación en el que en base a datos y respuestas el sistema nos provee de reglas que podamos aplicar en datos. Enfoque opuesto al de la programación tradicional. Según François Chollet[10], investigador de IA en Google Brain y creador de la biblioteca Keras, el ML descubre reglas para ejecutar una tarea de procesamiento de datos, puesto que se le proporcionan ejemplos de lo que espera, según el tipo de aprendizaje.

Existen varios tipos de aprendizaje mediante los cuales las maquinas pueden aprender. Se engloban en tres tipos: supervisado, no supervisado y por refuerzo.

El aprendizaje supervisado se basa en proveer al modelo de los datos etiquetados o con la salida esperada, con el objetivo de generalizar lo aprendido a datos diferentes. En el caso del no supervisado, al contrario, el modelo deberá agrupar conjuntos de datos sin etiquetar, se usan para extraer patrones nuevos en los datos. El aprendizaje por refuerzo se basa en un sistema de recompensa en el que se “premia” o “castiga” al modelo según cumpla los objetivos o no, por tanto, aprende por prueba y error, buscando el “premio”.

3.2 Datos

Antes de aplicar cualquier tipo de algoritmo sobre los datos y obtener una solución o un modelo capaz de hacer predicciones, es necesario tratar o preprocesar los datos a usar para el entrenamiento. La potencia de los algoritmos basados en ML se basa en la cantidad de datos que se puedan reunir sobre un problema para poder alimentar al modelo de la mejor forma posible. Lo que resulta en una alta precisión en las predicciones con nuevos datos. Pero este es no es el único requisito para alcanzar buenos resultados, no solo importa la cantidad sino también la calidad de estos.

[10] Chollet, 2019, p. 26

Se suele usar la frase "*garbage in, garbage out*", basura que entra basura que sale, para ilustrar que, si disponemos de una enorme cantidad de datos, pero de mala calidad, el rendimiento del modelo igual. Por ello, la importancia de los datos de entrada y cuan representativos son del problema, así como las técnicas de limpieza y preprocesado de los mismos.

En este caso se va a usar el dataset proporcionado conocido como PID (Pima Indians Dataset). El uso de este dataset, como la presentación de los presentes resultados, tiene como objetivo predecir si un paciente tiene o no diabetes basado en ciertos predictores clínicos y una variable objetivo. Según la página de Kaggle, se ha hecho una selección de un conjunto de pacientes de una base de datos más grande proveniente del National Institute of Diabetes and Digestive and Kidney Diseases.

3.2.1 Herramientas

Se han usado una serie de librerías de Python que facilitan el AED:

- Pandas: biblioteca de análisis de datos de Python muy popular en el sector. Sirve como herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar a través de sus Dataframes.

- Matplotlib: biblioteca para la generación de gráficos en dos dimensiones a partir de una variedad de fuentes de información, ya sean listas, Dataframes de pandas, arrays u otras fuentes.

- Seaborn: biblioteca de visualización de datos de alto nivel basada en matplotlib, con la ventaja de que ofrece una interfaz mucho más limpia, atractiva y personalizable.

- Pyplot: es una biblioteca de gráficos interactiva y de código abierto que admite más de 40 tipos de gráficos únicos que cubren una amplia gama de casos de uso entre ellos estadísticos y científicos[11].

3.3 Análisis exploratorio de datos (AED)

Antes de cualquier análisis de los datos, he de mencionar que en la propia página de Kaggle, se indica que todas las apariencias son de pacientes mujeres de al menos 21 años de herencia india Pima. Donde no hay ninguna variable categórica, todas las variables son numéricas.

Los predictores constan de 8 variables independientes y una variable dependiente:

- Pregnancies: número total de veces que la paciente ha estado embarazada
- Glucose: concentración plasmática de glucosa a las 2 horas en una prueba oral de tolerancia a la glucosa
- Blood Pressure: presión arterial diastólica (mm Hg)
- Skin thickness: grosor del pliegue cutáneo del tríceps (mm)
- Insulin: insulina sérica a las 2 horas (mu U/ml)
- BMI: índice de masa corporal ($\frac{peso\ en\ g}{(alturaenm)^2}$
- Diabetes pedigree function: función que asigna la probabilidad de padecer diabetes a partir de la historia familiar.

[11] (Plotly, s.f.)

- Age: edad de las pacientes en años
- Outcome: variable binaria objetivo, 0 (no diabetes) o 1 (diabetes)

Se ha analizado el dataset importándolo con Pandas, en líneas generales, se ha visto que contiene información de 768 pacientes mujeres de al menos 21 años, como se ha mencionado anteriormente. Se ha podido comprobar que todos los valores contenidos en el dataset son numéricos, la mayoría de las variables (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, Age, Outcome) son de tipo int64, números enteros de 64 bits de longitud. En cambio, otras como BMI o DiabetesPedigreeFunction son de tipo float64, números reales de punto flotante de 64 bits.

En un primer análisis de valores nulos o valores NaN (Not a Number), no se han identificado ninguno de los dos tipos dentro del dataset proporcionado, como se puede ver en las Figura 1 y 2.

También se ha analizado la variable objetivo, Outcome, para saber con precisión el número de pacientes de cada tipo. Las pacientes con un 0 en la columna Outcome, indicará que no tiene diabetes, por el contrario, si aparece un 1 en dicha columna, sí que tendrán la enfermedad. Los resultados se pueden ver en la Figura 3. Hay un total de 500 pacientes que no portan la enfermedad y 268 pacientes con diabetes.

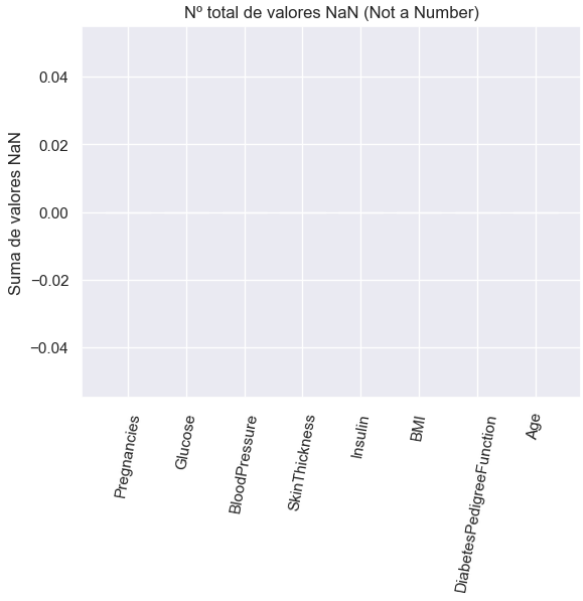


Figura 1 - Nº total de valores NaN

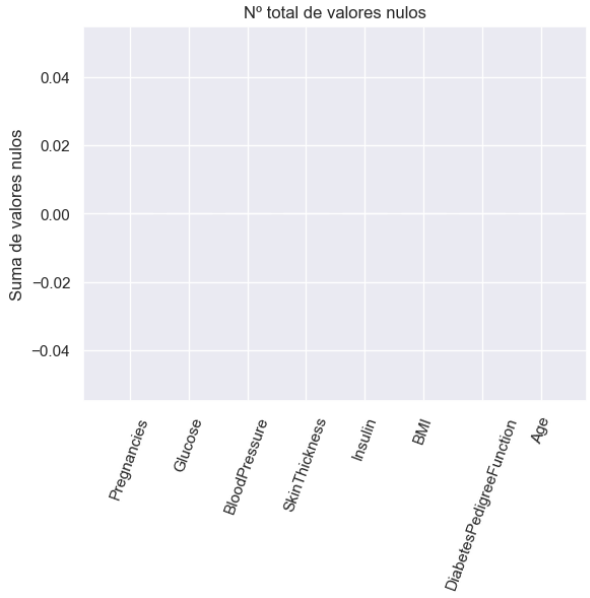


Figura 2 - Nº total de valores nulos

Sin embargo, estas conclusiones pueden ser confusas, ya que los métodos empleados solo buscan valores no numéricos (NaN) o nulos. Pero no nos permiten hacernos una idea de los posibles valores faltantes. Por ello, haciendo uso de las funciones que provee Pandas, podemos obtener un resumen de información estadística descriptiva de la tendencia central, dispersión, rango etc. La Figura 3 ilustra esta información del dataset inicial.

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.85	3.37	0.00	1.00	3.00	6.00	17.00
Glucose	768.0	120.89	31.97	0.00	99.00	117.00	140.25	199.00
BloodPressure	768.0	69.11	19.36	0.00	62.00	72.00	80.00	122.00
SkinThickness	768.0	20.54	15.95	0.00	0.00	23.00	32.00	99.00
Insulin	768.0	79.80	115.24	0.00	0.00	30.50	127.25	846.00
BMI	768.0	31.99	7.88	0.00	27.30	32.00	36.60	67.10
DiabetesPedigreeFunction	768.0	0.47	0.33	0.08	0.24	0.37	0.63	2.42
Age	768.0	33.24	11.76	21.00	24.00	29.00	41.00	81.00
Outcome	768.0	0.35	0.48	0.00	0.00	0.00	1.00	1.00

Figura 3 - Estadística descriptiva de las columnas del dataset

Como se puede observar, se ha marcado en rojo la columna que indica el valor mínimo de cada columna. Estas columnas son: Glucose, SkinThickness, BloodPressure, BMI, Insulin. Esto es inusualmente raro, puesto que un nivel de glucosa a 0 es incompatible con la vida, un valor de 0 en la presión sanguínea revela un nulo flujo sanguíneo, así como un valor 0 en BMI (Body Mass Index) un problema extremo de nutrición. De la misma forma con los valores de insulina o grosor de piel, indican valores no fisiológicamente posibles. A la vista de estos datos, se puede inferir que el dataset contiene valores faltantes, posiblemente debido a errores durante la medición de estos.

En las Figuras 4 y 5, se puede visualizar el número total de apariencias con valores 0 en las columnas arriba mencionadas.

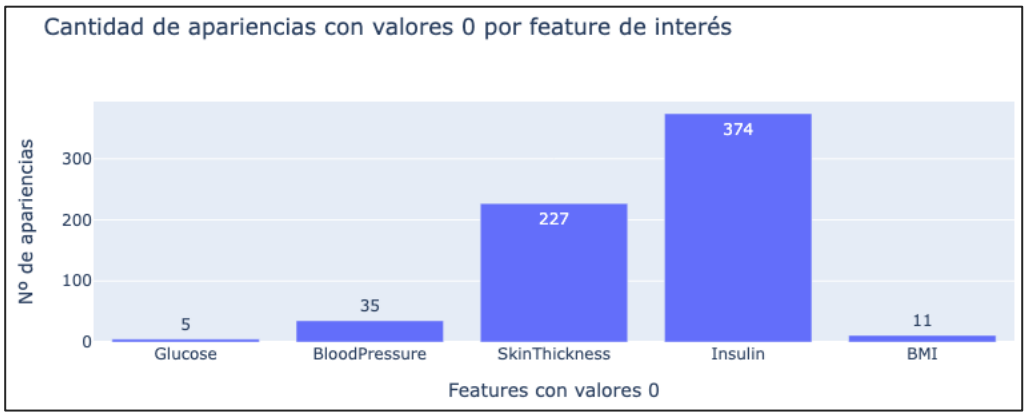


Figura 4 - Cantidad de valores en 0 para las columnas indicadas

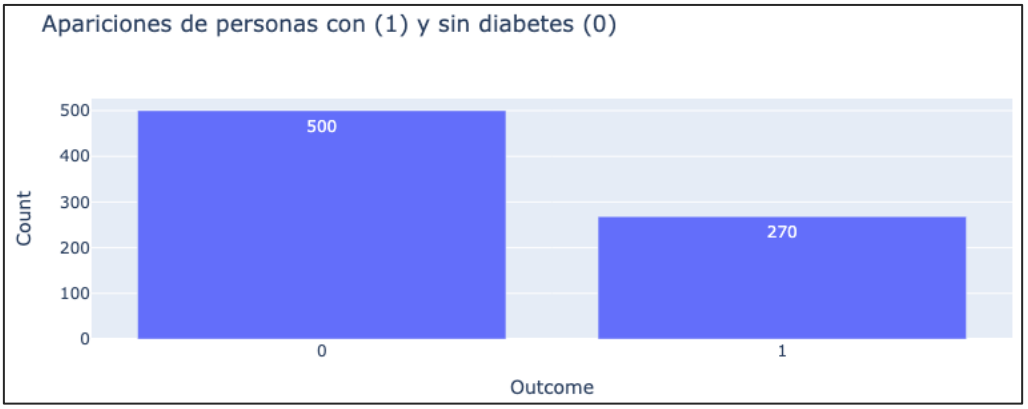


Figura 5 - Valores de columna Outcome

Hay bastantes valores en 0 en varias columnas, lo que justifica aplicar un tratamiento específico a los datos

Con el objetivo de sacar más información de los datos, se ha explorado la columna *Age*. Se ha visto que la mayoría de los pacientes están comprendidas en el rango de 20-30 años, no habiendo ninguna pasando lo 90 años. El número exacto de personas por rango de edad se muestra en la Figura 6 y el gráfico en la Figura 7**:**

Rango de edad	**(20, 30]**	**(30, 40]**	**(40, 50]**	**(50, 60]**	**(60, 70]**	**(70, 80]**	**(80, 90]**	**(90, 100]**
Nº de pacientes	417	157	113	54	25	1	1	0

Figura 6 - Nº total de pacientes por rango de edad

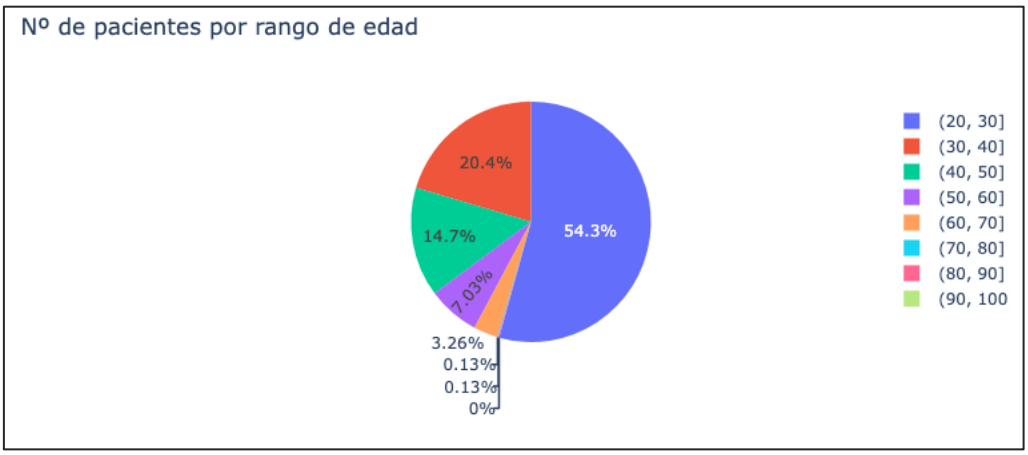


Figura 7 - Número de pacientes por grupo de edad

3.3.2 Análisis de distribuciones de las variables

Este apartado se vertebra sobre las distribuciones de probabilidad existentes en las variables predictoras que condicionan la variable objetivo:

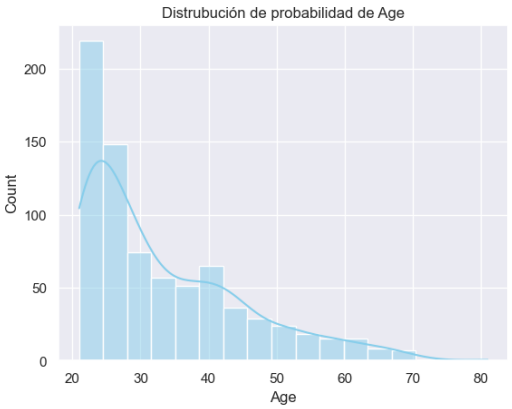


Figura 8 - Distribución de probabilidad de Age

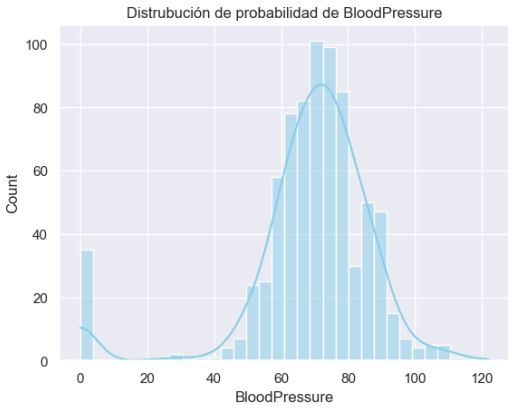


Figura 9 - Distribución de probabilidad de Blood Pressure

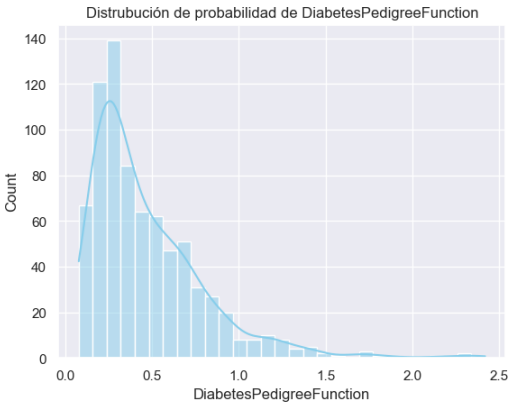


Figura 10 - Distribución de probabilidad de Diabetes Pedigree Function

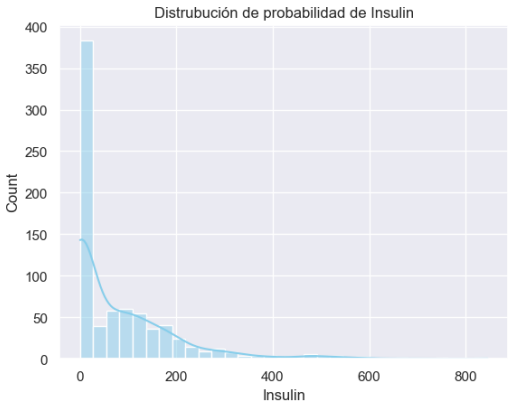


Figura 11 - Distribución de probabilidad de Insulin

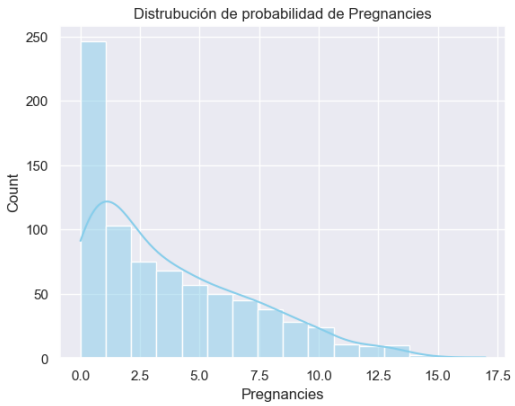


Figura 12 - Distribución de probabilidad de Pregnancies

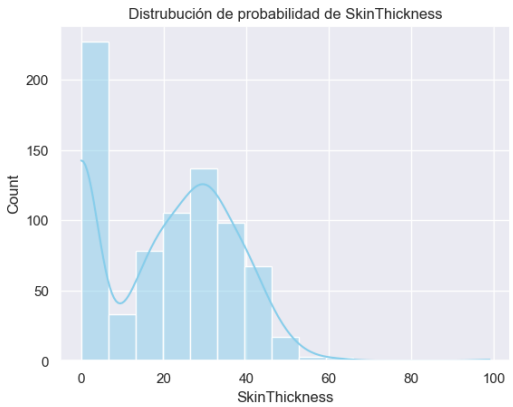


Figura 13 - Distribución de probabilidad de Skin Thickness

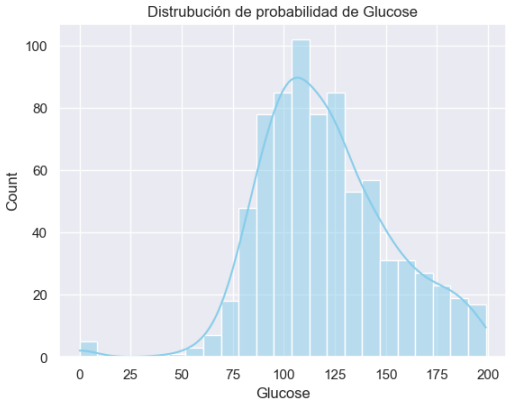


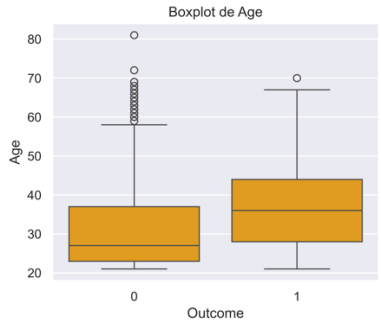
Figura 14 - Distribución de probabilidad de Glucose

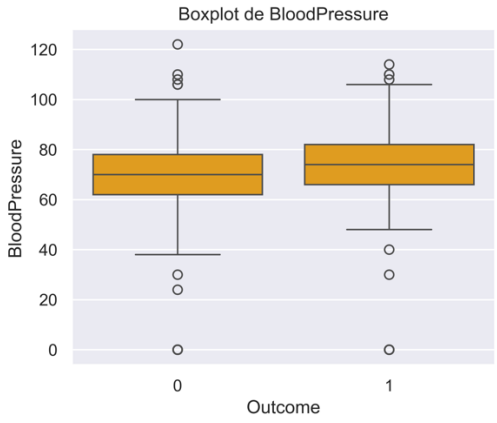
Para el análisis de las distribuciones se ha usado el histograma, ya que es una representación visual muy útil de la distribución de un conjunto de datos. Se organiza en barras, donde cada barra representa la frecuencia o la cantidad de datos en un intervalo específico. La altura de cada barra indica la frecuencia de datos en ese intervalo, proporcionando una visión rápida de la forma y la tendencia de la distribución de los datos. Permite ver de forma rápida la simetría de los datos, la forma de la distribución y lo sesgados o no que están los datos en ellos representados, así como aportar información valiosa acerca de la tendencia central de los mismos.

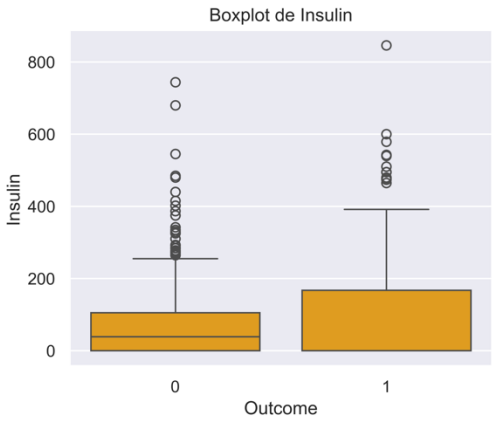
Por ejemplo, en la Figura 12**,** basándonos en el gráfico, se podría decir que la variable Pregnancies sigue una distribución geométrica. Observando las gráficas de las Figuras 9 y 14, se puede ver que ambas son simétricas respecto de un valor centrado y la forma acampanada de la función de densidad, puede sugerir que ambas variables (*Blood pressure* y *Glucose*) sigan una distribución normal centrada en un valor de la media entre 100 y 125 y entre 70 y 80 respectivamente.

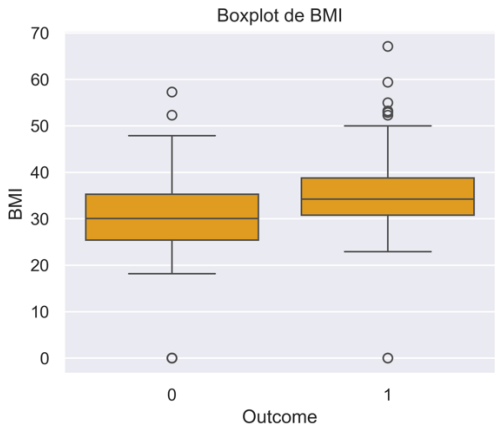
Otra de las técnicas que se pueden emplear, nuevamente, con el objetivo de obtener más información de los datos y su distribución es el uso de diagramas de caja y bigotes o *box plot.* Son una representación gráfica que muestra la distribución y resumen estadístico de un conjunto de datos, son especialmente útiles a la hora de identificar valores atípicos. Constan de varias partes, el cuadro central representa el rango Inter cuartil (IQR), la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1), que contiene el 50% central

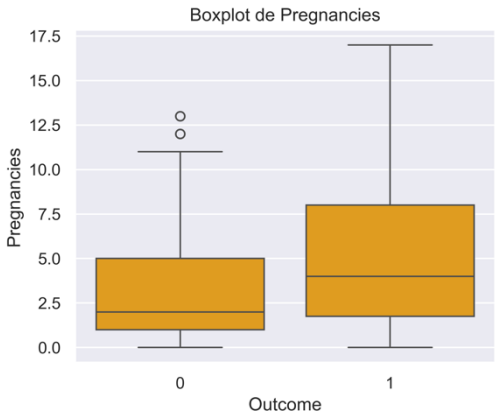
de los datos. La mediana se muestra como una línea dentro del cuadro. Los "bigotes" se extienden desde el cuadro hasta los valores extremos, proporcionando una indicación visual de la dispersión de los datos. Los valores atípicos pueden mostrarse como puntos individuales más allá de los bigotes, lo que facilita su visualización e identificación.

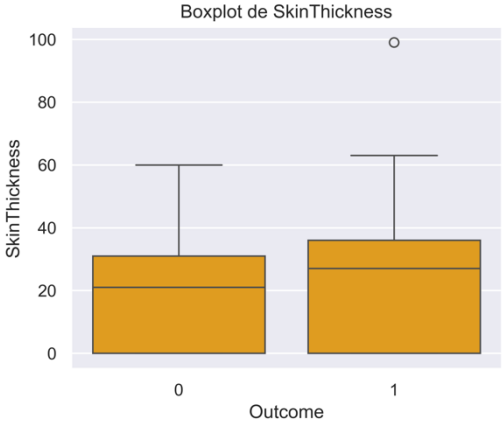












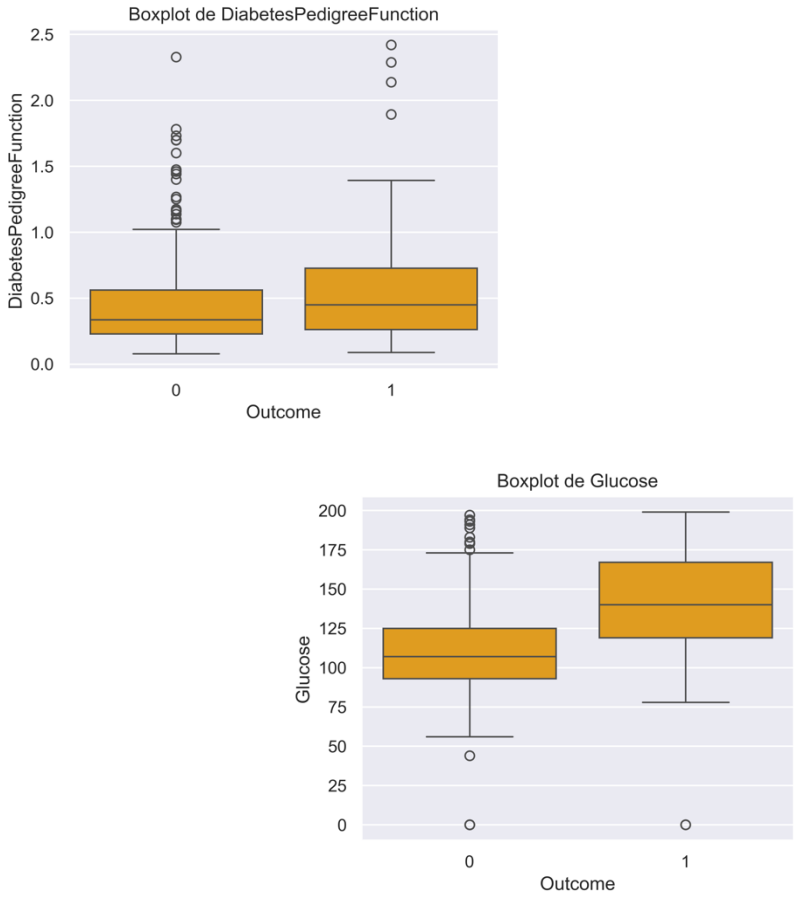


Figura 15 - Boxplot de las variables predictoras

En base al artículo proporcionado como fuente, se muestran los boxplot de cada variable predictora en base a la variable a predecir, *Outcome*.

3.3.3 Imputación de valores atípicos

Con el objetivo de eliminar la máxima confusión y ruido que pueda afectar al rendimiento de los modelos, se han transformado los filas del dataset que contengan valores 0 con la media de los valores de esa columna. Se muestran algunos resultados en las Figura 16 y 17:

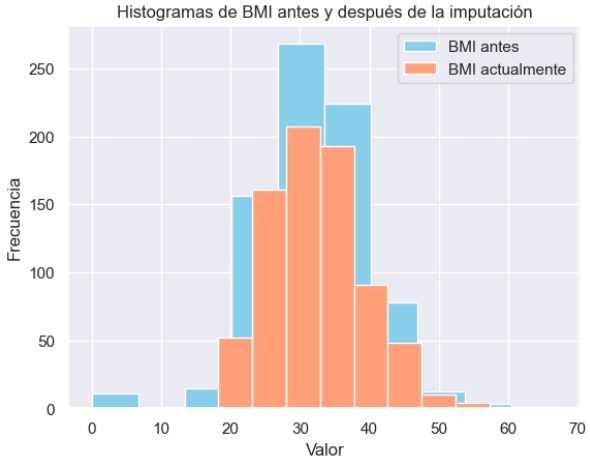


Figura 16 - Histogramas de BMI previa y posteriormente a la imputación

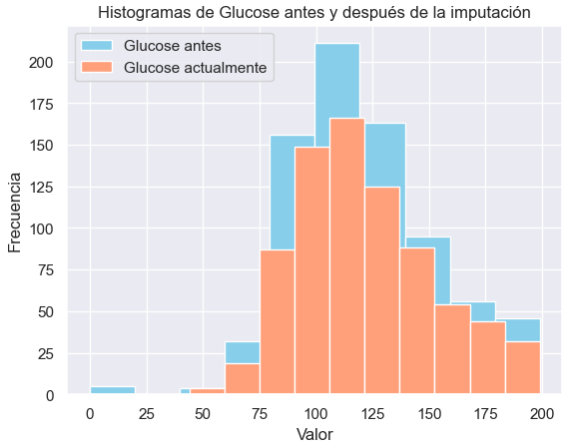


Figura 17 - Histogramas de Glucose previa y posteriormente a la imputación

3.3.4 Reescalado y normalización de datos

Una vez se han gestionado los valores atípicos presentes en el conjunto de datos, es conveniente también realizar un reescalado y normalización de estos para aumentar la precisión y rendimiento de los modelos a utilizar.

Primero es necesario aplicar un reescalado a los datos. Debido a que normalmente, los datos se encuentran en un rango diverso de valores es preciso transformarlos a un rango común que facilite el trabajo a los modelos.

Una vez reescalados los datos se procede a normalizarlos. Este proceso consiste en ajustar las variables de entrada para que tengan una distribución de valores común. El objetivo

es centrar y escalar las características para que tengan propiedades estadísticas similares, lo que beneficiará al rendimiento de los modelos que reciban estos datos como entrada.

La técnica más común es la estándar o *z-score normalization*, se lleva a cabo restando la media del conjunto a cada valor y dividiendo el resultado por la desviación estándar:

$$z = \frac{x - \mu}{\sigma}$$

La función StandardScaler[12] implementa esta fórmula en sklearn, y por tanto ha sido la seleccionada para reescalar los datos. Los resultados se pueden observar en la Figura 19 un ejemplo de reescalado de la media y la desviación estándar de cada feature.

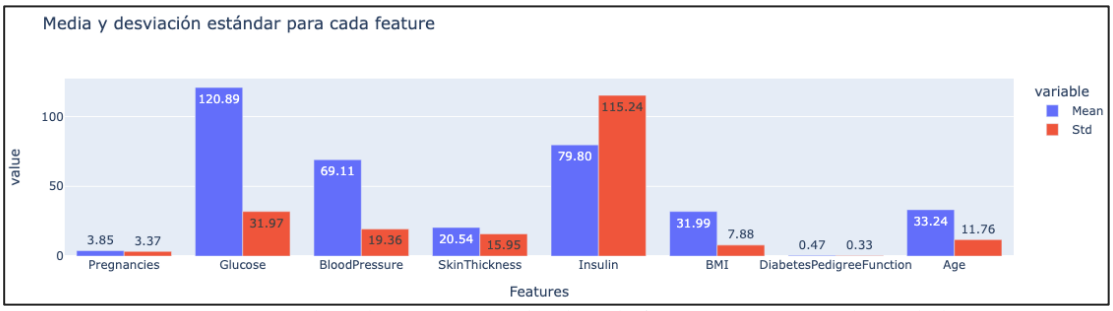


Figura 18 - Media y desviación estándar de cada feature previamente al rescalado

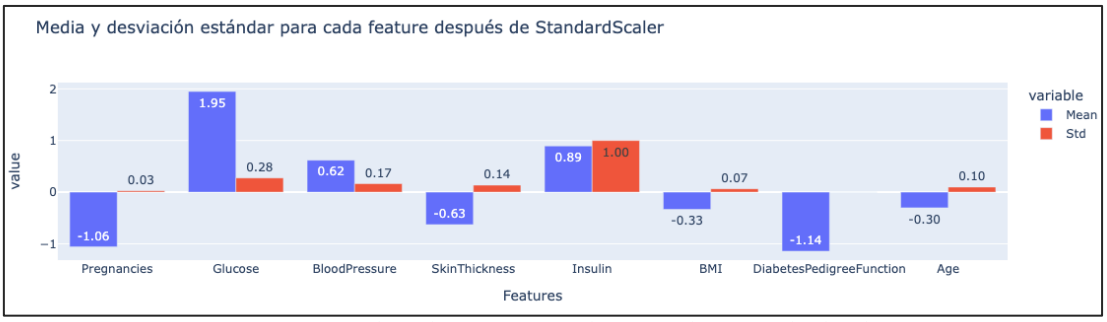


Figura 19 - Media y desviación estándar tras aplicar StandardScaler

3.4 División del dataset en conjuntos de entrenamiento y test

Antes de continuar y mostrar los datos a los algoritmos para que puedan efectuar predicciones, es necesario dividir el conjunto inicial en tres subconjuntos: entrenamiento, validación y test.

Este paso es de vital importancia para que los modelos aprendan a generalizar y no solo memorizar los valores de entrada. Cuando ocurre este último hecho durante el entrenamiento, se dice que el modelo sufre de overfitting o sobreajuste. Consiste en que el modelo se ajusta demasiado a los datos de entrenamiento, capturando patrones específicos de ese conjunto, pero teniendo dificultades para generalizar bien a nuevos datos no vistos.

[12] scikit-learn developers, s. f.

Una solución a este problema es no proporcionar todos los datos del conjunto durante el entrenamiento y reservar algunos para comprobar que el modelo generaliza bien. Como el conjunto de validación, una porción dedicada a optimizar los hiperparámetros (configuración de cada modelo que no se aprende de forma automática, debe hacerse previo al modelo), con el objetivo de mejorar el rendimiento de este independientemente del resto de conjunto de datos.

Otro de los subconjuntos mencionados es el de *test* o prueba. Es de utilidad cuando se quiere probar de forma imparcial la precisión y el rendimiento del modelo de forma no sesgada, puesto que para estas pruebas se utiliza un conjunto nunca visto para el modelo. Por ello, el modelo debe demostrar su capacidad de generalización.

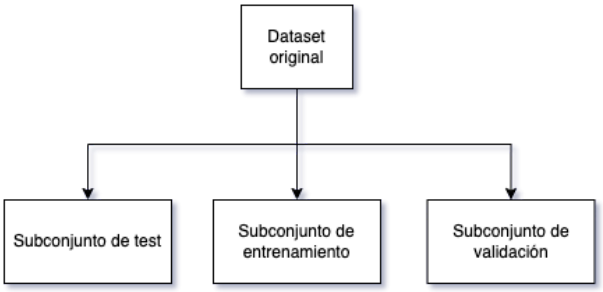


Figura 20 - Estructura del dataset original tras la división

4. Resultados y discusión

El enfoque utilizado para predecir la variable dependiente en cuestión es el de modelos basados en aprendizaje supervisado. Modelos que requieren de un gran conjunto de datos etiquetado con ejemplos de entrada y las salidas esperadas. El objetivo es que el modelo aprenda a mapear las entradas y las salidas.

Esta sección recoge los resultados de aplicar sobre el conjunto de datos original, tres algoritmos de inteligencia artificial. Dos algoritmos basados en machine learning (ML) o aprendizaje automático (AA): random forest o bosque aleatorio y máquinas de vectores de soporte o support vector machines (SVM). Y un algoritmo basado en aprendizaje profundo o deep learning (DL): perceptrón multicapa o multi-layer perceptron (MLP)

4.1 Modelos basado en machine learning

A modo de recordatorio, es un campo de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a los ordenadores aprender patrones y realizar tareas que no han sido programadas explícitamente, mejorando su rendimiento a medida que se exponen a más datos.

4.1.2 Random forest o bosque aleatorio

Se trata de un algoritmo de aprendizaje supervisado que se basa en el uso de varios árboles de decisión que realizan sus predicciones por separado para, después elegir la mejor opción, formando así un *bosque* de árboles de decisión.

Es un ejemplo de lo que se conoce como ensemble learning o aprendizaje por ensamblado, donde se combinan varios modelos individuales. En este caso, cada árbol se entrena en un subconjunto aleatorio de datos y utiliza un subconjunto aleatorio de características, de

ahí el nombre de bosque aleatorio. Cuando se hace una predicción, cada árbol en el bosque emite su propia predicción. En el caso de un problema de clasificación, se realiza una votación para determinar la clase final.

La principal ventaja de los bosques aleatorios es que están menos predispuestos al overfitting o sobreajuste, mencionado en la sección 2.4, se debe a que cada árbol del bosque solo ve un subconjunto del subconjunto completo de entrada, esto reduce las posibilidades de que un árbol concreto memorice los datos. Por tanto, esta medida de prevención ayuda a mejor la capacidad de generalización del modelo y su rendimiento.

4.1.3 Máquinas de vectores de soporte o support vector machines (SVM)

Se trata de un enfoque de ML basado en aprendizaje supervisado, donde se busca encontrar el hiperplano óptimo o hiperplano marginal máximo, que mejor separe las clases. Este hiperplano es el que mejor define la separación geométrica entre dichas clases, apoyándose en los vectores de soporte. Estos son aquellos puntos de los datos que definen la posición y el margen al hiperplano óptimo, indican las coordenadas que representan a las clases.

En resumen, este algoritmo trata de buscar el mencionado hiperplano, que deberá ser aquel que maximice el margen entre las clases, lo que indicará menor similitud entre las mismas y minimice el error de clasificación.

4.2 Modelos basados en deep learning

Esta sección recoge los resultados obtenidos a partir de usar un algoritmo de aprendizaje supervisado basado en DL. Se conoce como una rama del ML basada en el uso de redes neuronales artificiales profundas (que imitan las redes neuronales humanas) para realizar tareas concretas de clasificación y/o predicción. Se dicen profundas por la cantidad de capas de las que están formadas estas redes.

La diferencia principal y mayor ventaja de este tipo de algoritmos frente a los basados en ML, es la extracción de características. Pues en estos últimos, la selección de las características que el modelo debe aprender se realiza previamente al entrenamiento del modelo por parte de humanos.

En cambio, en los primeros, los basados en DL, en lugar de depender de que humanos extraigan las características y las entreguen al modelo, es el propio modelo el que descubre durante el entrenamiento los patrones presentes en los datos a medida que se van procesando por las diferentes capas.

4.2.1 Perceptrón multicapa o multi-layer perceptrón (MLP)

Es de las formas más básicas de redes neuronales artificiales, solo superadas por el perceptrón simple. Se trata de una red de nodos llamados neuronas con una capa de n neuronas de entrada, m capas ocultas o de procesamiento con p neuronas en cada capa y k capas de salida con *z* neuronas en la capa de salida. Donde en cada caso se utiliza una función de activación y una función de perdida.

Sin embargo, no es común encontrarse con *k+1 (con k>0)* capas de salida. Es más habitual encontrar configuraciones con una única capa de salida con tantas *z* neuronas en dicha capa, es decir con tantas neuronas como clases a clasificar.

Este caso que nos ocupa, dirimir si un paciente tiene diabetes o no, es un problema de clasificación binaria, por lo que no será necesario incluir más de una neurona en la capa de salida. Puesto que se interpretará como la probabilidad de que el dato de entrada pertenezca a la clase positiva (1). Con este objetivo, se utiliza una función de activación, en esta última capa, que se encarga de transformar la salida de la neurona de la capa de salida en un valor 0 o 1, revelando la pertenencia a una de las dos clases.

Una función de activación es una función matemática que transforma el resultado del procesamiento lineal de una neurona en una salida determinada en base a unas reglas. El principal beneficio que aportan es la no linealidad. Lo que permite al modelo aprender patrones y relaciones en los datos más complejos y realizar tareas más avanzadas. Decide si la neurona debe enviar una señal o no, si debe activarse o no.

Se utilizan varias dentro de una red. Las más usadas y que mejores resultados han demostrado para este tipo de tareas son: la función ReLU y la función sigmoide.

La función ReLU se usa en las capas ocultas, y sirve para eliminar los valores negativos presentes en los datos y mantener inalterados los valores positivos. Su fórmula es la siguiente:

$$\mathrm{f}(x) = \max(0, x)$$

La función sigmoide se usa principalmente en la capa de salida de la red, puesto que toma un valor del real del conjunto y lo transforma en otro valor real dentro del rango [0, 1]. Por ello, esta función es útil para problema de clasificación binaria. En el caso de que la función sigmoide produzca un resultado cercano o muy cercano a 1, se puede interpretar como pertenencia a esa clase, ídem para resultados cercanos a 0. Su fórmula es la siguiente:

$$\sigma(\mathrm{x}) = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}}$$

El funcionamiento general de la red consta de dos fases clave: propagación hacia delante y propagación hacia atrás (*forward* y *backpropagation*).

La propagación hacia delante se basa como el nombre indica en la transferencia de los valores de entrada a través de las capas de la neurona con el objetivo de hacer predicciones. Esta transferencia por las capas consiste en que los valores de entrada se computen con los valores de cada neurona de forma lineal, para, más tarde, ser procesados por la correspondiente función de activación y ser transferidas a la siguiente capa hasta llegar a la de salida. Representando la predicción de la red para el valor de entrada proporcionado.

Una vez se ha completado este proceso, entra en acción el proceso inverso, conocido como *backpropagation* o propagación hacia atrás. Es la fase en la que la red ajusta sus pesos y sesgos en base a los resultados de la función de pérdida o de error. Se calcula cuanta ha diferido la predicción del valor real y se transfiere a las capas anteriores para que la red aprenda de sus errores y mejore su rendimiento en futuras predicciones.

Este proceso es iterativo que se obtiene el rendimiento necesario para cada tarea.

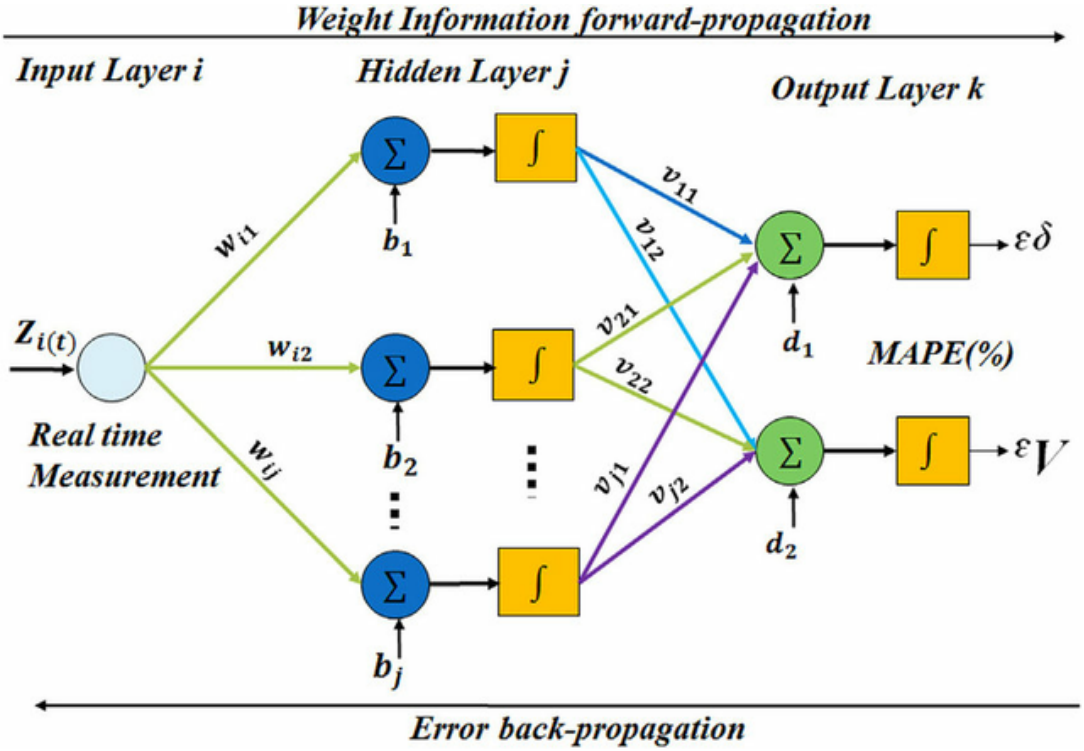


Figura 21 - Representación de una red neuronal[13]

4.3 Resultados

4.3.1 Random forest

Se ha entrenado el modelo 50 veces a través de una función propia (`train_model_iterative`). A la que se le pasan como argumentos: las características de entrada (features), la condición a predecir (target), número de veces que se quiere entrenar (iterations) y, por último, el modelo que se quiere usar para el entrenamiento. Esta función va almacenando en listas las métricas de interés (sensibilidad, especificidad y AUC) para cada iteración.

De esta forma, al finalizar el entrenamiento, se obtiene una lista con todas las métricas obtenidas. Estos valores son los que devuelve la función para, más adelante, poder tomar la media y desviación estándar de dichos valores, con el objetivo de dilucidar su rendimiento frente a otros enfoques. En la Figura 22 se puede observar el uso de dicha función con `RandomForestClassifier`:

```
(rf_sensibilities, rf_specificities, rf_AUC_scores, rf_classifier, rf_cm) = \
train_model_iterative(features=X,
                      target=y,
                      iterations=50,
                      classifier=RandomForestClassifier(n_estimators=50,random_state=42))
```

Figura 22 - Ejemplo de uso de la función train_model_iterative

[13] Mosbah, H., & El-Hawary, M. (2017)

Esta función imprimirá las métricas de interés en cada iteración, lo que nos permitirá seguir el progreso del entrenamiento:

```
                    classifier=RandomForestClassifier(n_estimators=50,random_state=42))

Entrenando el modelo 50 veces...

Iteración 1: Sensibilidad = 0.9200, Especificidad = 0.6111, AUC: 0.8736
Iteración 2: Sensibilidad = 0.8400, Especificidad = 0.4630, AUC: 0.7899
Iteración 3: Sensibilidad = 0.8800, Especificidad = 0.5000, AUC: 0.7837
Iteración 4: Sensibilidad = 0.8900, Especificidad = 0.5370, AUC: 0.7932
Iteración 5: Sensibilidad = 0.7200, Especificidad = 0.6667, AUC: 0.7643
Iteración 6: Sensibilidad = 0.8800, Especificidad = 0.5926, AUC: 0.8180
Iteración 7: Sensibilidad = 0.9400, Especificidad = 0.5370, AUC: 0.8560
Iteración 8: Sensibilidad = 0.8700, Especificidad = 0.5370, AUC: 0.8164
Iteración 9: Sensibilidad = 0.7700, Especificidad = 0.5741, AUC: 0.7594
Iteración 10: Sensibilidad = 0.8600, Especificidad = 0.7222, AUC: 0.8689
```

Figura 23 - Ejemplo de entrenamiento

Hasta que llegue a su fin:

```
Iteración 49: Sensibilidad = 0.8400, Especificidad = 0.5926, AUC: 0.8046
Iteración 50: Sensibilidad = 0.8500, Especificidad = 0.5926, AUC: 0.8378

######## Entrenamiento finalizado!!! ########
```

Figura 24 - Entrenamiento del modelo indicando que ha finalizado

Una vez finalizado, devolverá en las variables rf_*, de la figura X, las listas mencionadas con cada métrica. será útil para imprimir un resumen de las medias de estas métricas del modelo, a través de otra función propia (`print_metrics_summary`):

```
print_metrics_summary(rf_sensibilities, rf_AUC_scores, rf_specificities, "Random forest")
```

Figura 25 - Ejemplo de uso de la función print_metrcis_summary

Esta función recibe como parámetros las métricas de interés y el nombre del modelo a mostrar en el resumen, e imprime por pantalla la media y la desviación estándar de las métricas proporcionadas como parámetros:

```
-------------    Random forest      --------------
###################### RESUMEN ######################
# Sensibilidad promedio: 0.8604                    #
# Desviación estándar de Sensibilidad: 0.0429      #
# ---------------------------------------------    #
# AUC promedio: 0.8225                             #
# Desviación estándar AUC: 0.0285                  #
# ---------------------------------------------    #
# Especificidad promedio: 0.5744                   #
# Desviación estándar especificidad: 0.0655        #
```

Figura 26 - Ejemplo de ejecución de la función print_metrics_summary con Random Forest

Estas funciones son reutilizables para el resto de los modelos, lo que nos ahorrará duplicidades en el código y nos proporcionará mayor limpieza y legibilidad de este.

4.3.2 Máquinas de vectores de soporte

Para este caso, dado que sklearn proporciona varios parámetros para este tipo de modelos, se han hecho una serie de pruebas con cada parámetro con el objetivo de discernir cual de todos proporciona un mayor rendimiento. Se ha usado el módulo de la librería sklearn llamado `svm`. Dado que se trata de un problema de clasificación, se ha hecho uso del módulo svc dentro de svm. SVC son las siglas de Support Vector Classification, según la documentación oficial[14], esta implementación está basada en libsvm (library for support vector machines). Es la implementación del algoritmo de aprendizaje supervisado para clasificación.

Análogamente a otros modelos, se ha usado la función propia (`train_model_iterative`) para llevar a cabo el entrenamiento del modelo. Para reutilizar código mejorar la legibilidad de este, como se ha mencionado anteriormente. En este caso particular se ha usado varias veces para comparar el rendimiento de los modelos basados en SVM, modificando los hiperparámetros en cada ocasión.

```
(svm_rbf_sensibilities, svm_rbf_specificities, svm_rbf_AUC_scores, svm_rbf_classifier, svm_rbf_cm)
train_model_iterative(X, y, 50, svm.SVC(kernel="rbf", C=1000, probability=True))
```

```
(svm_linear_sens, svm_linear_spec, svm_linear_AUC_scores, svm_linear_clf, svm_linear_cm) = \
train_model_iterative(X, y, 50, svm.SVC(kernel="linear", C=1000, probability=True))
```

Figura 27 - Ejemplo de uso de la función train_model_iterative para SVM con diferentes parámetros

Se han probado distintos valores de *kernel y de C*. Estos parámetros controlan tanto el enfoque del SVM con el primero como el sobreajuste con el segundo. Gracias a los kernels, podemos aplicar transformaciones a los datos y operar con ellos en espacios de mayor dimensión, esto nos permite poder operar con datos no linealmente separables en ciertas dimensiones. En resumen, con estas transformaciones, se quiere conseguir la separación lineal de datos en tres dimensiones que no lo son en dos dimensiones[15]. Sklearn ofrece varios enfoques, de los cuales he probado con los más usados[16]:

- **Linear**: Realiza el producto escalar de los vectores de entrada, esto permite calcular la similitud de dichos valores.

- **Radial Basis Function (RBF)** o kernel Gaussiano: Mide la similitud entre dos puntos de datos en infinitas dimensiones y, a continuación, aborda la clasificación por voto mayoritario. Dado que la función matemática incluye, en otras, la distancia euclidiana, Cuanto mayor sea entre dos puntos, la función kernel más se acercará a cero. Como consecuencia, dos puntos alejados tienen más probabilidades de no ser similares.

El parámetro C, es el parámetro de regularización, penaliza la clasificación errónea del modelo. Por tanto, su valor influirá en la capacidad de generalización de este. Sin

[14] scikit-learn developers, s. f.
[15] KeepCoding, 2023
[16] *Plot classification boundaries with different SVM kernels*, s. f.

embargo, la elección de su valor afectará al sobreajuste (memorización de los datos de entrada y baja o nula generalización del modelo en nuevos datos) del modelo.

Un valor bajo de C significa que el modelo está dispuesto a aceptar un margen de error mayor, lo que afectará a su rendimiento y a su capacidad de generalización. Por otro lado, un valor muy grande de C implicará que el modelo no permitirá casi ningún error y un margen de error muy pequeño, con el riesgo de sobreajustar el modelo, como se ha mencionado anteriormente. Por tanto, se deberán hacer varias ejecuciones con distintos valores para encontrar el valor del hiperparámetro que proporcione un mejor rendimiento.

Como se ha mencionado anteriormente, se han probado varios casos con el objetivo de obtener el mayor rendimiento, como se aprecia en la Figura 28:

```
(svm_rbf_sensibilities_1000, svm_rbf_specificities_1000,
 svm_rbf_AUC_scores_1000, svm_rbf_classifier_1000, svm_rbf_cm_1000) = \
train_model_iterative(X, y, 50, svm.SVC(kernel="rbf", C=1000, probability=True))

(svm_linear_sensibilities_1000, svm_linear_specificities_1000,
 svm_linear_AUC_scores_1000, svm_linear_classifier_1000, svm_linear_cm_1000) = \
train_model_iterative(X, y, 50, svm.SVC(kernel="linear", C=1000, probability=True))

(svm_linear_sens_100, svm_linear_spec_100, svm_linear_AUC_scores_100,
 svm_linear_clf_100, svm_linear_cm_100) = \
train_model_iterative(X, y, 50, svm.SVC(kernel="linea", C=100, probability=True))

(svm_rbf_sensibilities_100, svm_rbf_specificities_100,
 svm_rbf_AUC_scores_100, svm_rbf_classifier_100, svm_rbf_cm_100) = \
train_model_iterative(X, y, 50, svm.SVC(kernel="rbf", C=100, probability=True))
```

Figura 28 - Combinaciones posibles de kernel y C

Se han obtenido los siguientes resultados:

```
-------------    SVM kernel=RBF C=1000       ---------
###################### RESUMEN ######################
# Sensibilidad promedio: 0.7522                      #
# Desviación estándar de Sensibilidad: 0.0476      #
# ---------------------------------------------    #
# AUC promedio: 0.5667                               #
# Desviación estándar AUC: 0.0599                    #
# ---------------------------------------------    #
# Especificidad promedio: 0.7095                     #
# Desviación estándar especificidad: 0.0394        #


-------------    SVM kernel=linear C=1000       ------
###################### RESUMEN ######################
# Sensibilidad promedio: 0.8796                      #
# Desviación estándar de Sensibilidad: 0.0351      #
# ---------------------------------------------    #
# AUC promedio: 0.5663                               #
# Desviación estándar AUC: 0.0491                    #
# ---------------------------------------------    #
# Especificidad promedio: 0.8321                     #
# Desviación estándar especificidad: 0.0271        #
```

Figura 29 - Resultado de ejecución para los parámetros mostrados

```
-------------    SVM kernel=RBF C=100       ---------
###################### RESUMEN ######################
# Sensibilidad promedio: 0.7828                     #
# Desviación estándar de Sensibilidad: 0.0437     #
# ---------------------------------------------    #
# AUC promedio: 0.5585                              #
# Desviación estándar AUC: 0.0646                   #
# ---------------------------------------------    #
# Especificidad promedio: 0.7266                    #
# Desviación estándar especificidad: 0.0405       #


-------------    SVM kernel=linear C=100       ------
###################### RESUMEN ######################
# Sensibilidad promedio: 0.8796                     #
# Desviación estándar de Sensibilidad: 0.0351     #
# ---------------------------------------------    #
# AUC promedio: 0.5663                              #
# Desviación estándar AUC: 0.0497                   #
# ---------------------------------------------    #
# Especificidad promedio: 0.8320                    #
# Desviación estándar especificidad: 0.0271       #
```

Figura 30 - Resultado de ejecución para los parámetros mostrados de kernel y C

El que mejor resultado ha obtenido en términos de sensibilidad es el kernel linear. Por lo tanto, será el modelo que se tome como referencia para comparar con el resto de los enfoques.

4.3.3 Redes neuronales artificiales

En este caso se ha usado un enfoque basado en DL, con arquitectura de tipo perceptrón multicapa. Se ha partido de un modelo *Sequential* que provee la librería Keras, comentada anteriormente. Según la documentación[17], El modelo *Sequential* de Keras es una forma de crear modelos de aprendizaje profundo donde se crea una instancia de la clase *Sequential* y se crean y añaden capas de modelo a ella. Se representa como una pila lineal de capas, donde cada capa tiene exactamente un tensor de entrada y un tensor de salida. Que son contenedores de datos, a menudo numéricos, por tanto, son contenedores numéricos. Son una generalización de matrices para un número arbitrario de dimensiones (ejes). Un tensor se define por tres atributos clave:

- **Número de ejes o número de dimensiones**. Donde un tensor 1D tiene 1 dimensión o un eje, es lo que se conoce como vector [1, 2, 3], contiene un solo eje con 3 elementos en el eje.

- **Forma:** Dupla de enteros describiendo cuántas dimensiones tiene el tensor en cada eje. El vector anterior tendría una forma del tipo (3,), una matriz del tipo [[1, 2, 3], [4, 5, 6]] tendría una forma del tipo (2, 3)

- **Tipo de datos:** Como su nombre indica, es el tipo de datos contenido en el tensor, puede ser float32, uint8, float64 etc...

A partir de aquí, crear una arquitectura de red neuronal tan sencillo como ir apilando capas en el modelo, en este caso capas de tipo *Dense*. Según Keras[18]**,** "*Sólo una capa NN normal densamente conectada*" donde una capa densa es aquella en la que todas las neuronas de entrada de la capa están conectadas con todas las neuronas de salida, y como se ha comentado anteriormente, se pueden definir las funciones de activación de cada capa, comúnmente ReLU y en este caso, por su naturaleza de clasificación binaria, función de activación sigmoide en la capa de salida. Se puede observar dicha arquitectura en la siguiente Figura:

```
# Crear modelo
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=X.columns.size))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Figura 31 - Arquitectura de red neuronal propuesta

Una vez definido el modelo, se debe compilar. La compilación del modelo es el proceso en el que se define cómo va a aprender y las métricas que queremos obtener tras el entrenamiento para valorar su rendimiento. En otras palabras, en este paso se especifica el optimizador que se utilizará para ajustar los pesos del modelo, la función de pérdida

[17] Keras, s. f.
[18] Keras, s.f.b

que se utilizará para evaluar como de bien está funcionando el modelo, y cualquier métrica de interés. Como se puede observar en la Figura 32.

```
# Compilar modelo
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figura 32 - Compilación del modelo Sequential con la función de pérdida, optimizador y métricas a monitorizar

Se ha usado como función de perdida *BinaryCrossentropy*[19]. Calcula la "entropía cruzada" entre las etiquetas verdaderas y las etiquetas predichas. Básicamente, mide la "distancia" entre las etiquetas reales y las predicciones que hace el modelo.

Después de este paso, se procede a entrenar el modelo. Se usa el método fit de Keras, que necesita los conjuntos de entrenamiento, test y validación, se define el número de veces que se dará una vuelta completa al conjunto de entrenamiento, llamados epochs. Y en este caso, se ha definido una función de callback para que Tensorboard monitorice el entrenamiento del modelo con las métricas basadas en los pesos, conjunto de test y validación.

Una callback es un objeto que puede realizar acciones en distintas fases del entrenamiento (por ejemplo, al principio o al final de un epoch, antes o después de un batch, etc.). Permite tener una visión de los estados internos y estadísticas del modelo durante el entrenamiento (Keras, s.f.c). Lo que puede ser útil para monitorizar el entrenamiento con herramientas como Tensorboard.

Tensorboard es un conjunto de herramientas de visualización que permiten visualizar métricas importantes como la función de la pérdida o la precisión, también permite ver histogramas con los pesos, los sesgos y cómo los tensores van cambiando a lo largo del tiempo del entrenamiento. Es una forma sencilla de visualizar el progreso y hacer un seguimiento[20].

Otra de las ventajas que nos ofrece el uso de callbacks durante el entrenamiento de redes neuronales artificiales es el uso de funciones de finalización temprana o *early stopping*. Esta función nos permite finalizar el entrenamiento automáticamente cuando una métrica monitorizada ha dejado de mejorar durante un cierto número de ejecuciones[21]. Si por ejemplo la función de pérdida no mejora en x durante 3 epochs, el entrenamiento se interrumpe, aunque no haya completado todos los epochs que se definieron la inicio del mismos.

Por lo tanto, el entrenamiento quedaría de la siguiente forma:

```
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=50, verbose=1, callbacks=[tensorboard_callback])
```

Figura 33 - Entrenamiento del modelo durante 50 EPOCHS

Un ejemplo de ejecución dentro de Jupyter Notebook es la siguiente:

[19] TensorFlow, 2023
[20] TensorFlow, s. f.
[21] Keras, s.f.d

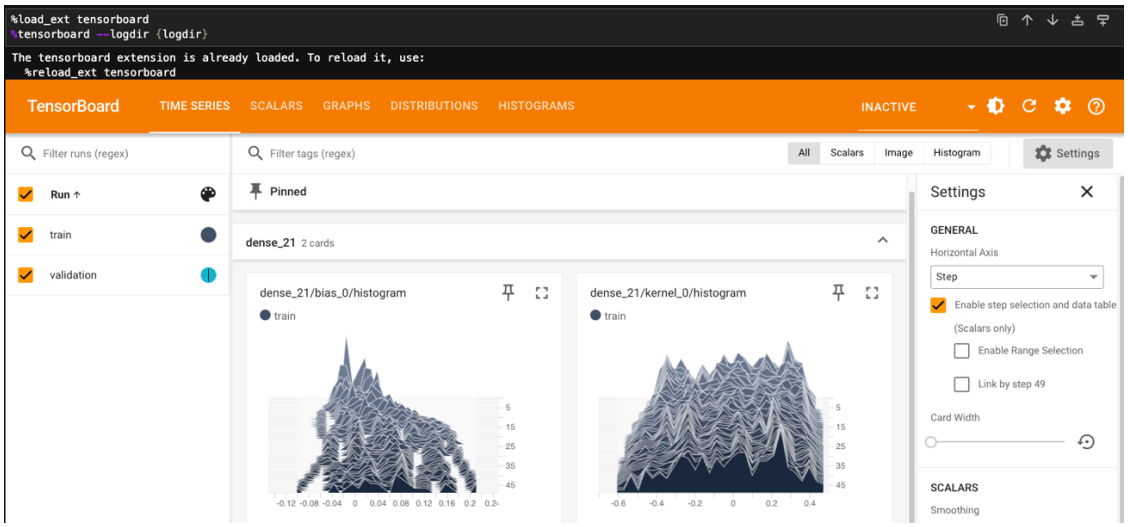


Figura 34 - Ejecución del panel de TensorBoard en Jupyter tras el entrenamiento

Y el resumen del entrenamiento de la red es el siguiente:

```
----------    Neural networks    ----------
################ RESUMEN ################
# Sensibilidad: 0.8356                  #
# Especificidad: 0.7381                 #
# Area Under ROC Curve: 0.8663          #
```

Figura 35 - Resumen de entrenamiento del modelo Sequential (basado en redes neuronales)

También se ha codificado un gráfico que visualice la curva ROC y el área bajo la misma (AUC), como se observa en la Figura 36. A modo de resumen, la métrica AUC es una forma muy útil de evaluar el rendimiento de un modelo. Se calcula bajo una curva ROC o Receiver Operator Curve que es una forma de visualizar la precisión de un problema de clasificación. Representa la sensibilidad frente a la especificidad.

Es el punto de partida para calcular el área bajo la curva o area under the curve (AUC). Este provee una medida útil para aceptar o descartar una prueba o un modelo. Cuanto mayor sea el AUC (más cercano a 1) más útil será el modelo, pues implicará que el modelo es perfecto, clasifica correctamente sin fallo todos los valores de entrada. En contrapartida, cuanto menor sea (más cercano a 0) peor clasificará el modelo.

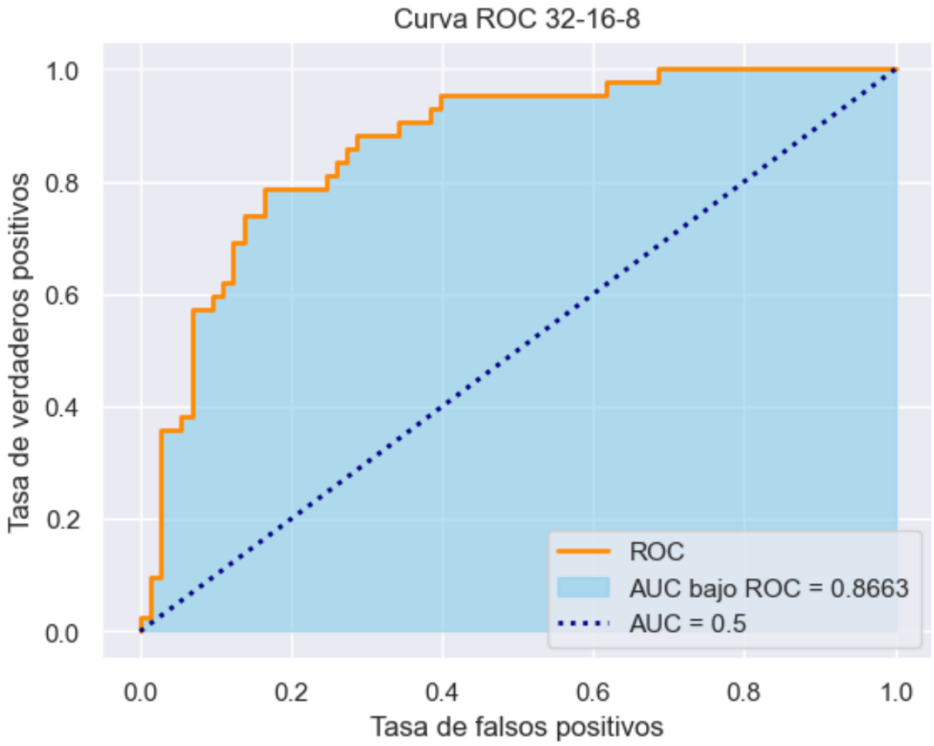


Figura 36 - Resultado de AUC tras el entrenamiento

Ningún modelo deberá aceptarse como válido con AUC menor que 0.5, esto sugeriría que el modelo clasifica peor que una clasificación aleatoria. Es una métrica bastante robusta de debido a su invarianza de escala e invariable respecto al umbral de clasificación[22].

Otra de las métricas interesantes para comprender y evaluar mejor cómo el modelo se comporta con nuevos datos o cómo se ha llevado a cabo el entrenamiento. Se visualiza el progreso de la función de pérdida tanto en entrenamiento como en validación, en la Figura 37.

Como se puede ver, para el conjunto de entrenamiento, la función de pérdida disminuye rápidamente, lo que implica que el modelo está aprendiendo bien. En cambio, la función de pérdida del conjunto de validación no decrece al mismo ritmo, lo que puede indicar la presencia de sobreajuste en el modelo.

Otra de las métricas más utilizadas en el contexto de entrenamiento de modelos basados en inteligencia artificial, es la matriz de confusión. Permite agrupar varias métricas y revela de forma directa el rendimiento del modelo, tanto para bien como para mal. Un ejemplo de matriz de confusión se puede apreciar en la figura X, nos da información muy útil sobre los fallos que el clasificador está cometiendo y además nos permite ver de forma clara y rápida los errores estadísticos (tipo I y II), Figura 38.

[22] González, 2023

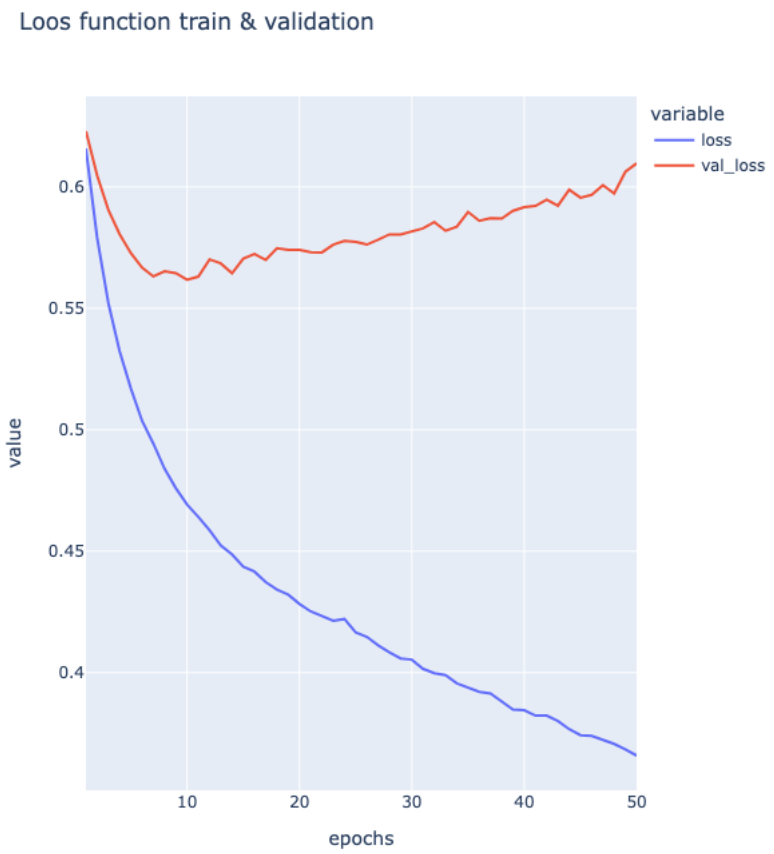


Figura 37 - Gráfica de la función de pérdida a lo largo del entrenamiento

Figura 38 - Matriz de confusión (elaboración propia)

La matriz de confusión del modelo basado en redes neuronales se observa en la Figura 39.

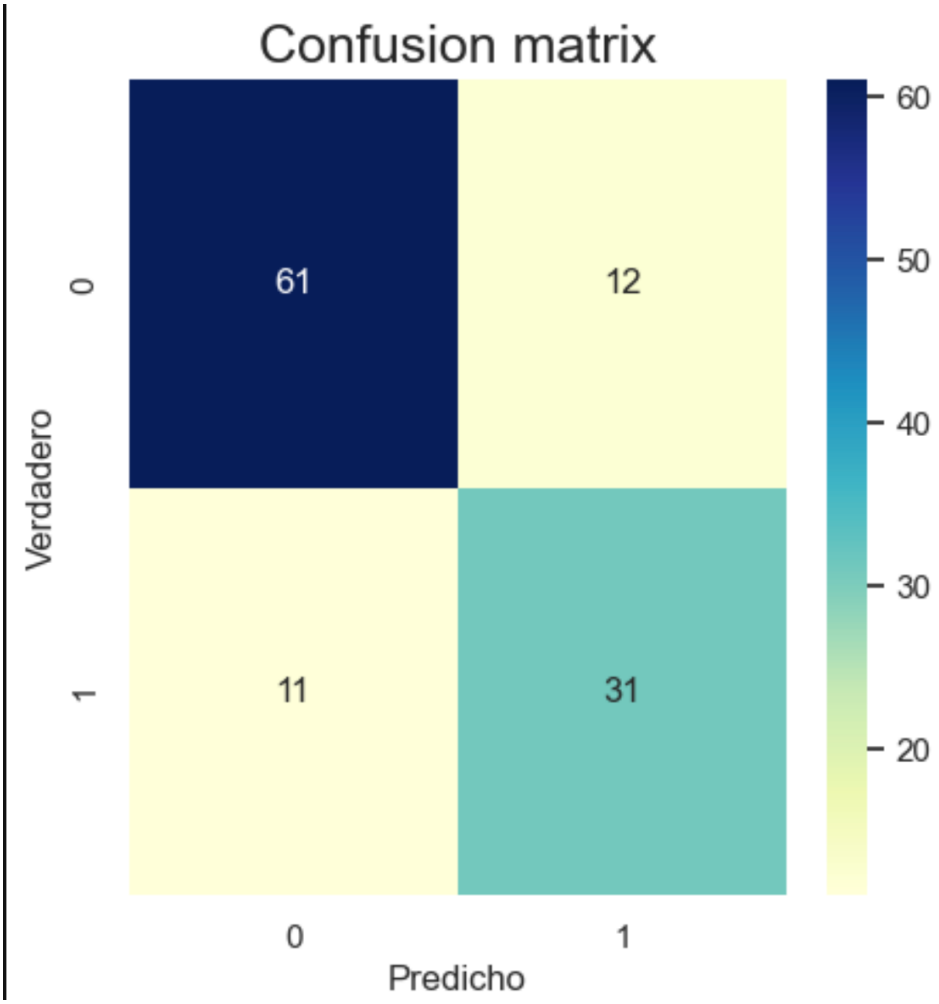


Figura 39 - Matriz de confusión resultante del entrenamiento del modelo Sequential

5. Conclusiones

Sin duda el potencial transformador de la inteligencia artificial en salud es muy alto. En medicina de precisión se considera desarrollada la tecnología en un nivel de madurez emergente e impacto previsto transformacional en los próximos 5-10 años, con actores como Adhera Health o Tempus. El análisis del lenguaje para el diagnóstico clínico también se considera en un nivel de madurez emergente e impacto potencial alto, en este campo se puede detectar de forma no invasiva anomalías clínicas. O el uso de sistemas de ayuda automatizados para la toma de decisiones del paciente, con el objetivo de ayudar a los mismos a decidir sobre las alternativas posibles. Se encuentra en el mismo nivel de madurez e impacto previsto[23].

[23] CEMP, s.f.

El problema que vertebra el presente trabajo versa acerca de este tipo de enfoques basados en inteligencia artificial en salud dentro del paradigma de impacto potencialmente alto como es la medicina de precisión. En concreto, se ha propuesto una comparación entre varias estrategias basadas en ML y DL que puedan ayudar a la detección precoz de la diabetes de tipo 2.

Dentro de la gran variedad de algoritmos y estrategias conocidas en el mundo de los algoritmos de clasificación basados en ML, se ha seleccionado dos de los más usados y conocidos en el sector: bosque aleatorio (random forest, RF) y máquinas de vectores de soporte (support vector machines, SVM). Un RF es una estrategia basada en la creación de múltiples árboles de decisión donde cada uno se entrena de forma independiente. La predicción final se construye con la combinación de los resultados de los árboles que forman el bosque. Una SVM es un enfoque de ML basado en aprendizaje supervisado, donde se busca encontrar el hiperplano óptimo o hiperplano marginal máximo, que mejor separe las clases a clasificar.

En el caso de algoritmos de clasificación basados en DL, se ha elegido uno de los modelos computacionales más populares y que mejores resultados han demostrado recientemente: las redes neuronales artificiales, en concreto, se ha usado una red de tipo perceptrón multicapa, modelo basado en neuronas humanas compuesto por nodos o neuronas y capas de entrada, ocultas y de salida. La capa de entrada introduce los patrones de entrada en la red, las capas ocultas procesan las entradas mediante funciones de activación y la capa de salida produce las salidas de toda la red con otro tipo de función de activación.

En resumen, los resultados obtenidos con los modelos propuestos anteriormente son los que se muestran en las Figuras 35-37. El modelo basado en redes neuronales es el que ha obtenido mejores datos en cuanto a AUC, sin embargo, la especificidad de este modelo podría mejorarse. Por tanto es el modelo elegido, y es que puede usarse para hacer predicciones sobre nuevos datos, aunque pueden realizarse mejoras tanto en los hiperparámetros del modelo como en el pre-procesado y limpieza de los datos, aunque sin duda, lo que más beneficiaría al modelo en términos de generalización es el aumento del número de pacientes usados para entrenar al modelo.

En un posible trabajo futuro, este modelo (y versiones mejoradas) podría utilizarse para potenciar un sistema basado en IA dentro de los centros hospitalarios. Así poder actuar como un sistema de soporte a toma de decisiones clínicas. El fin de aplicar este tipo de técnicas a la sanidad (y a la atención hospitalaria que se le brinda al paciente) es el de mejorar los diagnósticos, tanto en tiempo, pues se harían de forma temprana beneficiando al tratamiento de la posible enfermedad, como en precisión. Ya que este tipo de sistemas puede ofrecer (o esa es la promesa), eventualmente, una precisión en los diagnósticos superior al humano.

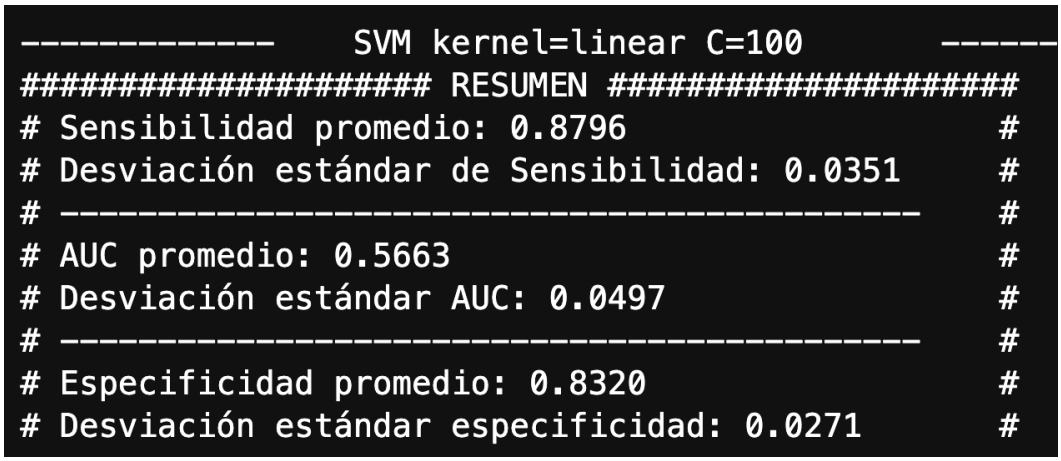


Figura 40 - Resumen de entrenamiento del modelo Random Forest

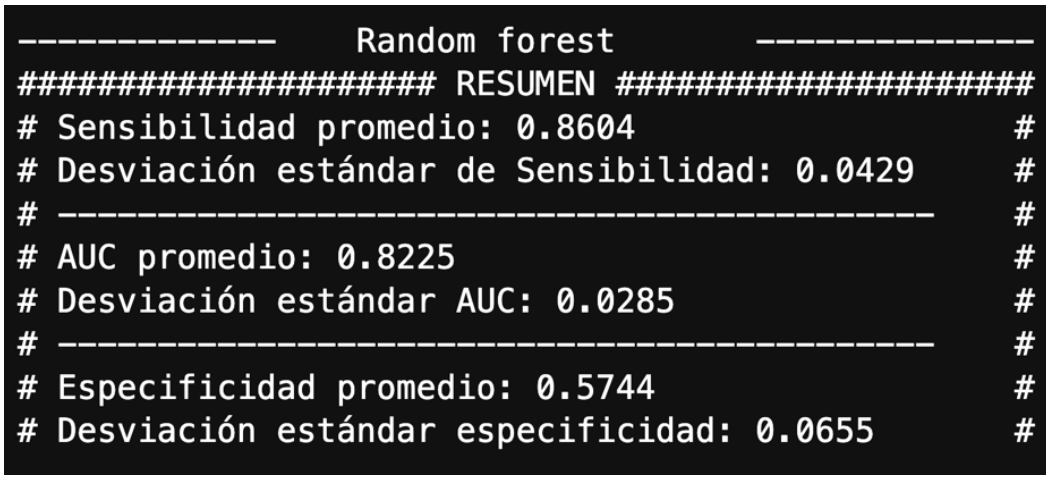


Figura 41 - Resumen de entrenamiento del modelo SVM

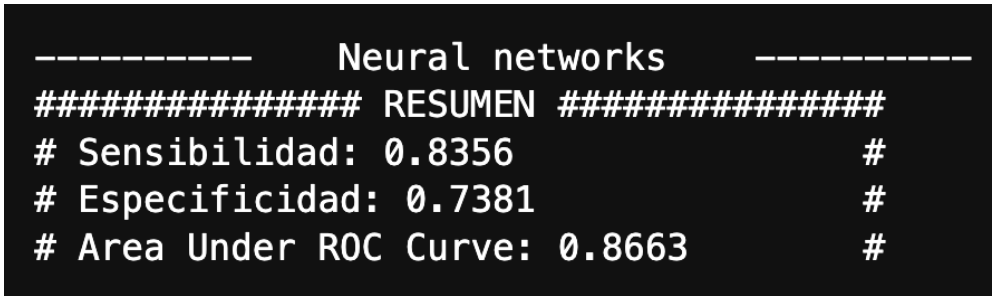


Figura 42 - Resumen de entrenamiento del modelo Sequential (red neuronal)

6. Bibliografía

Fourth Industrial Revolution. (2020, 7 abril). World Economic Forum. Recuperado de https://www.weforum.org/focus/fourth-industrial-revolution

La medicina de personalizada en diabetes: una oportunidad de mejora que empuja a modificar la clasificación actual de la diabetes. Sociedad Española de Diabetes. (2023, 21 abril). Recuperado de https://www.sediabetes.org/comunicacion/sala-de-prensa/la-medicina-de-personalizada-en-diabetes-una-oportunidad-de-mejora-que-empuja-a-modificar-la-clasificacion-actual-de-la-diabetes/

La diabetes en España (y en el mundo): del aumento de casos al infradiagnóstico, pasando por los déficits en educación terapéutica. Sociedad Española de Diabetes. (2023, 19 abril). Recuperado de https://www.sediabetes.org/comunicacion/sala-de-prensa/la-diabetes-en-espana-y-en-el-mundo-del-aumento-de-casos-al-infradiagnostico-pasando-por-los-deficits-en-educacion-terapeutica/

¿Qué es la diabetes? | Información básica | Diabetes. Centro para el Control y Prevención de Enfermedades. (2022, 13 de julio). Recuperado de https://www.cdc.gov/diabetes/spanish/basics/diabetes.html#:~:text=Existen%20tres%20tipos%20principales%20de,(diabetes%20durante%20el%20embarazo)

Biosensores y su salud. (2023, 13 de marzo). NIH, National Health Institute. Recuperado de https://salud.nih.gov/recursos-de-salud/nih-noticias-de-salud/biosensores-y-su-salud

Fuentes, R. M. D. (2017). *Avances en medición de glucosa: del glucómetro tradicional al sistema flash*. Dialnet. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=6003937

Bhatia, G., Dutta, P. K., & McClure, J. (2022, 15 julio). Los datos, gráficos y mapas más recientes a nivel global sobre el coronavirus. Reuters. Recuperado de https://www.reuters.com/graphics/world-coronavirus-tracker-and-maps/es/

España es el segundo país con mayor prevalencia de diabetes de Europa. Sociedad Española de Diabetes. (2021, 12 noviembre). Recuperado de https://www.sediabetes.org/comunicacion/sala-de-prensa/espana-es-el-segundo-pais-con-mayor-prevalencia-de-diabetes-de-europa/#:~:text=La%20prevalencia%20de%20la%20diabetes,tasa%20m%C3%A1s%20alta%20de%20Europa

World Health Organization: WHO. (2021, junio 28). La OMS publica el primer informe mundial sobre inteligencia artificial (IA) aplicada a la salud y seis principios rectores relativos a su concepción y utilización. *World Health Organization*. Recuperado de https://www.who.int/es/news/item/28-06-2021-who-issues-first-global-report-on-ai-in-health-and-six-guiding-principles-for-its-design-and-use

Chollet, F. (2019, p.26). Deep Learning with Python. ANAYA.

scikit-learn developers. (s. f.). sklearn.preprocessing.StandardScaler — scikit-learn 1.3.2 documentation. scikit-learn. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Mosbah, H., & El-Hawary, M. (2017). Optimization of neural network parameters by stochastic fractal search for dynamic state estimation under communication failure. *Electric Power Systems Research*, *147*, 288-301. Recuperado de https://doi.org/10.1016/j.epsr.2017.03.002

KeepCoding, R. (2023, 3 octubre). Importancia de los kernels para los SVM. *KeepCoding Bootcamps*. Recuperado de https://keepcoding.io/blog/importancia-de-los-kernels-para-los-svm/#Kernels_para_los_SVM

scikit-learn. *Plot classification boundaries with different SVM kernels*. (s. f.). https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#linear-kernel

Keras. (s. f.). *The Sequential model*. Keras. Recuperado de https://keras.io/guides/sequential_model/

Keras. (s. f.b). *Dense Layer*. Recuperado de https://keras.io/api/layers/core_layers/dense/

Keras. (s. f.c). *Callbacks API*. Recuperado de https://keras.io/api/callbacks/

Keras (s. f.d). *EarlyStopping*. Recuperado de https://keras.io/api/callbacks/early_stopping/

TensorFlow. (2023, 27 septiembre). *tf.keras.losses.BinaryCrossentropy*.Recuperado de https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy

TensorFlow. (s. f.). Recuperado de https://www.tensorflow.org/tensorboard?hl=es-419

Gonzalez, L. (2023, 2 mayo). *Curvas ROC y área bajo la curva (AUC)*. 🤖Aprende IA. Recuperado de https://aprendeia.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning/

CEMP. (s. f.). *Actores relevantes en sanidad*.

Plotly. (s.f.). Recuperado de https://plotly.com/python/getting-started/

7. Material suplementario

Enlace al repositorio público de GitHub donde está alojado el código:
https://github.com/luismerinou/CEMP-TFM-MASTER-IA-2023/

Los gráficos interactivos están implementados en Plotly, debido a ello, se recomienda abrir el archivo .ipynb directamente desde Jupyter Notebook. Al abrirlo desde Google Colab, solo se visualizarán los gráficos estáticos.