UNIVERSIDAD CATÓLICA SANTO TORIBIO DE MOGROVEJO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Desarrollo de un sistema inteligente web basado en redes neuronales artificiales para la predicción del riesgo de mortalidad del COVID-19

TESIS PARA OPTAR EL TÍTULO DE
INGENIERO DE SISTEMAS Y COMPUTACIÓN

AUTOR
Sergio Alexander Mondragon Silva

ASESOR
Guadalupe Teresa Lip Curo
https://orcid.org/0000-0002-0353-939X

Chiclayo, 2023

Desarrollo de un sistema inteligente web basado en redes neuronales artificiales para la predicción del riesgo de mortalidad del COVID-19

PRESENTADA POR

Sergio Alexander Mondragon Silva

A la Facultad de Ingeniería de la
Universidad Católica Santo Toribio de Mogrovejo
para optar el título de

INGENIERO DE SISTEMAS Y COMPUTACIÓN

APROBADA POR

Maria Ysabel Aranguri Garcia
PRESIDENTE

Ricardo David Iman Espinoza
SECRETARIO

Guadalupe Teresa Lip Curo
VOCAL

Dedicatoria

El presente trabajo de investigación es dedicado a mi familia que siempre me ha mostrado su apoyo y me han permitido tener la oportunidad de crecer académicamente. Además, a todas las personas que durante el transcurso de esta investigación me mostraron su apoyo y confianza.

Agradecimientos

A mi asesora de tesis por las correcciones recibidas en cada asesoría con el fin de mejorar los resultados y poder lograr los objetivos propuestos.

A la institución médica y a sus colaboradores por permitirme trabajar con la información y brindarme los recursos necesarios.

A mis compañeros de estudio los cuales siempre me brindaron su apoyo.

INFORME DE ORIGINALIDAD

21%	19%	12%	10%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	hdl.handle.net Fuente de Internet	2%
2	tesis.usat.edu.pe Fuente de Internet	1%
3	www.researchgate.net Fuente de Internet	1%
4	cybertesis.unmsm.edu.pe Fuente de Internet	1%
5	Bermejo Guevara Mario Alberto. "Los costos ocultos y su relación con los eventos adversos hospitalarios en la gestión de las instituciones de salud", TESIUNAM, 2016 Publicación	<1%
6	Guevara Castillo Marco Antonio. "Análisis operativo y competitivo de una empresa de caramelo", TESIUNAM, 2022 Publicación	<1%
7	renati.sunedu.gob.pe Fuente de Internet	<1%

Índice

Resumen

El presente trabajo de investigación surge ante el problema mundial de la pandemia del COVID-19, de tal manera, esta investigación tiene como objetivo general el implementar una solución basada en redes neuronales artificiales para predecir el riesgo de mortalidad de la COVID-19 en pacientes infectados, utilizando datos de una clínica de la ciudad de Chiclayo. Así, se decidió utilizar la metodología Scrum para la gestión del proyecto de investigación y para el modelado de la red neuronal artificial (RNA) se siguió la metodología de desarrollo utilizada en la investigación de I. Kaastra y M. Boyd incluyendo ciertas mejoras de la metodología utilizada en la investigación de Abdulaal A et al. En consecuencia, este proyecto generó una herramienta médica la cual es accesible a través de cualquier navegador web y cuenta con dos RNA implementadas las cuales tienen la capacidad de aprender de nuevos registros clínicos ingresados. La mejor RNA implementada tiene una exactitud del 82.72%, AUROC de 88.48%, desviación estándar de 0.0848 y un F1Score de 83.72%; además, este sistema web fue validado para el diagnóstico con las siguientes métricas médicas: 85.71% de sensibilidad. 79.48% de especificidad y un AUROC del 88.48%. Así pues, del desarrollo de esta herramienta se concluye que para identificar la correcta arquitectura e hiperparámetros se deben generar diferentes iteraciones de entrenamiento de la RNA; asimismo, en cada iteración se debe utilizar diferentes combinaciones de arquitectura e hiperparámetros calculando las métricas de exactitud, F-measure, AUROC y desviación estándar. Finalmente, se podrá elegir a la combinación con las mejores métricas

Palabras clave: Redes neuronales artificiales, RNA, COVID-19, inteligencia artificial, aprendizaje automático.

Abstract

The present research work arises from the global problem of the COVID-19 pandemic, thus, the general objective of this research is to implement a solution based on artificial neural networks to predict the mortality risk of COVID-19 in infected patients, using data from a clinic in the city of Chiclayo. Thus, it was decided to use the Scrum methodology for the management of the research project and for the modeling of the artificial neural network (ANN) the development methodology used in the research of I. Kaastra and M. Boyd was followed including certain improvements of the methodology used in the research of Abdulaal A et al. Consequently, this project generated a medical tool which is accessible through any web browser and has two implemented ANNs which have the ability to learn from new clinical records entered. The best implemented ANN has an accuracy of 82.72%, AUROC of 88.48%, standard deviation of 0.0848 and a F1Score of 83.72%; furthermore, this web system was validated for diagnosis with the following medical metrics: 85.71% sensitivity. 79.48% specificity and an AUROC of 88.48%. Thus, from the development of this tool it is concluded that in order to identify the correct architecture and hyperparameters, different iterations of ANN training must be generated; likewise, in each iteration different combinations of architecture and hyperparameters must be used by calculating the metrics of accuracy, F-measure, AUROC and standard deviation. Finally, the combination with the best metrics can be chosen.

Keywords: Artificial neural networks, ANN, COVID-19, artificial intelligence, machine learning.

Introducción

Hasta junio del 2021, el SARS-Cov-2 había afectado a 173 463 517 personas alrededor del mundo con una letalidad del 2.15 % [1] convirtiéndose en un gran problema para la humanidad. Asimismo, Lambert considera en [2], que esta pandemia ha puesto en descubierto la interdependencia global que sustenta la economía y las fallas que este sistema tiene. Del mismo modo en el año 2020, el banco mundial aseguró que la economía mundial se reduciría un 5,2% ese año donde varias economías sufrirán una disminución en su producto per cápita [3]. En consecuencia, diversos países impulsaron las investigaciones científicas para encontrar más respuestas acerca de este virus, así como también aceleraron la creación de vacunas para evitar que este virus siguiera afectando a más personas[4]. Sin embargo, estamos frente una enfermedad muy variable debido a que los factores que llevan a algunos a enfermarse gravemente y a otros no, aún no están bien definidos [5]. Por tal motivo, se realizó una revisión bibliográfica de investigaciones enfocadas en esta enfermedad y se logró concluir distintos factores de riesgo los cuales podemos agrupar en: factores demográficos, comorbilidades, antecedentes personales, síntomas y análisis de laboratorio. Así, entre los factores demográficos encontramos que la edad tiene una influencia clave en la mortalidad de esta enfermedad [5]–[10], donde los adultos mayores tienen mayor riesgo de fallecer. Asimismo, el sexo es otro factor demográfico estudiado en [5]–[7], [9], [10] donde se afirma que los varones tienen mayor riesgo de mortalidad. Por otro lado, tenemos las comorbilidades estudiadas en [5], [7], [8] como la diabetes, hipertensión, enfermedades respiratorias, enfermedades cardiacas crónicas. Otro aspecto importante, son los antecedentes personales del paciente como el historial de tabaquismo [5], [6], [8] y los trasplantes de corazón [6]. Con relación a los síntomas que afectan la mortalidad según [6], [9] tenemos a la: fiebre, tos, disnea, fatiga, anorexia, mareos, náuseas y escalofríos. Por último, según Yang et al en [9] los análisis de laboratorio también arrojan valores influyentes para evaluar la mortalidad de los pacientes antes del tratamiento, tales como, el aumento en el recuento absoluto de leucocitos, procalcitonina (PCT), dímero D, lactato deshidrogenasa (LDH), ferritina y la disminución en el recuento absoluto de linfocitos.

Otro aspecto característico del SARS-Cov-2 es su alto nivel de propagación y su capacidad para mutar rápidamente complicando el análisis sobre los factores de riesgo y síntomas de esta enfermedad. Según Kuhn en [11], el SARS-Cov-2 ha mutado rápidamente debido a que es un virus ARN; esta familia de virus altera su material genético a mayor velocidad que los virus ADN. Sin embargo, estas alteraciones solo son posibles mientras más personas se encuentren infectadas, así cuando una nueva población es expuesta a este virus puede surgir una nueva variante. De este modo, apareció la variante de Reino Unido (conocida como B.1.1.7) que parece se puede unir más fácilmente a la superficie de las células humanas permitiéndole así ser transmitida rápida y eficientemente. Además, existe la variante Sur Africana (conocida como 502.V2) que presenta alteraciones en su proteína spike proporcionándole la facilidad de evadir la respuesta de los anticuerpos.

Durante el planteamiento de este proyecto en el 2021 el Perú no había sido ajeno a este virus contando para junio de dicho año con un número de casos de 1 983 570 personas con un índice de letalidad del 9.40 % [12] ubicándose entre los 20 primeros países del mundo con mayor número de contagiados según el Dashboard presentado por la Universidad Johns Hopking [1]. Según el MINSA en [12], los más vulnerables a esta enfermedad fueron los adultos mayores con 130 271 fallecidos seguidos por los adultos con 53 054 fallecidos, además el sexo masculino tenía la mayor cantidad de fallecidos a comparación del sexo femenino representando el 63.79% y 36.21% respectivamente. Asimismo, nuestra región se encontraba en la tercera posición en cuanto a mortalidad (14. 11 %) y el número de contagiados hasta junio del 2021 asciende a 56 144 [12]. Si bien la mortalidad es baja a comparación de otras enfermedades no debemos olvidarnos de que cada cifra representa a una vida humana invaluable y debemos hacer todo lo posible para preservarla. Otra consecuencia de esta enfermedad es el colapso del sistema médico de nuestra región debido a que los implementos médicos son escasos; la red asistencial de Lambayeque cuenta con 52 camas UCI [13], no obstante, han sido insuficientes en varias ocasiones [14], [15].

Por todo lo anteriormente explicado, debemos intentar estar un paso adelante en la lucha contra este virus con los recursos disponibles llegando a formular la

siguiente pregunta: ¿De qué manera podemos hacer accesible una precisa predicción del riesgo de mortalidad en pacientes infectados con el COVID-19 en un centro médico de la ciudad de Chiclayo?

Un recurso del cual podemos hacer uso, debido a su capacidad de procesamiento de datos y su capacidad de aprender para generar modelos predictivos son las redes neuronales artificiales [16]. Las redes neuronales artificiales (RNA), son utilizadas para la predicción en diversos campos como las apuestas de videojuegos [17], la mecánica [18], [19], el mercado bursátil [20], la medicina [16], [21], [22], entre otros. Las RNA han sido utilizadas para el pronóstico de contagios de la COVID-19 analizando los datos de la Universidad de Johns Hopkings obteniendo una precisión del 87.70% [23]. Por consiguiente, el objetivo de esta investigación es implementar una solución basado en redes neuronales artificiales para predecir el riesgo de mortalidad de la COVID-19 en pacientes infectados, utilizando datos de una clínica de la ciudad de Chiclayo. De lo anterior, necesitamos primero identificar las variables que influyen el riesgo de la COVID-19 para modelar una RNA precisa, luego identificar la arquitectura y los hiperparámetros de la RNA para demostrar la validez del modelo y por último implementar un modelo de RNA eficiente para demostrar la validez de la solución al diagnosticar el riesgo de mortalidad de un paciente infectado con COVID-19.

La presente investigación del tipo Desarrollo Experimental con diseño experimento puro con posprueba únicamente y grupo de control se aplicó en la ciudad de Chiclayo eligiéndose para esta investigación como caso de estudio un establecimiento especializado en el tratamiento de pacientes con COVID-19, el propósito de utilizar los datos de esta entidad local fue contextualizar de una mejor manera las características que presenta la enfermedad en esta ciudad. Así, según un análisis inicial de los datos disponibles, se observaron pacientes atendidos desde mayo del 2020 a enero del 2022 este establecimiento ha atendido 3471 pacientes con COVID-19. Por consiguiente, se analizaron los datos de las historias clínicas electrónicas para poder definir las características iniciales que se le suministrará a nuestra RNA y generar un modelo matemático de predicción. El sistema predictivo web usó a las redes neuronales artificiales

entrenadas y seleccionadas para generar predicciones del riesgo de mortalidad en un paciente infectado con COVID-19, con el fin de ser un instrumento que ayude al personal médico para tomar mejores decisiones y gestionar mejor los recursos escasos.

La investigación se justifica desde el aspecto científico porque según la Unesco [4] “La pandemia de COVID-19 nos sirve para tomar conciencia de la importancia de la ciencia tanto para la investigación como para la cooperación internacional.” Lo cual nos explica que existe la necesidad de investigaciones que nos ayuden a combatir esta enfermedad. Además, esta investigación plantea la implementación de un software basado en nuevas tecnologías para promover nuevas líneas de investigación e innovación. Por otro lado, desde el aspecto financiero nuestro estado posee insuficientes recursos médicos tanto materiales como humanos [14]. Por tal motivo, es necesario predecir el comportamiento de la enfermedad sobre los pacientes infectados, y así realizar trabajos específicos sobre dichos pacientes para evitar la hospitalización o requerir intervenciones costosas para el estado. Asimismo, desde el aspecto social Kuhn en [11] afirma que el SARS-CoV-2 es altamente contagioso y evoluciona rápidamente, lo que ocasiona que muchas personas se vean afectadas por este virus. Así, el presente proyecto de investigación servirá de apoyo para que los especialistas puedan identificar prontamente a los pacientes infectados más vulnerables y proveerles el adecuado tratamiento para prevenir la mortalidad. Finalmente, desde el aspecto tecnológico es necesario la aplicación de nuevas tecnologías para procesar los datos y comprenderlos, por tal motivo, se busca utilizar redes neuronales artificiales debido a su gran precisión en trabajos de investigación relacionados con diagnósticos médicos [17], [22]–[26].

Revisión de literatura

Antecedentes

Antecedentes internacionales

Abdulaal A et al. en [24] nos indica la gravedad del SARS-Cov-2 y cómo afecta a la salud pública de Inglaterra; además, afirma que aún no existen modelos validados de pronósticos o sistemas de puntuación específicos para los pacientes afectados con SARS-CoV-2, por lo cual el principal objetivo es la creación de un sistema de puntuación sobre el riesgo de mortalidad en el punto de admisión usando la tecnología de redes neuronales artificiales.

En la metodología utilizadas para esta investigación [24] se utilizaron datos de 398 pacientes obteniendo de cada uno 22 características de entrada para la RNA, asignándose los datos de 318 pacientes para el conjunto de entrenamiento y los 80 pacientes restantes para el conjunto de prueba. Posteriormente, se realizó el preprocesado de datos donde se definieron los factores posibles de pronóstico como los datos demográficos, comorbilidades, historial de tabaquismo y síntomas presentados.

Para concluir, el trabajo de investigación [24] se relaciona con la investigación realizada por los siguientes motivos: nos muestra una metodología probada para la creación del modelo de una RNA, extrapolable a nuestra realidad utilizando datos demográficos propios de nuestra ciudad; nos expone los indicadores de sensibilidad y especificidad, que se utilizaron para la evaluación de nuestra RNA especializada; nos demuestra que no es necesario una gran cantidad de datos para que la RNA tenga valores por encima del 80% de precisión; nos indica debemos utilizar el dropout en ambas capas ocultas, así se evita el sobreajuste en nuestro modelo de RNA..

Un segundo trabajo de Michał Wieczorek en [23] nos informa que la predicción de la propagación de los virus es muy importante para poder tener planes de acción adecuados. En este trabajo, nos explica que se han aplicado diversas

metodologías de predicción utilizando machine learning, modelos matemáticos, redes euclídeas, predictores estocásticos con el afán de generar herramientas para entender mejor la propagación del SARS-Cov-2. Estas herramientas fueron aplicadas en distintas partes del mundo y de forma independiente, por tal motivo, en [23] plantean una solución más compleja y global utilizando RNA debido a su flexibilidad de adaptarse a los cambios. El modelo de la RNA propuesta cuenta con 1 capa de entrada, 6 capas ocultas y una capa de salida; este modelo fue entrenado con el algoritmo NAdam utilizando información normalizada de diversos países del mundo. Además, esta investigación tiene como resultados porcentajes de predicción elevados plasmados en tablas donde se muestran los datos de entrenamiento y prueba de la RNA.

Se tomó en cuenta este trabajo debido a que nos muestran los diferentes algoritmos de optimización del entrenamiento de la RNA, de los cuales se escogió el que demuestra ser el más eficiente con los datos propios de nuestra investigación.

Un tercer trabajo es el de Steed et al. En [26] donde se nos describe la problemática en torno a la detección de la gravedad de pacientes que sufren de hemorragia subaracnoidea aneurismática. De acuerdo con esta gravedad se realizan procedimientos diferentes, en consecuencia, es necesario poder predecir adecuadamente. Así, en esta investigación se generaron modelos basados en RNA a partir de la base de datos disponible de un centro médico (855 registros). Se obtuvieron tres modelos de acuerdo con el procedimiento médico que se debería aplicar: desvío prolongado del líquido cefalorraquídeo (LCR), retraso cerebral isquemia y vasoespasmo. Por otro lado, se generó (con los mismos datos) un cuarto modelo de predicción de la disposición categórica al momento del alta (hogar, rehabilitación subaguda, rehabilitación aguda, cuidados intensivos a largo plazo, caducidad en el hospital). Todos los modelos fueron validados por la métrica AUROC con valores mayores al 0.98.

La anterior investigación tiene relevancia con la presente tesis, puesto que, se muestra que los modelos de clasificación deben ser puntuales en lo que se desea predecir, y que pueden ser validados por la AUROC. Por tal motivo, la presente

tesis limita la predicción en el riesgo de mortalidad de un paciente de acuerdo con la probabilidad que existe de fallecer y es validado por la AUROC.

Antecedentes nacionales

Como primer antecedente nacional tenemos a la tesis llevada a cabo por Aguilar et al [27], en esta tesis se abarca la problemática de deserción de clientes en bancos y si es posible identificar con anticipación cuál es el nivel de riesgo de deserción de un cliente microcrédito en la banca de la ciudad de Lima. Por lo tanto, se desarrolló un sistema inteligente basado en un modelo híbrido para mejorar esta identificación. Así, este modelo híbrido combina las técnicas de Máquinas de soporte vectorial (SVM), Árboles de decisión (DT) y Redes neuronales artificiales (ANN) logrando 97.38% en la tasa de acierto. La investigación de Aguilar et al en [27] es relevante con la presente tesis, ya que, nos muestra que para realizar un modelo de predicción de riesgo (en este caso relacionado con la banca) es importante seleccionar las variables a través de revisiones bibliográficas. Además, en el desbalance de datos con respecto a la variable objetivo propone dos técnicas de balanceo: Random Under Sampling y SMOTE; en esta investigación también existió este desbalance y se tomó de referencia estas técnicas.

Como segundo trabajo de investigación nacional tenemos al desarrollado por Carrión en [28] donde se describe la problemática de la escasez de mineral por lo que es necesario mejorar y optimizar los procesos de recuperación de mineral. Debido a eso, Carrión planteó el uso de la inteligencia artificial para predecir variables para una eficiente recuperación de estos minerales y así poder tomar mejores decisiones. Por tal motivo, se utilizó la herramienta NeuraTools para la creación de la RNA y se entrenó el modelo con 348 datos. La investigación de Carrión en [28] es relevante en la presente tesis debido a que nos muestra una nueva herramienta para la creación de RNA.

Como tercer trabajo nacional se ha tenido en cuenta la investigación de Nazario [29] la cual aborda la problemática de los costos presupuestales, costos de

tiempo y costos de recursos humanos que derivan de la elaboración de proyectos de ley que pueden resultar inviables. En consecuencia, se plantea predecir la viabilidad legislativa de las propuestas parlamentarias con el fin de reducir los costos anteriormente mencionados. Nazario desarrolló una RNA con una precisión del 96.1% siguiendo la metodología CRISP-DM, además se contó con una muestra de 377 casos y se utilizó la herramienta SPSS.

El anterior trabajo es relevante en la investigación debido a que utilizan como función de costes a la entropía cruzada la cual se utilizó en el entrenamiento de las redes neuronales artificiales de la presente tesis.

Antecedentes locales

Garcia en [30] nos narra el problema de la morosidad crediticia, debido a esto plantea un sistema de información basado en redes neuronales con el fin de predecir los riesgos de otorgar créditos personales. En la metodología, nos muestra la limpieza de los datos previos al preprocesado de datos y una tabla de las variables que han sido consideradas como valores de entrada de la RNA. Por otro lado, se utiliza una matriz de confusión, una tabla de precisión y una gráfica de la curva ROC para evaluar el modelo de RNA; en la evaluación de la RNA logra una precisión del 87.88%.

Este trabajo es referente con la investigación planteada, puesto que muestra algunos indicadores para medir la precisión de una RNA.

Por otro lado, Luis García en [31] nos indica la existencia de un bajo desempeño académico influenciado por factores psicológicos ante lo cual plantea un sistema predictivo para poder medir el rendimiento académico. Así, lo que se busca principalmente es definir todos los factores (psicológicos y académicos) posibles que influyan en el rendimiento académico del estudiante. El autor utiliza la metodología de CRISP-DM para el desarrollo del modelo de minería de datos mientras que para el sistema utiliza la metodología RUP. Luis García, utilizó dos algoritmos para su sistema predictivo: Random Forest y Decision Tree. Asimismo, para el entrenamiento de sus modelos utilizó 360 datos con 42 variables cada una logrando una predicción del 95% con el algoritmo de Random Forest.

De este modo, la anterior investigación se tomó en cuenta, ya que se trabaja con un número no muy elevado de datos y se obtiene una alta grado de precisión. Además, se nos muestra la importancia del análisis detallado de las variables que pueden influir en un modelo de predicción.

Monteza en [32] nos narra la problemática en torno a la selección de catequistas donde no existe un correcto diagnóstico, así plantea un sistema de diagnóstico de perfiles psicotécnicos. Para el desarrollo de este sistema se utilizó el algoritmo de regresión lineal múltiple y se clasificaron las variables en: habilidades, actitudes y conocimiento. El sistema fue validado por expertos obteniéndose una 93% de acierto.
Esta tesis se tomó en cuenta debido a que se nos muestra la influencia de los expertos para la definición de las variables y la comprobación de los resultados. De esta forma, para el desarrollo de la presente tesis se realizaron distintas reuniones con los expertos médicos para poder definir las variables de predicción.

Bases teórico-científicas

A continuación, mostraré los principales conceptos que se tuvieron en cuenta para el desarrollo de la presente investigación:

Inteligencia artificial

De acuerdo con Amador [33], definir la inteligencia artificial (IA) es un reto debido a la dificultad de conceptualizar la inteligencia, de este modo muchos investigadores de IA tienen diferentes conceptos sobre la inteligencia. La inteligencia se puede considerar como la capacidad de respuesta de un ser inteligente ante determinadas circunstancias [34]. Por otra parte, la inteligencia también se define como la capacidad global de un individuo que piensa racionalmente y se relaciona eficazmente con su entorno para lograr sus objetivos. Por consiguiente, de estos dos conceptos se concluye que determinar sin un ser es inteligente o no depende del enfoque. Del primer concepto se

resalta que la inteligencia es responder eficientemente ante un problema, y del segundo concepto que la inteligencia es pensar racionalmente.

Citando a Banda [35] la primera definición de inteligencia artificial fue propuesta en la Conferencia de Dartmouth: “Hacer que una máquina se comporte como lo haría un ser humano, de tal manera que se la podría llamar inteligente.” No obstante, en la actualidad el área de la inteligencia artificial abarca e influye en muchos más campos del conocimiento sin limitarse a estudiar solamente el comportamiento de las máquinas; la IA abarca la resolución de teoremas matemáticos, la detección de imágenes, el procesamiento del lenguaje, entre otras aplicaciones. Asimismo, Banda [35] indica que la IA abarca tanto la investigación científica como la tecnológica de los sistemas inteligentes; considera que un sistema inteligente tiene capacidad para aprender y poder tomar decisiones actuando racionalmente guiado por sus metas en un entorno dado.

Uno de los ámbitos que comprende la Inteligencia Artificial es la Ingeniería Neuronal; Banda en [35] afirma que la ingeniería neuronal consiste en reproducir las distintas funcionalidades de los modelos neuronales para la construcción de máquinas inteligentes. A continuación, se mostrarán los modelos neuronales:

Modelo neuronal biológico

Según Escobar en [36], el sistema nervioso está constituido en gran parte por redes neuronales que permiten la conexión local y a distancia dentro de este sistema; el elemento básico de la red neuronal es la neurona [37]. De acuerdo con Banda en [35] “Una neurona es una célula altamente especializada en el proceso de información”, es decir, las neuronas son capaces de procesar datos en información. Asimismo, la neurona está compuesta morfológicamente de tres componentes resaltantes: el soma, las dendritas y el axón. Por otro lado, Nacelle en [37] nos explica el proceso de comunicación entre las neuronas la cual está basada en la sinapsis (la unión entre el axón y la dendrita de dos neuronas); esta comunicación se basa en la activación de las terminales bioquímicas en el extremo del axón que son captados por la dendrita. Además, de acuerdo con la

experiencia en los estímulos produce que la sinapsis correspondiente se refuerce de forma excitatoria o inhibitoria.

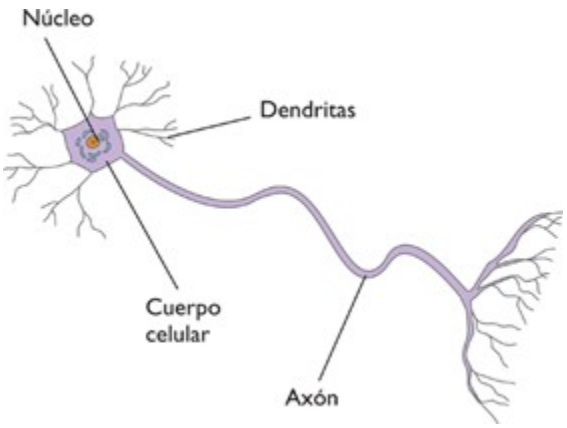


Fig. 1. Esquema elemental de una neurona

Fuente: Extraído de [37]

McCulloch y Pitts propusieron un modelo neuronal artificial simple que consiste en un elemento que consta de una función de activación tipo escalón unitaria [35].

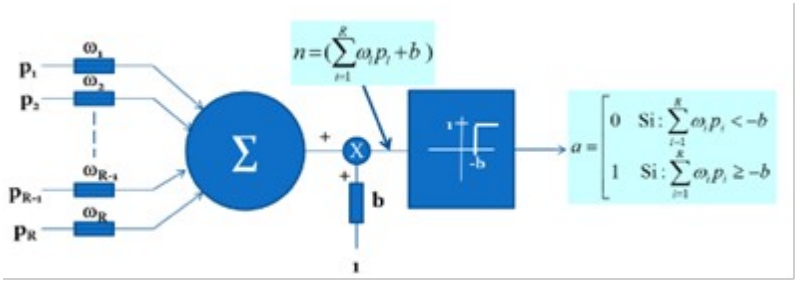


Fig. 2. Modelo de una neurona propuesta por McCulloch y Pitts

Fuente: Extraído de [35]

Como afirma Banda en [35], el modelo planteado presenta diversas analogías al modelo biológico: las dendritas y axones equivalen a los cables o conexiones; la sinapsis, a los pesos de ponderación de las conexiones; y la actividad neuronal, a el umbral de activación de la neurona. Además, el signo de los pesos sinápticos emula el proceso de inhibición (signo negativo) y activación (signo positivo) dentro de una red neuronal.

Aprendizaje de una red neuronal

Del resultado de observaciones de experimentos neurobiológicos se dedujo el proceso de aprendizaje de una red neuronal; este proceso demuestra que, el valor de la sinapsis se incrementa mientras más veces ocurra la activación simultanea

de las neuronas de ambos lados. Al principio anteriormente mencionado se le conoce como el postulado de Aprendizaje Hebbiano [38].

Como señala Banda en [35] red neuronal aprende adaptando los pesos en sus conexiones sinápticas disminuyendo el error entre la salida incorrecta y la salida correcta producida ante una entrada. En primer lugar, la red neuronal es expuesta a estímulos externos de su entorno; luego, la red neuronal sufre cambios como resultado de los estímulos; finalmente, la red neuronal es capaz de responder al entorno de acuerdo con los cambios ocurrido en su estructura interna.

Prueba diagnóstica médica

Las pruebas diagnósticas médicas están estrechamente relacionadas con los estudios epidemiológicos para clasificar a los sujetos de estudio como sanos o enfermos de una determinada enfermedad [39]. Además, las buenas pruebas diagnósticas deben ser válidas, reproducibles y seguras [40]; por tal motivo se pueden agrupar en:

- **Indicadores que determinan la validez:**

  Pita en [40] considera los siguientes indicadores:

  - **Sensibilidad**

    Se define como la probabilidad de clasificar correctamente a un individuo enfermo, en otras palabras, es la razón entre los individuos enfermos con un resultado positivo en la prueba y la cantidad total de individuos enfermos.

$$Sensibilidad = \frac{VP}{VP + FN}$$

    Donde:

    VP: Verdaderos positivos

    FN: Falsos negativos

  - **Especificidad**

    Se define como la probabilidad de clasificar correctamente a un individuo sano, es decir, es la razón entre los individuos sanos con un resultado negativo en la prueba y la cantidad total de individuos sanos.

$$Especificidad = \frac{VN}{VN + FP}$$

Donde:

VN: Verdaderos negativos

FP: Falsos positivos

Ávalos en [39] indica que una buena prueba debe tener alta especificidad y sensibilidad. Así mismo, menciona el siguiente indicador:

- **Área de la curva ROC**

  Las curvas ROC (Receiver Operating Characteristics) son gráficas de los diferentes puntos de corte de una prueba donde los indicadores de sensibilidad y especificidad están ubicados en el eje de las ordenadas y abscisas respectivamente. Además, el área debajo de esta curva nos indica el valor discriminante de la prueba, en otras palabras, mientras mayor sea el área mejor será la prueba diagnóstica.

- **Indicadores que determinan la seguridad:**
  - **Valor predictivo positivo**

    Es la probabilidad de padecer la enfermedad si se obtiene un resultado positivo en el test [40].

$$VPP = \frac{VP}{VP + FP}$$

    Donde:

    VP: Verdaderos positivos

    FP: Falsos positivos

    VPP: Valor predictivo positivo

  - **Valor predictivo negativo**

    Es la probabilidad de que un sujeto con un resultado negativo en la prueba esté realmente sano [40].

$$VPN = \frac{VN}{VN + FN}$$

    Donde:

    VN: Verdaderos negativos

    FN: Falsos negativos

    VPN: Valor predictivo negativo

- **Indicador influyente:**
  - **Prevalencia**

La prevalencia puede definirse como la frecuencia estadística aplicada a grupos de seres humanos o eventos relacionados con la salud [41]. Además, la prevalencia puede influir en el resto de los indicadores tal y como se presenta en la tabla 1.

TABLA *I.*

INFLUENCIA DE LA PREVALENCIA

Indicadores	Prevalencia baja	Prevalencia alta
Sensibilidad	No afecta	No afecta
Especificidad	No afecta	No afecta
VPP	Bajo	Alto
VPN	Alto	Bajo

Se puede concluir de la tabla I que la sensibilidad y la especificidad sirven para verificar la validez de la prueba sin ser influenciados por la prevalencia, sin embargo, los valores predictivos que son usados para tomar decisiones clínicas son altamente influenciados.

Finalmente, en la presente investigación se busca predecir el riesgo de la mortalidad de un paciente infectado con SARS-CoV-2, por lo tanto, haremos uso de los indicadores anteriormente presentados para validar la precisión de la RNA modelada.

Materiales y métodos

Tipo de investigación

Según el manual de Frascati [42], el tipo de investigación denominado desarrollo experimental abarca el desarrollo de nuevo software de aplicaciones. Así, la presente investigación cumplió con este tipo de investigación debido a que se desarrolló una aplicación web utilizando una RNA entrenada con datos extraídos de una clínica de la ciudad de Chiclayo.

Según la metodología que se aplicó para el modelamiento de la RNA, se debe tener dos conjuntos de datos: uno de entrenamiento y otro de prueba. El primero conjunto de datos será utilizado y formó parte del algoritmo interno de la RNA, mientras que el otro grupo sirvió para contrastar el nivel de precisión

de la RNA. Por lo tanto, podemos definir que el diseño de la investigación en la construcción de la RNA fue un experimento puro con posprueba únicamente y grupo de control [43].

Así el diseño de la investigación será:

RG1	X	O1
RG2	-	O2

RG1: Datos obtenidos aleatoriamente de pacientes infectados con COVID-19.

X: Modelamiento y entrenamiento de la RNA.

O1: Validación de la precisión de la RNA en el primer grupo.

RG2: Datos obtenidos aleatoriamente de pacientes infectados con COVID-19.

O2: Validación de la precisión de la RNA en el grupo que no fue utilizado para el modelamiento y entrenamiento de la RNA.

Métodos de investigación

Los métodos de investigación empleados serán los siguientes:

TABLA II

Métodos de investigación

Método	**Descripción**
Inductivo	Planteamiento de la propuesta de solución utilizando RNA
Analítico	Estudio y análisis de antecedentes y datos de historias clínicas
Deductivo	Estudio de la situación problemática global a cerca de la COVID-19
Implementación	La propuesta será ejecutada y validada

Técnicas e instrumentos de recolección de datos

A continuación, en la siguiente tabla se muestra las técnicas e instrumentos que fueron útiles para la recolección de datos.

TABLA III

Técnicas e instrumentos de recolección de datos

Técnicas	**Instrumentos**	**Elementos de la población**	**Propósito**
Análisis documental	Fichas de análisis documental	Datos Abiertos Minsa COVID 19	Definir mejor la situación problemática local
Análisis de distribución	Informe de calidad de datos	Datos de historias clínicas	Identificar los datos sucios
Análisis documental	Fichas de análisis documental	Historias clínicas recogidas	Identificar las características clínicas

Resultados y discusión

Los resultados de esta investigación se mostrarán de acuerdo con la metodología utilizada. Al utilizar Scrum las iteraciones reciben el nombre de Sprint. A continuación, se mostrarán los resultados a partir de cada Sprint.

Sprint 1: Reconocimiento de los datos

Se definió que los datos de los casos COVID-19 deberían abarcar desde el año 2020 [44] que comenzaron los primeros casos del COVID-19 en el Perú hasta el año actual de ejecución de la tesis (2022), además los pacientes deben ser casos que ingresaron al centro de salud para atenderse por sospechas de estar infectado con este virus. Se consideraron los datos de las historias clínicas desde el momento que son atendidos; sin embargo, existió el dato del fallecimiento o alta médica el cual se recolectó en la última fase de atención.

Teniendo en cuenta las revisiones bibliográficas, las escalas de riesgo utilizadas con pacientes COVID y las primeras revisiones de las historias clínicas disponibles se pueden definir las variables en la tabla IV (Ver Anexo 03).

Se necesitó confirmar con los especialistas médicos si estos indicadores eran los correctos para detectar el riesgo de mortalidad de un paciente infectado con COVID-19. Esta historia de usuario no se pudo culminar en un primer momento debido a la poca disponibilidad de los doctores para poder realizar las consultas necesaria.

Sprint 2: Exploración y muestreo

Se consideró tres orígenes de datos, el primero son los datos resumidos de los datos de los pacientes con COVID-19 provenientes del área de epidemiología del hospital II Luis Heysen Inchaústegui, el segundo origen de datos contenía la información de los pacientes fallecidos por esta enfermedad. De estos dos primeros orígenes de datos (archivos xlsx) existen los siguientes datos: edad, sexo, fecha de toma de la muestra, fecha de notificación, estado del

fallecimiento, DNI, fecha de ingreso, servicio, diagnostico, evolución, vacunas, observación, teléfono celular, fecha de nacimiento. Por último, el tercer origen de datos es la base de datos electrónica de los pacientes donde se guarda la información de su historia clínica.

Dado que los orígenes de datos estaban en formato Excel, se unieron utilizando la misma herramienta teniendo en cuenta la información relevante y necesaria para la realización de este proyecto. Se generó un documento donde no se muestran los datos personales (DNI o nombres), los datos considerados son: número, edad, sexo, fecha de notificación, fallecidos. El dato 'número' corresponde a un código para mantener los datos personales del paciente sin revelar. En esta integración de datos se obtuvieron 3471 filas y se analizaron las variables de edad, sexo y fallecidos utilizando la herramienta de Google Colab (Ver Anexo 04) y se obtuvieron los siguientes hallazgos:

- La edad promedio de los pacientes infectados con covid-19 es de 55 años y una mediana de 56 años. Además, en el histograma se puede ver la tendencia hacia la derecha indicando que los pacientes de esta población suelen tener edades elevadas.
- De los datos el sexo tiene dos valores posibles F(Femenino) y M(Masculino) donde la mayor cantidad de pacientes son hombres (2123).
- La variable "fallecido" tiene dos valores posibles 0 (No ha fallecido) y 1 (Falleció) donde la mayor cantidad de pacientes no falleció a causa del COVID-19 (2455).

Debido a las limitaciones de tiempo, se obtuvo una muestra de la población total utilizando la fórmula definida en el Anexo 05. Así, la muestra obtenida es: 345.968864, es decir, 346 pacientes. Además, se realizó un muestreo estratificado o statrified sample, donde se tuvo en consideración la edad y el sexo. Para la consideración de los grupos etarios, se tuvo en cuenta la clasificación mostrada en [45]. Asimismo, para esta estratificación se generó una función para clasificar a los pacientes en los grupos etarios (Ver Fig.8 del Anexo 06), luego utilizando la función de "map" podemos aplicar la función anterior a todo el dataframe (Ver Fig. 9 del Anexo 06). En consecuencia, se generaron los grupos etarios mostrados en la Fig.10 (Ver Anexo 06). Debido a que en el grupo de 100 años solo hubo uno se ampliará el anterior rango de 90

a 99 años utilizando la función de la Fig. 11 (Ver Anexo 06). Así, la distribución de los grupos etarios quedaría definida como se muestra en la Fig. 12 (Ver Anexo 06).

Para el muestreo se consideró el muestreo estratificado considerando los grupos formados por el sexo y la edad. Para ello se debe calcular la proporción que existe entre la cantidad total de la población y la muestra:

$$proporcion = \frac{n_{muestra}}{n_{poblacion}} = 0.0996$$

Así, multiplicando por esta proporción podemos observar lo mostrado en la Fig. 13 (Ver Anexo 06). Sin embargo, debido al redondeo si se calcula la cantidad de datos obtendremos 347 datos, pero necesitamos 346 según la muestra calculada. Así, para que se mantenga el número de datos de la muestra, se retira un elemento del grupo etario con mayor cantidad de datos '60-69' y del sexo 'M': mujer (Ver Fig. 15 del Anexo 06).

Posteriormente, se analizaron los indicadores inicialmente planteados y con la ayuda de los especialistas se confirmó que existían indicadores muy necesarios para medir el riesgo de mortalidad, sin embargo, la aparición en las historias clínicas depende del criterio del médico y si le parece relevante o no pedir algunos análisis. Por tal motivo, cualquier resultado de análisis de laboratorio no será considerado en el presente proyecto sin quitarle la importancia que tendría en un futuro proyecto utilizando estos indicadores. Los indicadores finales serían:

- **Factores demográficos:** Edad, Sexo
- **Comorbilidades:** Diabetes, enfermedades respiratorias previas, enfermedades cardiacas crónicas, Hipertensión, Obesidad/Sobrepeso, Cáncer
- **Síntomas:** fiebre, tos, malestar general, diarrea, mareos, náusea, escalofríos, cansancio, disnea.
- **Signos vitales:** saturación de oxígeno**,** número de dosis de la vacuna

Para la recolección de los datos de la muestra aleatoria se dividió el dataframe de toda la población de acuerdo con los grupos etarios (Ver Fig. 15 del Anexo 06).

Sprint 3: Limpieza y procesamiento de los datos

La recolección de la información de la historia clínica se llevó a cabo gracias al acceso de la base de datos electrónica de EsSalud donde se tomaron la información de la muestra definida anteriormente en la Fig.15. Este proceso fue el que más tiempo tomó debido a la terminología médica que en un comienzo no era comprensible. Además, se apreció la existencia de historias clínicas con información incompleta e información errada. Asimismo, se tuvo el apoyo del personal especialista para la mejor comprensión de los datos.
Algunos hallazgos importantes son:

- Los datos de análisis clínicos no son muy comunes en las historias clínicas revisadas por diversos motivos como: falta de reactivos, falta de personal para la toma o no ser considerados necesarios por el médico.
- Existen pacientes que han tenido más de una visita al hospital debido al COVID-19.
- Dentro de las historias clínicas existen pacientes con pruebas reactivas negativas de COVID-19, sin embargo, por consejo del especialista son consideradas debido a que fueron diagnosticados por contacto epidemiológico.
- Algunos pacientes estuvieron en el hospital por razones distintas a la infección con COVID-19 y con los exámenes de rutina descubrieron que estaban infectados con este virus. Así, estos pacientes no fueron considerados

Sprint 4: Análisis de las variables

Se analizó la muestra de 346 pacientes, para poder encontrar patrones y describir las variables relevantes (Ver Fig. 16 del Anexo 04). De estas variables, el código es simplemente el código asignado a los pacientes para

proteger su identidad. Además, se analizaron cada una de las variables y se realizaron procesos de limpieza como se puede observar en el Anexo 07.

Para los modelos de clasificación es necesario que los datos no sean desbalanceados sobre la variable que se requiere clasificar. En este caso tenemos aproximadamente 71 % de no fallecidos y 29 % de fallecidos. Con lo cual ya se puede sospechar que el modelo que se entrene aprenderá mejor a detectar a los de bajo riesgo de fallecer. Por lo tanto, se debe tener en cuenta esto para el entrenamiento de la red neuronal: se puede mejorar la cantidad de datos haciendo uso del oversampling aleatorio; al momento de separar los conjuntos de entrenamiento y prueba se puede utilizar la estratificación, pero de acuerdo con la variable objetivo(fallecido).

En esta etapa, se realizó el oversampling aleatorio para tener mejores resultados en el entrenamiento del modelo, así se usó la librería SMOTENC (Ver Fig.70 Anexo 08). También se codificaron todas las variables categóricas con valores diferentes a ‘0’ / ‘1’. Para esto se usó la librería Sklearn y la clase LabelEncoder, para nuestro caso, la variable categórica que necesitaba ser codificada era la primera (Ver Fig.71 Anexo 08). Además, las variables deben ser escaladas para evitar que una variable pueda influir más en el modelo por el simple hecho de tener valores elevados. Así, se hizo uso de la librería sklearn y la clase StandardScaler (Ver Fig. 72 Anexo 08)

Por otro lado, para dividir los grupos de entrenamiento y prueba se debe tener en cuenta que mientras más datos existan en el entrenamiento la RNA puede aprender más de los datos. Sin embargo, también se deben tener datos para validar que el valor de la predicción sea mayor al 80%. Así, se decidió que el 66.66% de la muestra sea para entrenar y el 33.33% restante para la validación. En esta investigación se usó la librería Sklearn y la función train_test_split (Ver Fig. 73 Anexo 09)

Para el proceso de aprendizaje, según Kaastra en [46] es recomendable plantear entre 1 o 2 capas ocultas dado que si aumentamos el número de capas aumentará la capacidad computacional que se necesite además de influenciar en un sobreajuste(overfitting) del modelo. Además, tomando como

antecedente la RNA planteada en Inglaterra [24] se puede concluir que 2 capas ocultas es el número de partida adecuado.

Al elegir este número de neuronas se debe tener en cuenta la cantidad de relaciones que se van formando debido a que mientras más relaciones pueden generar un overfitting. Asimismo, mientras más neuronas más capacidad computacional se requiere. Siguiendo lo planteado en [24] se comenzará con 6 neuronas por cada capa oculta. Por último, en la capa de salida solo habrá una neurona la cual es nuestra variable de salida referida al fallecimiento del paciente. Gráficamente la primera arquitectura se define de en la Fig.74 del Anexo 09.

Las funciones de transferencia son necesarias para evitar valores demasiados grandes que puedan alterar negativamente a la RNA [46], de las funciones de transferencia existentes se ha decidido utilizar en las neuronas ocultas la función de rectificador lineal unitario (ReLU) debido a que nuestro modelo pretende predecir entre dos categorías “fallecido” y “no fallecido”. Así, como esta función de transferencia nos arroja valores entre 0 y 1 es una buena función de transferencia como punto de partida. Por otro lado, en la neurona de salida se considerará la función sigmoidal debido a que la salida de esta función son valores continuos entre 0 y 1, así se podrá entender mejor el riesgo donde los valores cercanos a 0 indican bajo riesgo y los valores cercanos a 1 elevado riesgo. Además, de esta forma se puede definir un valor de corte para poder decidir si un paciente tiene elevado o bajo riesgo de mortalidad por el COVID-19. Las funciones de transferencia se pueden observar en la Fig.75 y Fig.76 del Anexo 09.

Con el fin de que nuestro sistema web pueda ir creciendo con el tiempo es necesario tener las funcionalidades divididas correctamente en aplicaciones. Además, las aplicaciones nos permiten ser reutilizadas en otros proyectos web desarrollados en Django [47].Así, nuestro proyecto web tendrá la siguiente estructura: Aplicación de usuarios, Aplicación de registro de RNA, Aplicación de predicción del riesgo, Aplicación de información clínica, Aplicación de gráficas estadísticas, Aplicación de inicio, Aplicación de administración

personalizada. Por otro lado, el sistema web debe ser distribuido para su fácil acceso así se ha planteado la arquitectura de la Fig. 146 (Ver Anexo 17).

Sprint 5: Entrenamiento de la RNA

Durante este Sprint se concluyó que existen métricas muy utilizadas, tales como: AUROC(El área debajo de esta curva ROC nos indica el valor discriminante de la prueba [39], [48]) y F-measure (es el promedio ponderado de la precisión y la sensibilidad[48], [49]).Otra medida para los modelos de clasificación es la exactitud, la cual en conjunto de las anteriores nos indican la validez de nuestra RNA implementada. Por otro lado, también es necesario tener en cuenta que las redes neuronales artificiales se inicializan con pesos aleatorios y en el proceso de entrenamiento se van regularizando estos pesos. Así, luego de entrenar nuestro modelo, bajo la misma arquitectura se pueden obtener diferentes resultados. Con el fin de que nuestra arquitectura de RNA consiga resultados cercanos (varianza baja) se evaluará la arquitectura utilizando la validación cruzada. Como resultado de la validación cruzada, se obtiene la métrica de la desviación estándar, la cual mientras más baja sea nos indica que nuestra arquitectura es más precisa.

En conclusión, los mejores parámetros que se utilizarán para evaluar la validez de la RNA serán: la exactitud, AUROC, F-measure, desviación estándar luego de aplicar la validación cruzada.

En este sprint, se construyó una primera RNA utilizando todos los parámetros inicialmente planteados para poder ejecutar el entrenamiento y posteriormente evaluar la RNA (Fig. 77 Anexo 09). Asimismo, se debió evaluar el modelo comparando las predicciones sobre los datos de entrenamiento y los datos de prueba. La evaluación del modelo se puede observar en el Anexo 10. Ahora se procede a evaluar según la validación cruzada utilizando la clase de KerasClassifier y aplicar la validación cruzada utilizando la función ‘cross_val_score’ de la librería sklearn, así se obtuvo una desviación estándar de: 0.1233 (Ver Fig. 78 Anexo 09). Como primera aproximación podemos ver que nuestra arquitectura de RNA alcanza la exactitud de 79%, un F-Measure de

0.8 y un AUROC de 0.85. Estos valores son menores a los esperados, por lo que se necesitará buscar nuevos parámetros para ir aumentando el valor de las métricas elegidas. Además, se puede observar que existe sobreajuste debido a que el valor de las métricas es mucho mayor en el entrenamiento que en la prueba, así se puede disminuir el sobreajuste utilizando una capa de olvido(dropout) después de cada capa oculta.

En este sprint se comprobaron diferentes hiperparámetros y se fueron evaluando las mejores RNA a partir de las métricas. Se comenzó por mejorar la RNA agregando una capa de dropout (Ver Fig.79 Anexo 09). Luego, se evaluaron diferentes valores de dropout para verificar el modelo iba mejorando, así se obtuvo la siguiente tabla VI del Anexo 09. De lo cual se puede considerar que el mejor dropout es de 0.3 según la desviación estándar, quedando como segundas opciones el 0.2 y 0.4. Ahora para la búsqueda de parámetros se hizo uso de la clase GridSearchCv de la librería Sklearn (Ver Fig. 80 Anexo 09). Así, la búsqueda por parrilla se ejecutó inicialmente para encontrar el número de épocas y el tamaño del batch, de lo cual se obtuvieron los datos descritos en la Tabla VII (Ver Anexo 09). Posteriormente, se siguen iteraciones del entrenamiento buscando la mejor RNA con ayuda del GridSearchCv concluyendo que, aunque se ajustaron los parámetros, no se logró tener una exactitud mayor al 80% ni un área AUROC mayor al 85%. La desviación estándar se mantiene en torno al 10% pero se pudo observar un mayor sobreajuste a los datos de entrenamiento según los gráficos ROC de entrenamiento en comparación a los de prueba. Este sobre ajuste se debe a la cantidad elevada de redes neuronales en las capas ocultas.

Para mejorar el modelo necesitamos cambiar parte de la arquitectura de la RNA. Se supo que desde el comienzo algunos datos de las variables de entrada no eran significativos debido a su baja presencia en la muestra, tales como se muestra en la Fig. 81 (Ver Anexo 09). Luego, se van quitando de 3 en 3 variables para comprobar la mejora en el modelo: 10 neuronas de entrada, 13 neuronas de entrada,16 neuronas de entrada y 19 neuronas de entrada. El resumen de las mejores redes neuronales con diferentes arquitecturas y parámetros se puede visualizar en la Tabla VIII (Ver Anexo 09). Las variables

de entrada (INPUTS): edad, sexo, hipertensión, diabetes, fiebre, tos, malestar general, cansancio, disnea, saturación de oxígeno. Así la mejor RNA se muestra en la Fig.82 del Anexo 09.

Sprint 6: Despliegue de la RNA

Para poder utilizar la RNA es necesario guardar el modelo y el escalador estándar unitario para poder generar las nuevas predicciones. Para guardar el escalador estándar unitario se hará uso de la librería de python joblib utilizando la función dump, así se guardó el escalador 'sc_X' en formato binario (Ver Fig.88 Anexo 11). En el caso de la RNA se guardó en formato .h5 usando el método save de la misma librera Keras (Ver Fig.89 Anexo 11). En este sprint también se diseñaron las principales interfaces las cuales se muestran en el Anexo 12. Debido a las condiciones del software y el tratamiento de la predicción del riesgo de mortalidad, este software tiene como usuarios finales a los médicos. Por tal motivo, existirán solo dos tipos de usuarios: usuario médico y administrador del sistema. De tal manera, el acceso al sistema estará permitido a través de un correo y contraseña, además que la activación de los usuarios será de forma manual luego de validar si tienen autorización a la utilización de este sistema.

Sprint 7: Gestión de acceso al sistema web

Django cuenta con su aplicación de usuarios la cual puede ser reemplazada por una aplicación propia. En este caso, se creó una aplicación de usuarios (accounts) para tener mayor control y personalización de esta clase. La estructura de la aplicación es la mostrada en la Fig.97 (Ver Anexo 13).

Sprint 8: Evaluación de riesgo

Para el ingreso de la arquitectura e hiperparámetros de la RNA se decidió utilizar el administrador de Django debido a que esta funcionalidad solo será utilizada por el administrador del sistema. La carga será utilizando un archivo .h5 que contendrá a la RNA y un archivo de formato .bin que contiene el escalador estándar unitario necesario para ingresar los datos para la predicción.

La función “predict” se encarga de leer la RNA activa a partir del archivo con formato “.h5” y llama a los datos preparados por la función “preparar_data”. Con estos datos se ejecuta la función “predict” propia de Keras. Finalmente, se muestra en formato de texto el riesgo según el umbral definido de “0.6”. Por otro lado, en caso de validar el formulario y obtener una predicción nueva se redirige a la vista de resultado dependiendo del modo simple o detallado. Ambas interfaces se pueden observar en la Fig.110 y Fig.111 del Anexo 13. La clase que maneja esta vista es la definida en la Fig.112 (Ver Anexo 13). También hereda de la clase LoginRequiredMixin, además de la clase DetailView. Esta última es utilizada para las vistas que muestran detalle, en este caso el detalle es de la predicción.

Sprint 9: Nuevos datos para la RNA

La funcionalidad de registros de nuevos datos se encuentra en la aplicación de “riskPredictor”. Se guardarán estos registros utilizando la clase Record del módulo models (Ver Fig.113 Anexo 13). Asimismo, para controlar la transacción se ejecutará el método registrar datos (Ver Fig.114 Anexo 13) donde el bloque de código “transaction. atomic ()” protege la creación de nuevos registros a través de una transacción. Por otro lado, la pantalla del registro de datos es controlada a través de la clase “NuevoRegistroView” (Ver Fig.115 Anexo 12) y “NuevoRegistroDetalladoView” (Ver Fig.116 Anexo 13). En el método “form_valid” se llama a la función “registrar_datos”, la cual realiza la transacción de la creación de nuevos registros. En caso de éxito, se redirige a la vista detallada en “succes_url” la cual tiene los siguientes resultados dependiendo del modo (simple o detallado) (Ver Fig. 117 y 118 Anexo 13). La clase ResultadoRegistroView muestra los datos del registro y los muestra en el template “resultado_registro” (Ver Fig.119 Anexo 13). Por otro lado, se necesitará el acceso a un servidor y poder configurarlo personalizadamente, así, se alquilará una máquina virtual básica como punto de inicio. Dentro de las diferentes opciones de servidores la más económica (con los mismos requerimientos) es DigitalOcean (Ver Fig.131 Anexo 14).

Sprint 10: Gráficas Resumen

Para esta funcionalidad se hará uso de la Chart definida en el módulo ‘chart.py’ que nos servirá como traductor entre python y js gracias al método “get_presentation”. Este método, nos devolverá las etiquetas html y el código js necesario para las gráficas. En JS se utilizará las librerías de jQuery y Chart.JS (Ver Fig.120 Anexo 13).

Sprint 11: Configuración de los servidores

El servidor que se utilizará es NGINX y para configurarlo se ha han colocado las siguientes configuraciones (Ver Fig.123 Anexo 14). Se coloca el dominio adquirido, la ubicación del archivo de gunicorn (librería python que es la interfaz entre la aplicación web y el servidor web) y la ubicación de los archivos estáticos utilizados en la aplicación de Django. En nuestro servidor de base de datos creamos la base de datos que será utilizada por Django. Django se encargará de la creación de la base de datos a través de su ORM (Ver Fig.124 Anexo 14). Asimismo, rn la configuración de la aplicación de Django se instaló la librería que conecta nuestra aplicación con el servidor web: Gunicorn. Por lo tanto, se instaló y se creó un archivo de configuración (Ver Fig.125 Anexo 14). Además, se instaló “supervisor” en nuestro servidor, ya que, es la herramienta que permite gestionar gunicorn y se editó el archivo de configuración de supervisor (Ver Fig.126 Anexo 14).

Sprint 12: Pruebas de software

En el desarrollo de las historias de usuario se obtuvo:

- **HU 50: Pruebas de Caja Negra (Ver Anexo 15)**
- **HU 50: Pruebas de Caja blanca (Ver Anexo 16)**

El desarrollo de esta investigación fue orientado en el cumplimiento de los objetivos específicos que se plantearon inicialmente. El primer objetivo específico consistía en identificar las variables que influyen el riesgo de la mortalidad de la COVID-19 para modelar una RNA precisa, puesto que, en todo análisis de datos es muy importante la correcta definición de las variables que

realmente influyen. Por consiguiente, para poder lograr este objetivo se tuvo el apoyo de especialistas médicos para poder seleccionar las variables que nos ayudarían a crear una RNA más precisa; tal como, Monteza en [32] nos demuestra en su investigación. Inicialmente, se contaron con 23 variables posibles las cuales deberían demostrar ser influyentes en el modelo, por tal motivo, se realizó un análisis de la frecuencia de las variables (características) en los datos de las historias clínicas de los pacientes del centro médico. Durante este análisis, se contaron con 3471 historias clínicas de pacientes que padecieron esta enfermedad; sin embargo, por las limitaciones de tiempo del proyecto y la dificultad para recopilar los datos se extrajo una muestra representativa de 346 pacientes. A pesar de este número de datos, se puede desarrollar modelos predictivos de alta precisión como se muestran en las investigaciones de Luis Garcia [31] y Abdulaal et al [24] que analizaron 360 y 398 registros respectivamente. Después de estudiar la muestra extraída, se descartaron las siguientes variables cuya presencia era nula o muy baja en los datos disponibles: Presencia de trasplantes de corazón, nivel de procalcitonina (PCT), Nivel de Lactato Deshidrogenasa (LDH), Nivel de Ferritina, Recuento absoluto de linfocitos y variable detectada. Asimismo, se incluyeron nuevas variables producto del consejo de los expertos como: Saturación de oxígeno, número de dosis, y la presencia de obesidad. Así, se lograron obtener 19 variables las cuales se encuentran analizadas a partir de su frecuencia en los datos en el Anexo 07. Del mismo modo, en la investigación realizada por Abdulaal A. et al en [25], se observaron 22 variables utilizadas para la creación de su modelo, dichas variables son listadas mostrando su valor de prevalencia con respecto a la muestra estudiada. Se debe tener en cuenta que la prevalencia es lo mismo que la frecuencia, ya que, ambos términos hacen referencia a la cantidad de ocurrencias de un factor en un conjunto de datos, sin embargo, la diferencia es que en los estudios médicos se le denomina prevalencia y en términos estadísticos generales se denomina frecuencia. Por otro lado, debido a la sugerencia de los médicos de hacer una herramienta simple y con menos variables se depuraron las variables con menor frecuencia las cuales se muestran en la Fig. 81. Por consiguiente, se obtuvieron diferentes modelos de RNA teniendo en cuenta como primer indicador la frecuencia de estas variables en los datos.

El segundo objetivo, consistía en identificar la arquitectura e hiperparámetros correctos de la RNA para demostrar la validez del modelo, puesto que, se necesita evitar el sobreajuste de nuestra RNA y encontrar la mejor configuración de hiperparámetros. En consecuencia, se implementó una arquitectura inicial donde las características del paciente corresponden a las neuronas de entrada y la neurona de salida corresponde al alta o deceso del paciente, además para definir el número de capas ocultas se tuvo en cuenta la sugerencia de Kaastra en [46] y la RNA implementada por Abdulaal A. et al en [25] decidiéndose implementar dos capas ocultas. Por otro parte, se decidió utilizar la función rectificador lineal unitario en las neuronas ocultas debido a que este modelo pretende predecir entre un paciente con “riesgo alto” y “riesgo bajo”, no obstante, se definió la función sigmoidal como función de transferencia de la neurona de salida debido a que se desea obtener valores continuos entre 0 y 1 los cuales pueden evidenciar un porcentaje (mientras más alto significa mayor riesgo de mortalidad). Después de haberse definido inicialmente la arquitectura es necesario entrenar el modelo, por tal motivo se definió que el entrenamiento sería guiado por la entropía cruzada, ya que, como se nos indica en la documentación de Keras [50] es utilizada para modelos binarios de clasificación (riesgo alto o riesgo bajo). Posteriormente, se realizó el entrenamiento del modelo y se aplicó el método de validación cruzada con diferentes configuraciones de hiperparámetros. El primer hiperparámetro verificado fue el “dropout” como se muestra en la Tabla IV; en esta tabla se compara las distintas desviaciones estándar obtenidas donde se obtuvo que el mejor hiperparámetro de dropout es de 0.3 (quedando como segundas opciones los valores de 0.2 y 0.4). Asimismo, se aceleró la prueba de diferentes hiperparámetros utilizando la clase de GridSearchCv de la librería SkLearn como se puede observar en la Tabla V con lo cual se obtuvieron métricas como: la desviación estándar, exactitud, F-Measure y AUROC. Posteriormente, se evaluaron diferentes hiperparámetros teniendo en cuenta las métricas anteriormente mencionadas como se puede observar en la Tabla V; los hiperparámetros analizados fueron: tasa de olvido (drop rate), batch, número de épocas, número de neuronas ocultas y el optimizador. Finalmente, se obtuvieron dos arquitecturas de RNA elegidas una que será utilizada porque tiene más detalle de características y la otra que tiene menor cantidad de variables pero que cubre con la necesidad de ser una herramienta práctica para el médico. El modo

simple posee una RNA que tiene una exactitud del 82.71%, AUROC de 88.44%, desviación estándar de 0.0848 y un F-Measure de 83.72% mientras que el modo detallado tiene una exactitud del 81.48%, AUROC de 86.43%, desviación estándar de 0.12 y un F-Measure de 82.14%. Cabe recalcar que ambas RNAs fueron optimizadas con el algoritmo de RMSProp a diferencia de la RNA implementada por Michaeł Wieczorek en [23] que utilizó el algoritmo NAdam para su optimización; esta decisión se tomó debido a que se obtuvieron mejores resultados con el primer algoritmo mencionado. Por otro lado, en ningún otro antecedente se pudo encontrar el detalle de las diferentes combinaciones de arquitecturas e hiperparámetros.

El tercer objetivo específico de este trabajo de investigación fue implementar un modelo de RNA eficiente para demostrar la validez de la solución al diagnosticar el riesgo de mortalidad de un paciente infectado con COVID-19, de ahí que, una vez entrenadas nuestras redes neuronales artificiales fueron almacenadas y replicadas para poder ser distribuibles en la aplicación web. Posteriormente, se calcularon las métricas que según [40] determina la validez de una prueba diagnóstica médica tales como: la sensibilidad, especificidad y el área de la curva ROC (AUROC). En la investigación de Abdulaal et al [24] se obtuvieron valores de 87.50% en sensibilidad, 85.94% en especificidad y un AUROC de 90.12%; mientras que nuestra red neuronal de modo simple tiene 85.71% de sensibilidad. 79.48% de especificidad y un AUROC del 88.48%; del mismo modo la red neuronal del modo detallado tiene una sensibilidad del 82.14%, 80.77% en especificidad y un AUROC del 86.43%. Aunque los valores obtenidos por nuestras redes neuronales son menores que los de la investigación de Abdulaal et al [24], el contexto es diferente lo cual enriquece el trabajo realizado con datos contextualizados a nuestra realidad, además nuestras neuronas pueden mejorar sus resultados debido a la capacidad de aprender de nuevos resultados registrados a través de la aplicación web.

Conclusiones

- Se concluye que para modelar una RNA precisa para el diagnóstico del riesgo de mortalidad del COVID-19 es necesario un estudio bibliográfico junto con

un análisis de las historias clínicas y el apoyo de los expertos médicos, y así obtener las variables relevantes.

- Se concluye que para identificar la correcta arquitectura e hiperparámetros se deben generar diferentes iteraciones de entrenamiento de la RNA; asimismo, en cada iteración se debe utilizar diferentes combinaciones de arquitectura e hiperparámetros calculando las métricas de exactitud, F1Score, AUROC y desviación estándar. Finalmente, se podrá elegir a la combinación con las mejores métricas.

- Se concluye que para demostrar la validez de la solución al diagnosticar el riesgo de mortalidad de un paciente infectado con COVID-19 se deben utilizar métricas de las pruebas diagnósticas que lo demuestren tales como la sensibilidad, especificidad y el AUROC.

Recomendaciones

- Debido a las limitaciones de tiempo del desarrollo de este proyecto no se pudieron analizar a la población general de 3471 pacientes. Por lo tanto, se recomienda analizar una mayor cantidad de datos y comprobar si existe una mejora en las métricas calculadas.
- Al diseñar los modelos de RNA se pasaron por varias iteraciones debido a que este proceso es puramente experimental de prueba y error hasta lograr obtener una RNA con indicadores elevados. Por consiguiente, es recomendable, probar con diferentes arquitecturas e hiperparámetros utilizando herramientas que aceleran este proceso como GridSearch.
- Debido a que este proyecto estaba limitado a la toma de datos del primer acercamiento al centro médico, es decir, son los datos registrados en su ingreso al área de Urgencias/Emergencias no se pudieron incluir los resultados de análisis de laboratorios que podrían ser relevantes para la predicción del riesgo. En consecuencia, se podría realizar otro proyecto incluyendo estas variables y la actualización del riesgo conforme evoluciona la enfermedad.

Referencias

[1] CSSE, «COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (JHU)», *Johns Hopkins University & Medicine*, 18 de septiembre de 2021. https://coronavirus.jhu.edu/map.html (accedido 18 de septiembre de 2021).

[2] H. Lambert *et al.*, «COVID-19 as a global challenge: towards an inclusive and sustainable future», *Lancet Planet. Health*, vol. 4, n.º 8, pp. e312-e314, ago. 2020, doi: 10.1016/S2542-5196(20)30168-6.

[3] Banco Mundial, «La COVID-19 (coronavirus) hunde a la economía mundial en la peor recesión desde la Segunda Guerra Mundial», *Banco Mundial*, 8 de junio de 2020. https://www.bancomundial.org/es/news/press-release/2020/06/08/covid-19-to-plunge-global-economy-into-worst-recession-since-world-war-ii (accedido 7 de junio de 2021).

[4] C. O'Hagan, «En el contexto de la COVID-19, la UNESCO moviliza a 122 países para promover la ciencia abierta y una mayor cooperación», *UNESCO*, 30 de marzo de 2020. https://es.unesco.org/news/contexto-covid-19-unesco-moviliza-122-paises-promover-ciencia-abierta-y-mayor-cooperacion (accedido 18 de septiembre de 2021).

[5] K. Dorjee, H. Kim, E. Bonomo, y R. Dolma, «Prevalence and predictors of death and severe disease in patients hospitalized due to COVID-19: A comprehensive systematic review and meta-analysis of 77 studies and 38,000 patients», *PLOS ONE*, vol. 15, n.º 12, p. e0243191, dic. 2020, doi: 10.1371/journal.pone.0243191.

[6] R. Alizadehsani *et al.*, «Risk factors prediction, clinical outcomes, and mortality in COVID-19 patients», *J. Med. Virol.*, vol. 93, n.º 4, pp. 2307-2320, abr. 2021, doi: 10.1002/jmv.26699.

[7] J. M. Velasco, W.-C. Tseng, y C.-L. Chang, «Factors Affecting the Cases and Deaths of COVID-19 Victims», *Int. J. Environ. Res. Public. Health*, vol. 18, n.º 2, p. 674, ene. 2021, doi: 10.3390/ijerph18020674.

[8] E. Keleş Peker, G. Bektemur, K. N. Baydili, y M. Aktaş, «Comparison of COVID-19 Case-Fatality-Rates by Socio- Demographic Factors», *J. Acad. Res. Med.*, vol. 10, n.º 3, pp. 246-251, dic. 2020, doi: 10.4274/jarem.galenos.2020.3790.

[9] L. Yang, J. Jin, W. Luo, Y. Gan, B. Chen, y W. Li, «Risk factors for predicting mortality of COVID-19 patients: A systematic review and meta-analysis», *PLOS ONE*, vol. 15, n.º 11, p. e0243124, nov. 2020, doi: 10.1371/journal.pone.0243124.

[10] Z. Zheng *et al.*, «Risk factors of critical & mortal COVID-19 cases: A systematic literature review and meta-analysis», *J. Infect.*, vol. 81, n.º 2, pp. e16-e25, ago. 2020, doi: 10.1016/j.jinf.2020.04.021.

[11] R. Kuhn, «Coronavirus variants, viral mutation and COVID-19 vaccines: The science you need to understand», *The Conversation U.S*, 2021. Accedido: 3 de abril de 2021. [En línea]. Disponible en: https://www.proquest.com/newspapers/coronavirus-variants-viral-mutation-covid-19/docview/2486058936/se-2?accountid=37610.

[12] MINSA, «Sala Situacional COVID-19 Perú», *Covid-19 en el Perú*, 18 de septiembre de 2021. https://covid19.minsa.gob.pe/sala_situacional.asp (accedido 18 de septiembre de 2021).

[13] Essalud, «EsSalud Lambayeque amplía camas UCI y fortalece su capacidad de respuesta frente a la Covid-19», Perú, 24 de marzo de 2021. Accedido: 4 de junio de 2021. [En línea]. Disponible en: http://noticias.essalud.gob.pe/?inno-noticia=essalud-lambayeque-amplia-camas-uci-y-fortalece-su-capacidad-de-respuesta-frente-a-la-covid-19

[14] Defensoría del Pueblo, «Defensoría del Pueblo: no hay camas UCI disponibles para pacientes COVID-19 en hospitales de Lambayeque», *Plataforma digital única del Estado Peruano*, 13 de enero de 2021. https://www.gob.pe/institucion/defensoria-del-pueblo/noticias/324783-defensoria-del-pueblo-no-hay-camas-uci-disponibles-para-pacientes-covid-19-en-hospitales-de-lambayeque (accedido 3 de abril de 2021).

[15] Canal N, «COVID-19: Conoce la disponibilidad de camas UCI diaria a nivel nacional», 25 de enero de 2021. Accedido: 7 de junio de 2021. [En línea]. Disponible en: https://canaln.pe/actualidad/covid-19-conoce-disponibilidad-diaria-camas-uci-nivel-nacional-n430186

[16] I. Ahmad y S. Muhammad Asad, «Predictions of coronavirus COVID-19 distinct cases in Pakistan through an artificial neural network», *Epidemiol. Infect.*, vol. 148, p. e222, 2020, doi: 10.1017/S0950268820002174.

[17] F. Beskyd, «Prediction of Dota 2 Game Result», Bachelor's thesis, Czech Technical University, Praga, 2018. Accedido: 3 de abril de 2021. [En línea]. Disponible en: https://core.ac.uk/download/pdf/159310682.pdf

[18] S.-H. Hsiang y J.-L. Kuo, «Applying ANN to predict the forming load and mechanical property of magnesium alloy under hot extrusion», *Int. J. Adv. Manuf. Technol.*, vol. 26, n.º 9-10, pp. 970-977, oct. 2005, doi: 10.1007/s00170-004-2064-0.

[19] A. Barcellona, D. Palmeri, y R. Riccobono, «ANN Model to predict the bake hardenability of Transformation-Induced Plasticity steels», presentado en MATERIALS CHARACTERISATION 2009, New Forest, UK, jun. 2009, pp. 33-44. doi: 10.2495/MC090041.

[20] B. W. Wanjawa y L. Muchemi, «NN Model to Predict Stock Prices at Stock Exchange Markets». Cornell University Library, arXiv.org, 14 de diciembre de 2014. Accedido: 3 de abril de 2021. [En línea]. Disponible en: https://www.proquest.com/working-papers/ann-model-predict-stock-prices-at-exchange/docview/2081135442/se-2?accountid=37610.

[21] J.-S. Chiu, Y.-F. Wang, Y.-C. Su, L.-H. Wei, J.-G. Liao, y Y.-C. Li, «Artificial Neural Network to Predict Skeletal Metastasis in Patients with Prostate Cancer», *J. Med. Syst.*, vol. 33, n.º 2, pp. 91-100, abr. 2009, doi: 10.1007/s10916-008-9168-2.

[22] M. Maia, J. S. Pimentel, I. S. Pereira, J. Gondim, M. E. Barreto, y A. Ara, «Convolutional Support Vector Models: Prediction of Coronavirus Disease Using Chest X-rays», *Information*, vol. 11, n.º 12, p. 548, nov. 2020, doi: 10.3390/info11120548.

[23] M. Wieczorek, J. Siłka, y M. Woźniak, «Neural network powered COVID-19 spread forecasting model», *Chaos Solitons Fractals*, vol. 140, p. 110203, nov. 2020, doi: 10.1016/j.chaos.2020.110203.

[24] A. Abdulaal, A. Patel, E. Charani, S. Denny, N. Mughal, y L. Moore, «Prognostic Modeling of COVID-19 Using Artificial Intelligence in the United Kingdom: Model Development and Validation», *J. Med. Internet Res.*, vol. 22, n.º 8, p. e20259, ago. 2020, doi: 10.2196/20259.

[25] L. A. Guerra Grados, «Reconocimiento del síndrome metabólico mediante el diseño de un sistema experto basado en redes neuronales en una población del Hospital Hipólito Unanue», Tesis de licenciatura, Universidad Nacional Mayor de San Marcos, LIMA, 2015. Accedido: 3 de abril de 2021. [En línea]. Disponible en: https://hdl.handle.net/20.500.12672/5787

[26] T. Steed, A. Alawieh, F. Akbik, O. Sadan, O. Samuels, y J. Grossberg, «P-036 Artificial neural network modeling of clinical outcomes in subarachnoid hemorrhage», en *SNIS 19th annual meeting oral poster abstracts*, jul. 2022, p. A71.2-A72. doi: 10.1136/neurintsurg-2022-SNIS.108.

[27] D. Aguilar Vilca y J. C. Camargo Ramos, «Sistema inteligente basado en redes neuronales, máquina de soporte vectorial y random forest para la predicción de deserción de clientes en microcréditos de bancos», Tesis de pregrado, Universidad Nacional Mayor de San Marcos, LIMA, 2021. Accedido: 8 de noviembre de 2022. [En línea]. Disponible en: https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/16390/Aguilar_vd.pdf?sequence=1&isAllowed=y

[28] C. J. CARRIÓN OSNAYO, «APLICACIÓN DE REDES NEURONALES ARTIFICIALES PARA LA PREDICCIÓN DE LA RECUPERACIÓN DE PLANTA CONCENTRADORA EN MINSUR S.A. – UNIDAD SAN RAFAEL», Tesis de pregrado, UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA, Arequipa, Perú, 2018. Accedido: 8 de noviembre de 202d. C. [En línea]. Disponible en: http://repositorio.unsa.edu.pe/bitstream/handle/UNSA/6629/IMcaoscj.pdf?sequence=1&isAllowed=y

[29] D. L. Nazario Huanaco, «REDES NEURONALES ARTIFICIALES Y LA PREDICCIÓN DE LA VIABILIDAD LEGISLATIVA EN LAS PROPOSICIONES PARLAMENTARIAS, PERÚ, AÑO 2020», Tesis de pregrado, Universidad Alas Peruanas, Lima,Perú, 2021. Accedido: 8 de noviembre de 2022. [En línea]. Disponible en: https://repositorio.uap.edu.pe/xmlui/bitstream/handle/20.500.12990/10143/Tesis_redes%20neuronales%20artificiales_predicci%c3%b3n_viabilidad%20legislativa_proposiciones%20parlamentarias_Per%c3%ba.pdf?sequence=1&isAllowed=y

[30] A. M. Garcia Nazario, «Sistema de información basado en redes neuronales para la predicción de riesgo en el otorgamiento de créditos personales en una cooperativa de ahorro y crédito en el departamento de Lambayeque», Ingeniero, Universidad Católica Santo Toribio de Mogrovejo, Perú, 2020. Accedido: 3 de abril de 2021. [En línea]. Disponible en: http://hdl.handle.net/20.500.12423/2975

[31] L. J. Garcia Peredo, «SISTEMA PREDICTIVO DE RENDIMIENTO ACADÉMICO EN BASE A FACTORES INFLUYENTES EN ESTUDIANTES DEL 1º SECUNDARIA EN UN COLEGIO DE LAMBAYEQUE», Tesis de pregrado, Universidad Católica Santo Toribio de Mogrovejo, Chiclayo, Perú, 2021. Accedido: 9 de noviembre de 2022. [En línea]. Disponible en: https://tesis.usat.edu.pe/bitstream/20.500.12423/4723/1/TL_GarciaPeredoLuis.pdf

[32] I. E. Monteza Vallejos, «SISTEMA DE DIAGNÓSTICO DE PERFILES PSICOTÉCNICOS BASADO EN REGRESIÓN MÚLTIPLE PARA MEJORAR LA SELECCIÓN DE CATEQUISTAS DE LA PARROQUIA SANTA ROSA DE LIMA DE CHICLAYO – PERÚ», Tesis de pregrado, Universidad Católica Santo Toribio de Mogrovejo, Chiclayo, Perú, 2021. Accedido: 9 de noviembre de 2022. [En línea]. Disponible en: https://tesis.usat.edu.pe/bitstream/20.500.12423/3355/1/TL_MontezaVallejosIrisElizabeth.pdf

[33] L. Amador Hidalgo, Universidad de Córdoba, y Servicio de Publicaciones, *Inteligencia artificial y sistemas expertos*. Córdoba: Servicio de Publicaciones de la Universidad de Córdoba, 1997.

[34] J. M. Angulo Usategui y A. del Moral Bueno, *Guía fácil de la inteligencia artificial*. Madrid: Paraninfo, 1986.
[35] H. Banda, *Inteligencia Artificial: Principios y Aplicaciones*, 1.ª ed. Escuela Politecnica Nacional Authors, 2014. Accedido: 5 de abril de 2021. [En línea]. Disponible en: https://www.researchgate.net/publication/262487459_Inteligencia_Artificial_Principios_y_Aplicaciones
[36] M. Escobar, *Sistema nervioso*. Cali, Colombia: Programa Editorial Universidad del Valle, 2006.
[37] A. Nacelle y E. Mizraji, «Redes neuronales artificiales», *Las Redes Neuronales Biol. En Algoritm. Clasif. Urug.*, 2009, Accedido: 24 de mayo de 2021. [En línea]. Disponible en: http://www.nib.fmed.edu.uy/Seminario%202009/Monografias%20seminario%202009/Nacell-Redes%20NeuronalesImplementacion.pdf
[38] R. G. M. Morris, «D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949», *Brain Res. Bull.*, vol. 50, n.º 5-6, p. 437, nov. 1999, doi: 10.1016/S0361-9230(99)00182-3.
[39] O. Ávalos, «Las pruebas diagnósticas. Su aplicación en los estudios epidemiológicos.», *Nefrología*, vol. 20, pp. 403-407, 2000.
[40] S. Pita Fernández y S. Pértegas Díaz, «Pruebas diagnósticas: Sensibilidad y especificidad», *Cad Aten Primaria*, vol. 10, pp. 120-124, 2003.
[41] J. Granados, «Medidas de prevalencia y relación incidencia-prevalencia», *Med Clin Barc*, vol. 105, pp. 216-218, 1995.
[42] «Manual de Frascati 2015: Guía para la recopilación y presentación de información sobre la investigación y el desarrollo experimental». https://www.oecd-ilibrary.org/science-and-technology/manual-de-frascati-2015_9789264310681-es;jsessionid=k3EMUDrqOQbqcmrXSyWouy-g.ip-10-240-5-114 (accedido 22 de diciembre de 2021).
[43] R. Hernández Sampieri, C. Fernández Collado, y P. Baptista Lucio, «Metodología de la investigación». McGraw-Hill, México, D.F., 2006.
[44] «Respuesta a la emergencia por COVID-19 en Perú - OPS/OMS | Organización Panamericana de la Salud». https://www.paho.org/es/respuesta-emergencia-por-covid-19-peru (accedido 31 de marzo de 2022).
[45] J. E. Díaz Pinzón, «Descripción estadística del COVID- 19 según el grupo etario en Colombia», *Rev. Repert. Med. Cir.*, pp. 79-85, jul. 2020, doi: 10.31260/RepertMedCir.01217372.1098.
[46] I. Kaastra y M. Boyd, «Designing a neural network for forecasting financial and economic time series», *Neurocomputing*, vol. 10, n.º 3, pp. 215-236, abr. 1996, doi: 10.1016/0925-2312(95)00039-9.
[47] «Advanced tutorial: How to write reusable apps | Django documentation | Django». https://docs.djangoproject.com/en/4.0/intro/reusable-apps/ (accedido 13 de mayo de 2022).
[48] D. M. W. Powers, «Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation», 2020, doi: 10.48550/ARXIV.2010.16061.
[49] G. Hripcsak, «Agreement, the F-Measure, and Reliability in Information Retrieval», *J. Am. Med. Inform. Assoc.*, vol. 12, n.º 3, pp. 296-298, ene. 2005, doi: 10.1197/jamia.M1733.
[50] K. Team, «Keras documentation: Probabilistic losses». https://keras.io/api/losses/probabilistic_losses/#binary_crossentropy-function (accedido 13 de mayo de 2022).

[51] J. Zhang *et al.*, «Risk factors for disease severity, unimprovement, and mortality in COVID-19 patients in Wuhan, China», *Clin. Microbiol. Infect.*, vol. 26, n.º 6, pp. 767-772, jun. 2020, doi: 10.1016/j.cmi.2020.04.012.
[52] «Figura 18. Función de activación ReLU.», *ResearchGate*. https://www.researchgate.net/figure/Figura-18-Funcion-de-activacion-ReLU_fig5_328600321 (accedido 13 de mayo de 2022).
[53] «Sigmoidal Nonlinearity», *DeepAI*, 17 de mayo de 2019. https://deepai.org/machine-learning-glossary-and-terms/sigmoidal-nonlinearity (accedido 13 de mayo de 2022).
[54] «Droplets | DigitalOcean's Scalable Virtual Machines». https://www.digitalocean.com/products/droplets (accedido 17 de mayo de 2022).

Anexos

ANEXO N° 01. CONSTANCIA DE APROBACIÒN DEL PRODUCTO ACREDITABLE DE LA ENTIDAD DONDE SE EJECUTÓ LA TESIS

CONSTANCIA DE APROBACIÓN DEL PRODUCTO ACREDITABLE

Yo, Ricardo Ponce Linares identificado con el número de DNI 17528941 indico la aprobación de la implementación del **"SISTEMA INTELIGENTE WEB BASADO EN REDES NEURONALES ARTIFICIALES PARA LA PREDICCIÓN DEL RIESGO DE MORTALIDAD DEL COVID-19"** desarrollada por el estudiante **SERGIO ALEXANDER MONDRAGON SILVA,** identificado con **N.°. DNI:72763743** de la carrera profesional de Ingeniería de Sistemas y Computación de la **"UNIVERSIDAD CATÓLICA SANTO TORIBIO DE MOGROVEJO"**, como producto acreditable de su trabajo de investigación de fin de grado.

Requerimiento del modelo de RNA			
Pregunta	**¿Conforme?**		**Observaciones**
	SI	**NO**	
¿Las métricas utilizadas para evaluar la herramienta médica son las correctas?	X		
¿El nivel de exactitud del modelo de RNA es aceptable?	X		
¿El nivel de sensibilidad del modelo de RNA es aceptable?	X		
¿El nivel de especificidad del modelo de RNA es aceptable?	X		
¿Cree usted que los modelos de RNA implementados funcionan como una herramienta para el apoyo de la predicción del riesgo de mortalidad en pacientes infectados con COVID-19?	X		

EsSalud HOSPITAL II-1 LUIS HEYSEN INCHAUSTEGUI

Dr. Ricardo A. Ponce Linares
Médico Otorrinolaringologia
Certificador de Discapacidad
CMP. N° 27544 - RNE: N° 11688

Requerimientos no funcionales del sistema				
Requerimiento	**Descripción**	**¿Conforme?**		**Observaciones**
		SI	**NO**	
USABILIDAD	El sistema debe ser fácil de usar	X		
	Se cuenta con un manual de usuarios	X		
SEGURIDAD	El ingreso al sistema está restringido utilizando usuarios y contraseñas.	X		
MULTIPLATAFORMA	El sistema funciona en distintos tipos de dispositivos sin presentar ningún contratiempo	X		
DESEMPEÑO	El sistema no presenta problemas para su manejo e implementación	X		

Requerimientos funcionales del sistema				
Requerimiento	**Descripción**	**¿Conforme?**		**Observaciones**
		SI	**NO**	
Evaluar el riesgo de mortalidad	El sistema permite el ingreso de datos del paciente a evaluar y muestra el riesgo de mortalidad.	X		
Registrar nuevos datos de pacientes infectados	El sistema permite el registro de los datos de un paciente infectado con COVID-19.	X		
Visualización de los datos resumidos	El sistema muestra gráficamente los datos analizados para el entrenamiento de la RNA.	X		

EsSalud
HOSPITAL II-1 LUIS HEYSEN INCHAUSTEGUI
Dr. Ricardo A. Ponce Linares
Médico Otorrinolaringología
Certificador de Discapacidad
CMP. N° 27564 - RNE: N° 11658

ANEXO N° 02. GUÍA DE ENTREVISTA

Entrevista 1

Dirigido: Ricardo Ponce Linares

1. ¿Existen fuentes de datos de los pacientes infectados y fallecidos?
2. ¿Cuáles son las principales características que se evaluaría en un paciente con COVID-19?
3. ¿Le sería útil una herramienta de apoyo a la predicción de mortalidad?
4. ¿Cuáles son las principales características que desearía en una herramienta de apoyo?

Entrevista 2

Dirigido: jefe de área de emergencias

1. ¿Cuáles son las principales características que se evaluaría en un paciente con COVID-19?
2. Según su punto de vista, ¿Cuál sería la variable más relevante?
3. ¿Le sería útil una herramienta de apoyo a la predicción de mortalidad?
4. ¿Cuáles son las principales características que desearía en una herramienta de apoyo?

ANEXO N° 03. GUÍA DE ENTREVISTA

TABLA IV

VARIABLES ANALIZADAS EN LA REVISIÓN BIBLIOGRÁFICA

Variable	Referencia bibliográfica
Edad	[6], [10]
Sexo	[5], [10], [51]
Diabetes	[5], [10]
Hipertensión	[5], [10]
Enfermedades respiratorias previas	[10]
Enfermedades cardiacas crónicas	[10]
Historial de tabaquismo	[5], [10]
Presencia de fiebre	[6], [10]
Presencia de tos	[24]
Presencia de fatiga	[6], [9]
Presencia de escalofríos	[6]
Presencia de disnea	[6], [9], [10]
Recuento absoluto de leucocitos	[9]
Nivel de procalcitonina	[9], [10]
Nivel de Dímero D	[10]
Nivel de Lactato deshidrogenasa	[9], [10]
Nivel de ferritina	[9]
Recuento absoluto de linfocitos	[9]

ANEXO N° 04. ANÁLISIS VARIABLES FUENTE INICIAL

```
Edad: Variable cuantitativa
count    3471.000000
mean       55.154422
std        17.778490
min         1.000000
25%        42.000000
50%        56.000000
75%        68.000000
max       100.000000
Name: edad, dtype: float64
```

Fig. 1. Análisis de la variable edad de la primera fuente de datos

Fuente: Elaboración propia

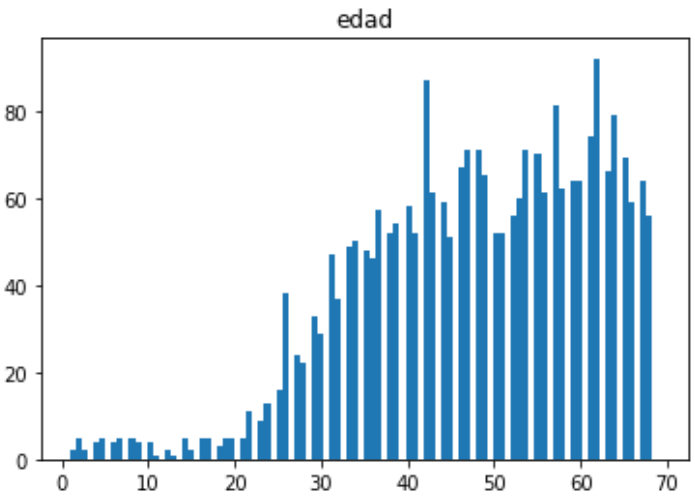


Fig. 2. Histograma de la variable edad

Fuente: Elaboración propia

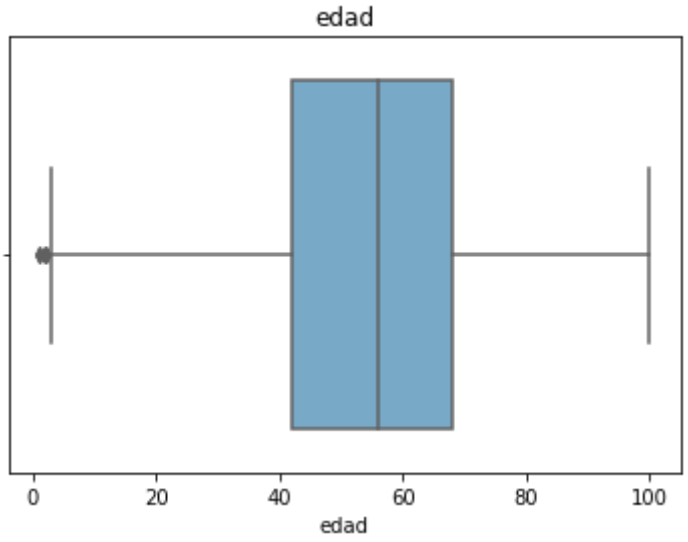


Fig. 3. Diagrama de caja de la variable edad

Fuente: Elaboración propia

```
Sexo: Variable cualitativa
count     3471
unique       2
top          M
freq      2123
Name: sexo, dtype: object
```

Fig. 4. Análisis de la variable sexo de la primera fuente de datos

Fuente: Elaboración propia

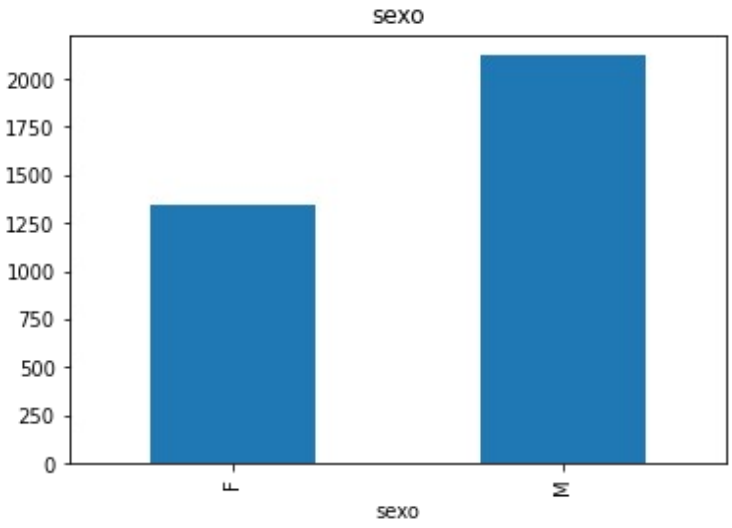


Fig. 5. Gráfico de barras de la variable sexo

Fuente: Elaboración propia

```
Fallecido: Variable cualitativa
count     3471
unique       2
top          0
freq      2455
Name: fallecido, dtype: object
```

Fig. 6. Análisis de la variable fallecido de la primera fuente de datos

Fuente: Elaboración propia

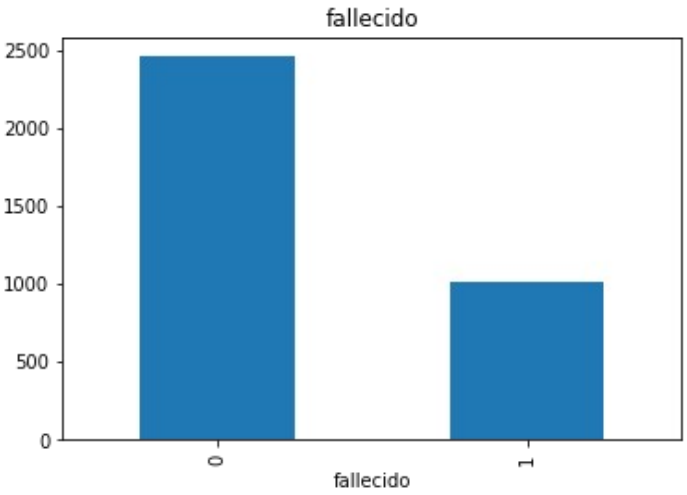


Fig. 7. Gráfico de barras de la variable fallecido

Fuente: Elaboración propia

ANEXO N° 05. FÓRMULA DE CÁLCULO DE LA MUESTRA

$$n = \frac{N*Z^2*p*q}{d^2*(N-1)+Z^2*p*q}$$

p = proporción aproximada del fenómeno en estudio en la población de referencia.

q = proporción de la población de referencia que no presenta el fenómeno en estudio (1-p).

N = tamaño de la población

Z = valor de Z crítico, calculado en las tablas del área de la curva normal. Llamado también nivel de confianza.

d = nivel de precisión absoluta. Referido a la amplitud del intervalo de confianza deseado en la determinación del valor promedio de la variable en estudio.

Se consideraron los siguientes parámetros:

n = 3471

p= 0.5

q = 0.5

z = 1.96 (95 % de confianza)

d = 0.05 (5 % de error)

ANEXO N° 06. DISTRIBUCIÓN DEL GRUPO ETARIO

```python
def grupo_etario(edad):
  if edad<10:
    return cambio[0]
  elif edad < 20:
    return cambio[1]
  elif edad <30:
    return cambio[2]
  elif edad < 40:
    return cambio[3]
  elif edad < 50:
    return cambio[4]
  elif edad < 60:
    return cambio[5]
  elif edad < 70:
    return cambio[6]
  elif edad < 80:
    return cambio[7]
  elif edad < 90:
    return cambio[8]
  elif edad < 100:
    return cambio[9]
  else :
    return cambio[10]
```

Fig. 8. Función de clasificación en grupos etarios

Fuente: Elaboración propia

```python
df_analisis_edad=df_final.assign(
                    grupo_etario=df_final['edad'].apply(lambda x : grupo_etario(x))
)[["edad","grupo_etario"]]
```

Fig. 9 Asignación de los grupos etarios al dataframe

Fuente: Elaboración propia

```
grupo_etario
0-9        36
10-19      33
100-        1
20-29     176
30-39     469
40-49     642
50-59     629
60-69     688
70-79     486
80-89     261
90-99      50
dtype: int64
```

Fig. 10. Grupos etarios del dataframe

Fuente: Elaboración propia

```python
cambio = [
  "0-9","10-19","20-29","30-39","40-49","50-59","60-69","70-79","80-89","90-100"
]
def grupo_etario(edad):
  if edad<10:
    return cambio[0]
  elif edad < 20:
    return cambio[1]
  elif edad <30:
    return cambio[2]
  elif edad < 40:
    return cambio[3]
  elif edad < 50:
    return cambio[4]
  elif edad < 60:
    return cambio[5]
  elif edad < 70:
    return cambio[6]
  elif edad < 80:
    return cambio[7]
  elif edad < 90:
    return cambio[8]
  else :
    return cambio[9]

df_analisis_edad=df_final.assign(
    grupo_etario=df_final['edad'].apply(lambda x : grupo_etario(x))
)
```

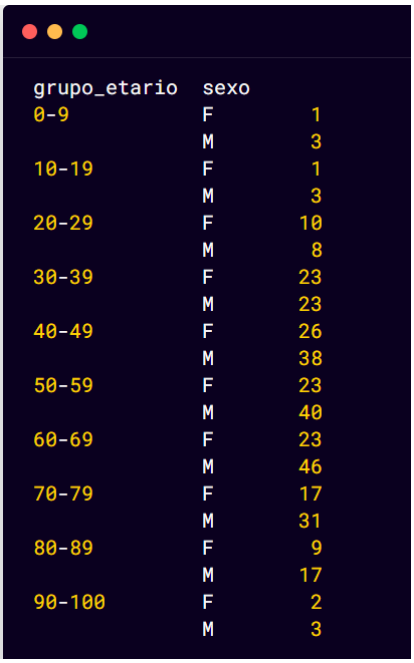
Fig. 11. Reasignación de los grupos etarios

Fuente: Elaboración propia

	grupo_etario	sexo	fallecido	Cantidad
1	0-9	F	0	8
2		M	0	28
3	10-19	F	0	7
4		M	0	26
5	20-29	F	0	93
6			1	3
7		M	0	77
8			1	3
9	30-39	F	0	233
10			1	2
11		M	0	211
12			1	23
13	40-49	F	0	242
14			1	18
15		M	0	322
16			1	60
17	50-59	F	0	180
18			1	52
19		M	0	292
20			1	105
21	60-69	F	0	133
22			1	96
23		M	0	254
24			1	205
25	70-79	F	0	80
26			1	93
27		M	0	138
28			1	175
29	80-89	F	0	52
30			1	38
31		M	0	59
32			1	112
33	90-100	F	0	6
34			1	12
35		M	0	14
36			1	19
37				

Fig. 12.Resumen de la reasignación de los grupos etarios

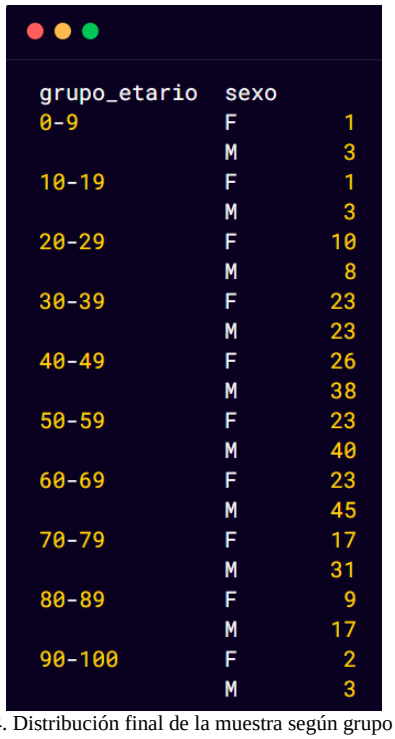
Fuente: Elaboración propia



grupo_etario	sexo	
0-9	F	1
	M	3
10-19	F	1
	M	3
20-29	F	10
	M	8
30-39	F	23
	M	23
40-49	F	26
	M	38
50-59	F	23
	M	40
60-69	F	23
	M	46
70-79	F	17
	M	31
80-89	F	9
	M	17
90-100	F	2
	M	3

Fig. 13. Distribución de la muestra según grupo etario

Fuente: Elaboración Propia



grupo_etario	sexo	
0-9	F	1
	M	3
10-19	F	1
	M	3
20-29	F	10
	M	8
30-39	F	23
	M	23
40-49	F	26
	M	38
50-59	F	23
	M	40
60-69	F	23
	M	45
70-79	F	17
	M	31
80-89	F	9
	M	17
90-100	F	2
	M	3

Fig. 14. Distribución final de la muestra según grupo etario

Fuente: Elaboración Propia

data frames	grupo	numero_muestra
df_0	0-9,F	1
df_1	0-9,M	3
df_2	10-19,F	1
df_3	10-19,M	3
df_4	20-29,F	10
df_5	20-29,M	8
df_6	30-39,F	23
df_7	30-39,M	23
df_8	40-49,F	26
df_9	40-49,M	38
df_10	50-59,F	23
df_11	50-59,M	40
df_12	60-69,F	23
df_13	60-69,M	45
df_14	70-79,F	17
df_15	70-79,M	31
df_16	80-89,F	9
df_17	80-89,M	17
df_18	90-100,F	2
df_19	90-100,M	3

Fig. 15. Tabla resumen dataframes y muestra

Fuente: Elaboración Propia

ANEXO N° 07. ANÁLISIS DE LAS VARIABLES

```
RangeIndex: 346 entries, 0 to 345
Data columns (total 21 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   CODIGO                                346 non-null    float64
 1   EDAD                                  346 non-null    float64
 2   SEXO                                  346 non-null    object
 3   DIABETES                              346 non-null    float64
 4   ENFERMEDADES_RESPIRATORIAS_PREVIAS    346 non-null    float64
 5   ENFERMEDADES_CARDIACAS_CRONICAS       346 non-null    float64
 6   HIPERTENSION                          346 non-null    float64
 7   OBESIDAD_SOBREPESO                    346 non-null    float64
 8   CANCER                                346 non-null    float64
 9   FIEBRE                                346 non-null    float64
 10  TOS                                   346 non-null    float64
 11  MALESTAR_GENERAL                      346 non-null    float64
 12  DIARREA                               346 non-null    float64
 13  MAREOS                                345 non-null    float64
 14  NAUSEAS                               345 non-null    float64
 15  ESCALOFRIOS                           345 non-null    float64
 16  FATIGA                                346 non-null    float64
 17  DISNEA                                346 non-null    float64
 18  SAT_O2_ambiental                      346 non-null    float64
 19  nro_dosis                             346 non-null    float64
 20  FALLECIDO                             346 non-null    float64
```

Fig. 16. Descripción general de las variables

Fuente: Elaboración propia

Variables Numéricas

- **Edad**

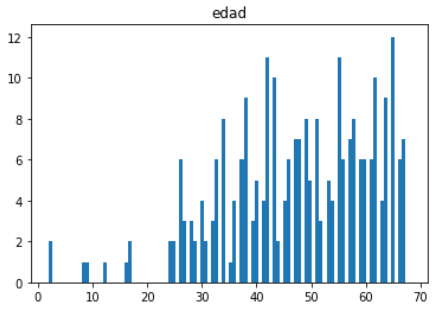


Fig. 17. Histograma de la variable edad

Fuente: Elaboración propia

Según el histograma, es una distribución asimétrica con cola a la izquierda

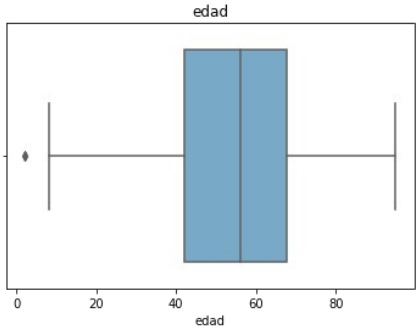


Fig. 18. Diagrama de caja de la variable edad

Fuente: Elaboración propia

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Cuantil	0.0	1.00	5.00	10.0	20.0	30.0	40.0	50.0	60.0	70.0	...	92.5	93.00	94.0	95.00	96.0	97.0	97.5	98.0	99.0	100.0
Valor	2.0	10.35	26.25	32.5	39.0	44.5	50.0	56.0	61.0	65.0	...	80.0	80.85	81.3	82.75	83.0	84.0	84.0	85.0	91.0	95.0

Fig. 19. Análisis por cuantil de la variable edad

Fuente: Elaboración propia

De acuerdo con el gráfico de caja y al análisis de los cuantiles se pueden observar los outliers, por lo tanto, se procede a corregir esto:

```
quantile_1 = np.percentile(df['edad'],1)
quantile_99 = np.percentile(df['edad'],99)
df.loc[df['edad']<quantile_1,'edad'] = quantile_1
df.loc[df['edad']>quantile_99,'edad'] = quantile_99
```

Fig. 20. Corrección del cuantil de la variable edad

Fuente: Elaboración propia

Después del tratamiento de los outliers:

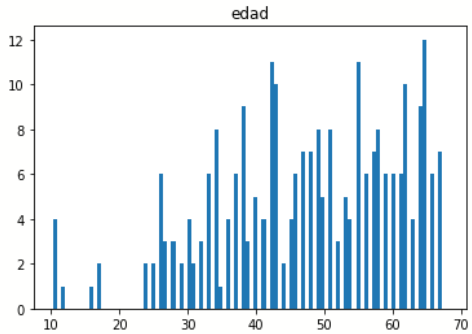


Fig. 21. Histograma de la variable edad después del tratamiento de outliers

Fuente: Elaboración propia

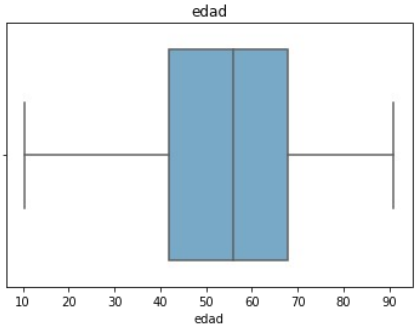


Fig. 22. Diagrama de caja de la variable edad después del tratamiento de outliers

Fuente: Elaboración propia

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Cuantil	0.00	1.0000	5.00	10.0	20.0	30.0	40.0	50.0	60.0	70.0	...	92.5	93.00	94.0	95.00	96.0	97.0	97.5	98.0	99.0	100.0
Valor	10.35	11.0925	26.25	32.5	39.0	44.5	50.0	56.0	61.0	65.0	...	80.0	80.85	81.3	82.75	83.0	84.0	84.0	85.0	91.0	91.0

Fig. 23. Análisis por cuantil de la variable edad después del tratamiento de outliers

Fuente: Elaboración propia

- **Saturación de Oxígeno**

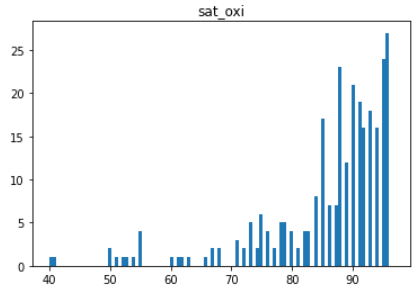


Fig. 24. Histograma de la variable saturación oxígeno

Fuente: Elaboración propia

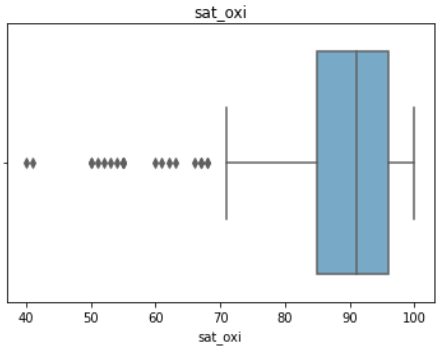


Fig. 25. Diagrama de caja de la variable saturación de oxígeno

Fuente: Elaboración propia

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Cuantil	0.0	1.00	5.0	10.0	20.0	30.0	40.0	50.0	60.0	70.0	...	92.5	93.0	94.0	95.0	96.0	97.0	97.5	98.0	99.0	100.0
Valor	40.0	50.45	67.0	75.0	84.0	87.0	89.0	91.0	93.0	95.0	...	98.0	98.0	98.0	98.0	98.0	98.0	99.0	99.1	100.0	100.0

Fig. 26. Análisis por cuantil de la variable saturación de oxígeno

Fuente: Elaboración propia

Tratamiento de outliers

```
quantile_1 = np.percentile(df['sat_oxi'],1)
df.loc[df['sat_oxi']<quantile_1,'sat_oxi'] = quantile_1
```

Fig. 27. Corrección del cuantil de la variable saturación de oxígeno

Fuente: Elaboración propia

Después del tratamiento de outliers

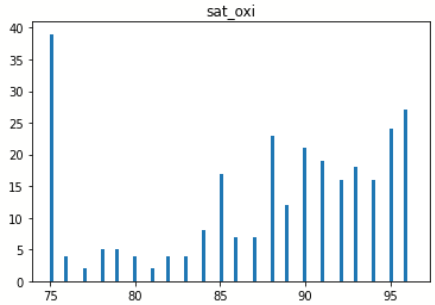


Fig. 28. Histograma de la variable saturación de oxígeno del tratamiento de outliers

Fuente: Elaboración propia

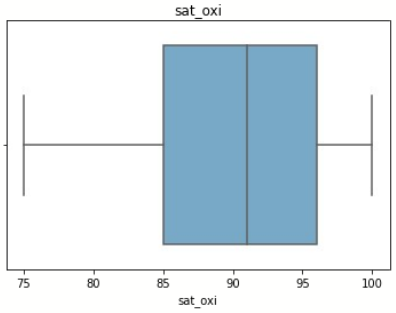


Fig. 29.Diagrama de caja de la variable saturación de oxígeno después del tratamiento de outliers

Fuente: Elaboración propia

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
Cuantil	0.00	1.0000	5.0	10.0	20.0	30.0	40.0	50.0	60.0	70.0	...	92.5	93.0	94.0	95.0	96.0	97.0	97.5	98.0	99.0	100.0
Valor	50.45	50.6975	67.0	75.0	84.0	87.0	89.0	91.0	93.0	95.0	...	98.0	98.0	98.0	98.0	98.0	98.0	99.0	99.1	100.0	100.0

Fig. 30.Análisis por cuantil de la variable saturación de oxígeno después del tratamiento de outliers

Fuente: Elaboración propia

Variables Categóricas:

- **Sexo**

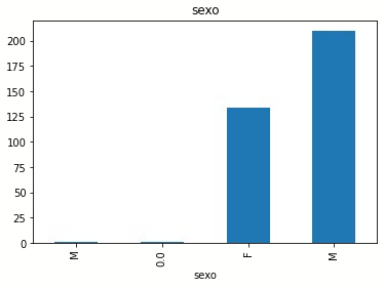


Fig. 31. Gráfica de barras de la variable sexo

Fuente: Elaboración propia

Como se muestra en la gráfica existen 4 valores diferentes de la columna sexo donde solo deberían existir dos valores 'F' y 'M'. Se buscará en el dataframe a los valores que no tienen 'F' ni 'M' como sexo.

	CODIGO	edad	sexo	diabetes	erp	ecc	hipertension	obesidad_sobrepeso	cancer	fiebre	...
91	2229.0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...
252	1038.0	65.0	M	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...

2 rows × 21 columns

Fig. 32. Datos sucios de la variable sexo

Fuente: Elaboración propia

Dado que se poseen los códigos será fácil encontrar los datos limpios de 2229 y 1038. Así el sexo de 2229 es 'F' y el de 1038 es 'M'. Es posible que el error se dé por algún espacio vacío en los laterales.

Quitando los espacios laterales:

```
df['sexo'] = df['sexo'].apply(lambda x: str(x).strip())
```

Fig. 33. Limpieza de la variable sexo

Fuente: Elaboración propia

Luego de esta pequeña limpieza:

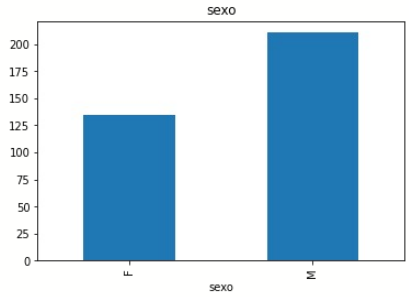


Fig. 34. Gráfico de barras de la variable sexo después de la limpieza

Fuente: Elaboración propia

De este gráfico de barras se puede observar que la mayor parte de la muestra son hombres y que la diferencia entre las dos categorías no es muy amplia.

- **Presencia de diabetes**

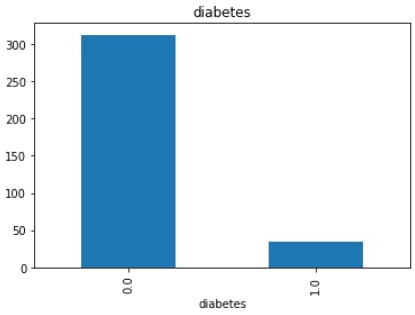


Fig. 35. Gráfica de barras de la variable presencia de diabetes

Fuente: Elaboración propia

Se observó un pequeño error que se verá continuamente en los datos categóricos de 0 y 1, por lo cual se implementó una función para convertirlo a tipo string.

```
def conversionVariableCategorica(column):
  df[column]=df[column].astype(float).apply(np.int64)
  df[column]=df[column].astype(str)
```

Fig. 36. Conversión a variable categórica

Fuente: Elaboración propia

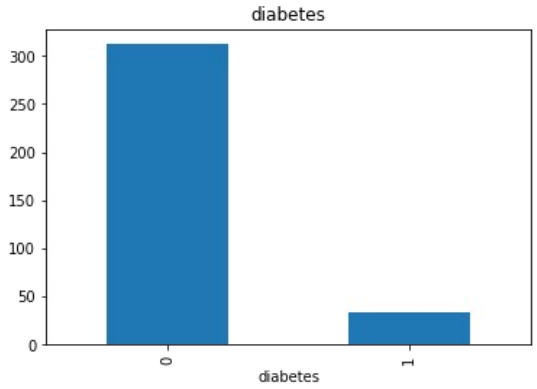


Fig. 37. Gráfica de barras de la variable presencia de diabetes después de la limpieza

Fuente: Elaboración propia

Podemos observar una gran diferencia entre la cantidad de personas que no tienen diabetes de las que sí.

- **Enfermedades respiratorias previas (ERP)**

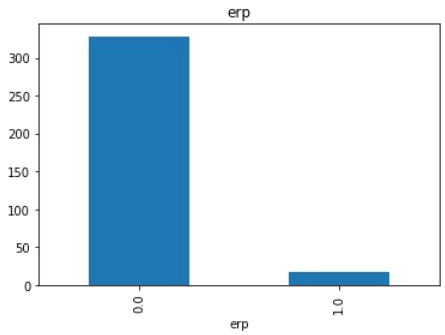


Fig. 38. Gráfica de barras de la variable presencia de ERP

Fuente: Elaboración propia

Aplicando la conversión:

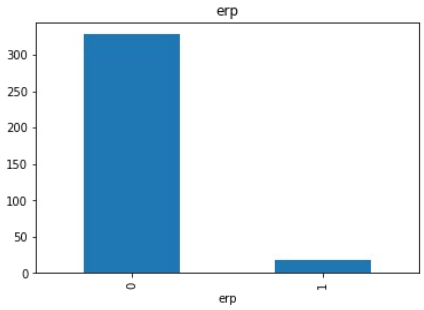


Fig. 39.Gráfica de barras la variable presencia de ERP después de la limpieza

Fuente: Elaboración propia

La mayoría de los pacientes de la muestra no registran enfermedades respiratorias previas.

- **Enfermedades cardiacas crónicas (ECC)**

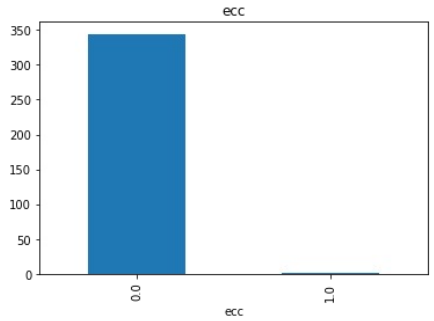


Fig. 40. Gráfica de barras de la variable presencia de ECC

Fuente: Elaboración propia

Aplicando la conversión:

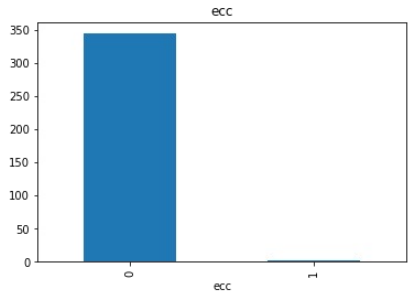


Fig. 41. Gráfica de barras de la variable presencia de ECC después de la limpieza

Fuente: Elaboración propia

La mayoría de los pacientes de la muestra no registran enfermedades cardiacas crónicas.

- **Presencia de hipertensión**

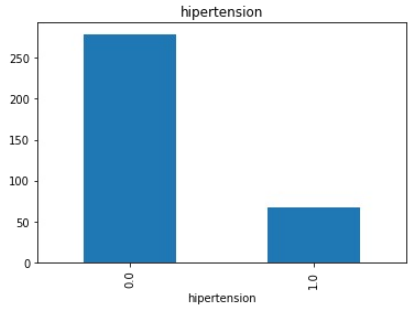


Fig. 42. Gráfico de barras de la variable presencia de la hipertensión

Fuente: Elaboración propia

Aplicando la conversión:

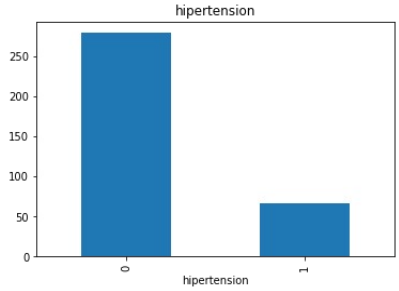


Fig. 43. Gráfico de barras de la variable presencia de la hipertensión después de la limpieza

Fuente: Elaboración propia

- **Presencia de obesidad/sobrepeso**

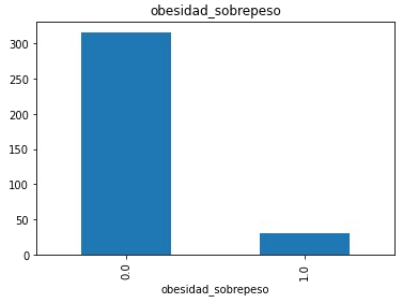


Fig. 44. Gráfica de barras de la presencia de obesidad o sobrepeso

Fuente: Elaboración propia

Aplicando la conversión:

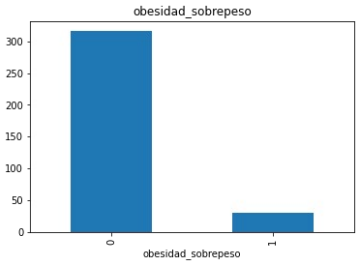


Fig. 45. Gráfica de barras de la presencia de obesidad o sobrepeso después de la limpieza

Fuente: Elaboración propia

- **Presencia de cáncer**

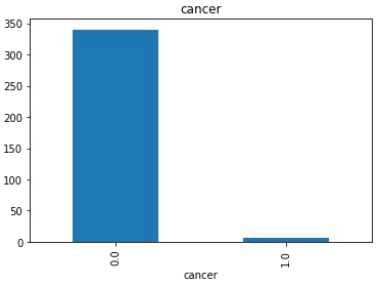


Fig. 46. Gráfica de barras de la presencia de cáncer

Fuente: Elaboración propia

Aplicando la conversión:

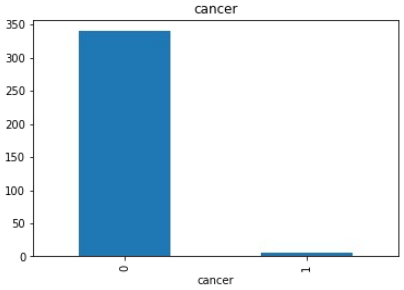


Fig. 47. Gráfica de barras de la presencia de cáncer después de la limpieza

Fuente: Elaboración propia

- **Presencia de fiebre**

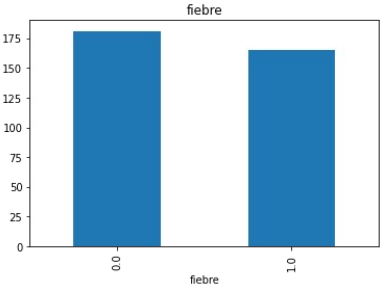


Fig. 48. Gráfica de barras de la presencia de fiebre

Fuente: Elaboración propia

Aplicando la conversión:

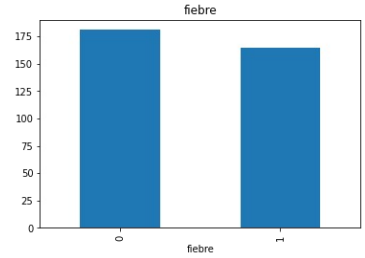


Fig. 49. Gráfica de barras de la presencia de fiebre después de la limpieza

Fuente: Elaboración propia

- **Presencia de tos**

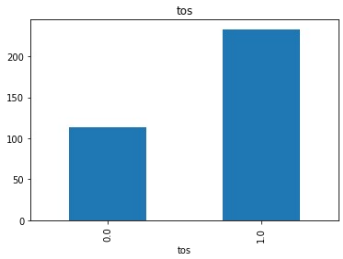


Fig. 50. Gráfica de barras de la presencia de tos

Fuente: Elaboración propia

Aplicando la conversión:

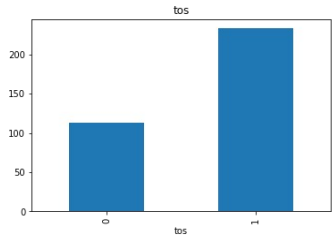


Fig. 51. Gráfica de barras de la presencia de tos después de la limpieza

Fuente: Elaboración propia

- **Presencia de malestar general**

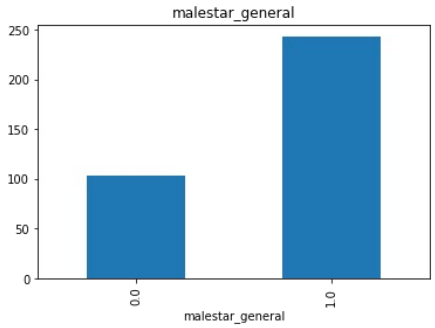


Fig. 52. Gráfica de barras de la presencia de malestar general

Fuente: Elaboración propia

Aplicando la conversión:

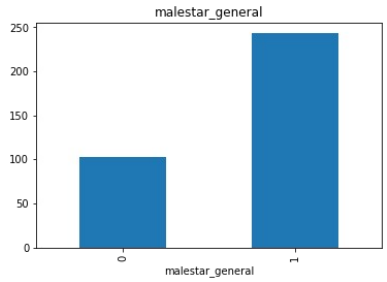


Fig. 53. Gráfico de barras de la presencia de malestar general después de la limpieza

Fuente: Elaboración propia

- **Presencia de diarrea**

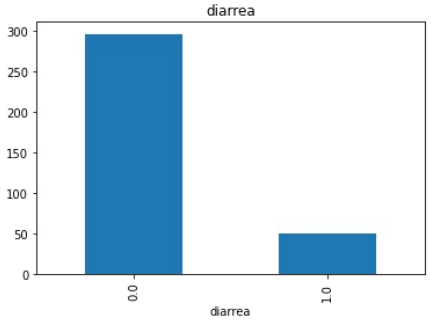


Fig. 54. Gráfico de barras de la presencia de diarrea

Fuente: Elaboración propia

Aplicando la conversión:

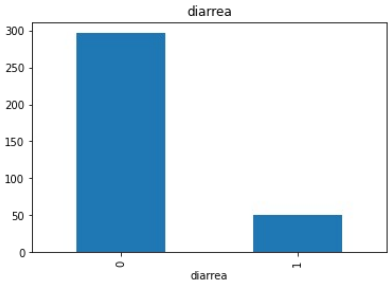


Fig. 55. Gráfico de barras de la presencia de diarrea después de la limpieza

Fuente: Elaboración propia

- **Presencia de mareos**

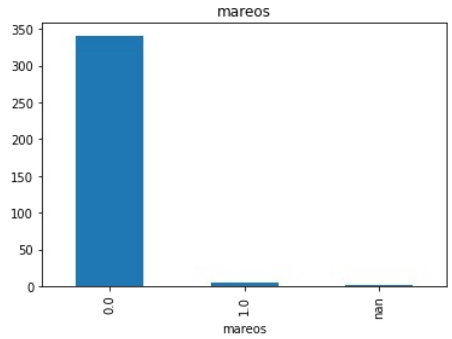


Fig. 56. Gráfica de barras de la variable presencia de mareos

Fuente: Elaboración propia

Aplicando la conversión:

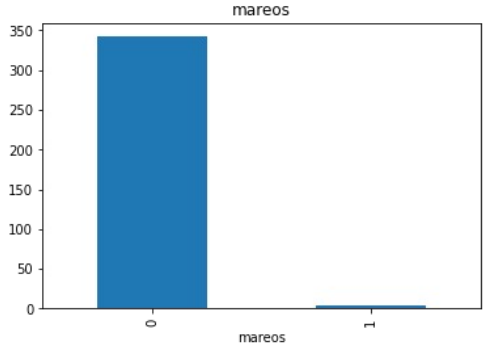


Fig. 57. Gráfico de barras de la variable presencia de mareos después de la limpieza

Fuente: Elaboración propia

- **Presencia de náuseas**

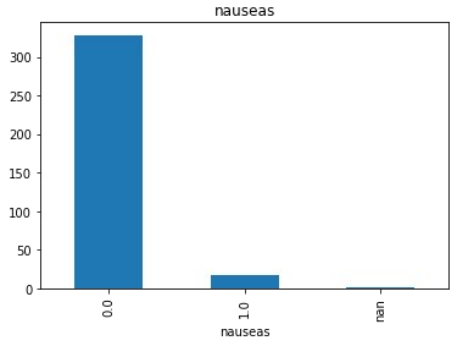


Fig. 58. Gráfico de barras de la variable presencia de náuseas

Fuente: Elaboración propia

Aplicando la limpieza correspondiente:

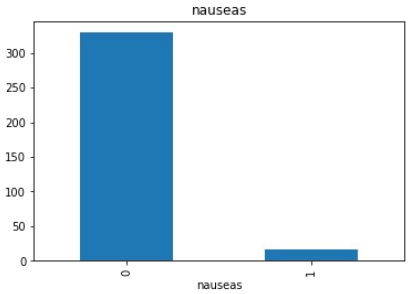


Fig. 59. Gráfico de barras de la variable presencia de nauseas después de la limpieza

Fuente: Elaboración propia

- **Presencia de escalofríos**

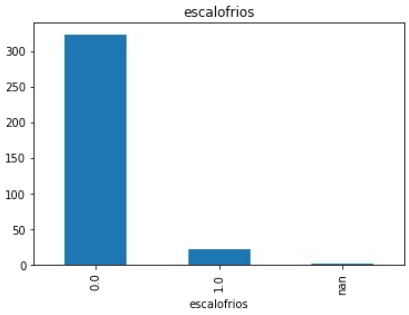


Fig. 60. Gráfico de barras de la variable presencia de escalofríos

Fuente: Elaboración propia

Aplicando la limpieza:

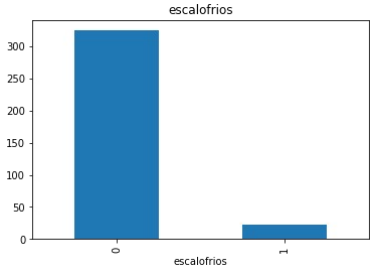


Fig. 61. Gráfico de barras de la variable presencia de escalofríos después de la limpieza

Fuente: Elaboración propia

- **Presencia de cansancio**

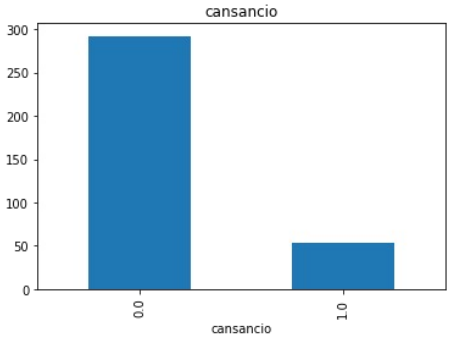


Fig. 62. Gráfico de barras de la variable presencia de cansancio

Fuente: Elaboración propia

Aplicando la conversión:

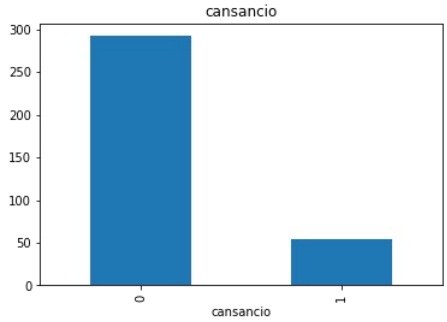


Fig. 63.Gráfico de barras de la variable presencia de cansancio después de la limpieza

Fuente: Elaboración propia

- **Presencia de disnea**

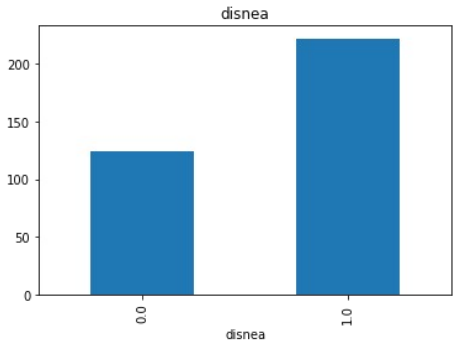


Fig. 64. Gráfico de barras de la variable presencia de disnea

Fuente: Elaboración propia

Aplicando la conversión:

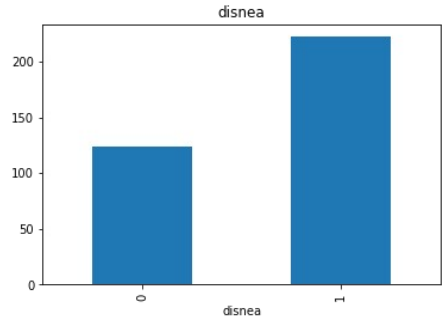


Fig. 65. Gráfico de barras de la variable presencia de disnea después de la limpieza

Fuente: Elaboración propia

- **Número de dosis**

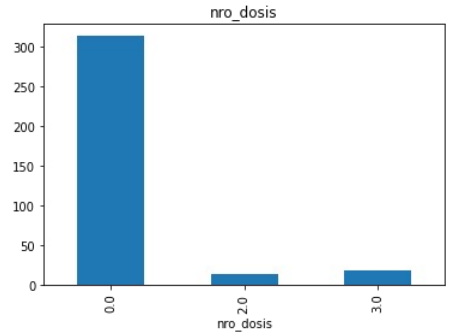


Fig. 66. Gráfico de barras de la variable número de dosis

Fuente: Elaboración propia

Aplicando la conversión:

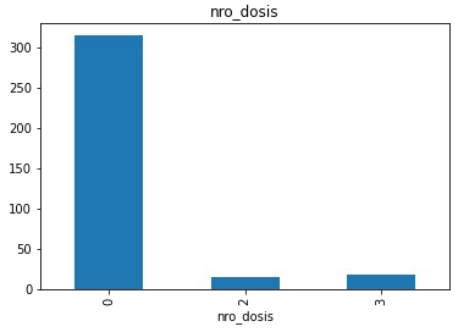


Fig. 67.Gráfico de barras de la variable número de dosis después de la limpieza

Fuente: Elaboración propia

- **Fallecimiento**

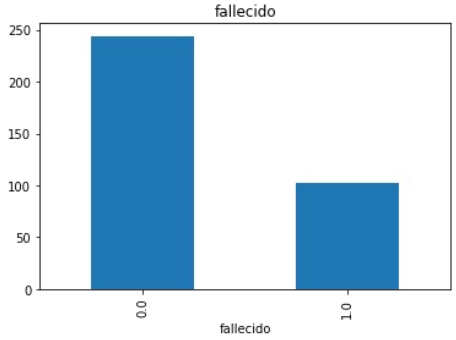


Fig. 68. Gráfico de barras de la variable fallecimiento

Fuente: Elaboración propia

Aplicando la conversión:

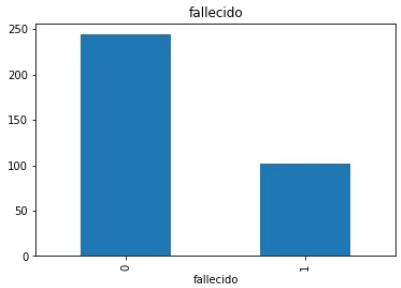


Fig. 69. Gráfico de barras de la variable fallecimiento después de la limpieza

Fuente: Elaboración propia

ANEXO N° 08. PREPROCESAMIENTO DE LAS VARIABLES

```
from imblearn.over_sampling import SMOTENC
oversample = SMOTENC(random_state=100,categorical_features=categorical_features)
oversample_fit = oversample.fit(X,y)
x_oversampled,y_oversampled = oversample.fit_resample(X,y)
df = pd.DataFrame(x_oversampled,columns=df.drop('fallecido',axis=1).columns)
df['fallecido'] = pd.DataFrame(y_oversampled)
```

Fig. 70. Oversampling aleatorio de la muestra

Fuente: Elaboración propia

```
from sklearn.preprocessing import LabelEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
```

Fig. 71. Codificación de las variables categóricas

Fuente: Elaboración propia

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X = sc_X.fit_transform(X)
```

Fig. *72*. Escalamiento de la muestra

Fuente: Elaboración propia

ANEXO N° 09. ENTRENAMIENTO DE LA RNA

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)
```

Fig. 73. División en grupos de entrenamiento y prueba
Fuente: Elaboración propia

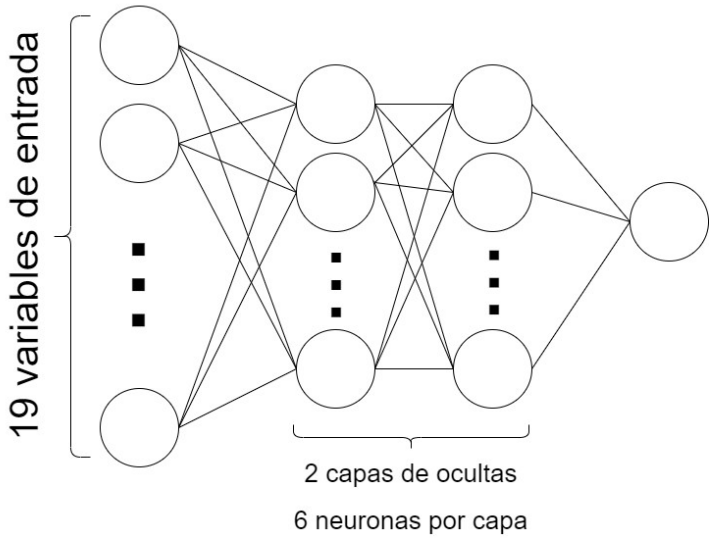


Fig. *74*. Diagrama de la RNA con 19 variables de entradas

Fuente: Elaboración propia

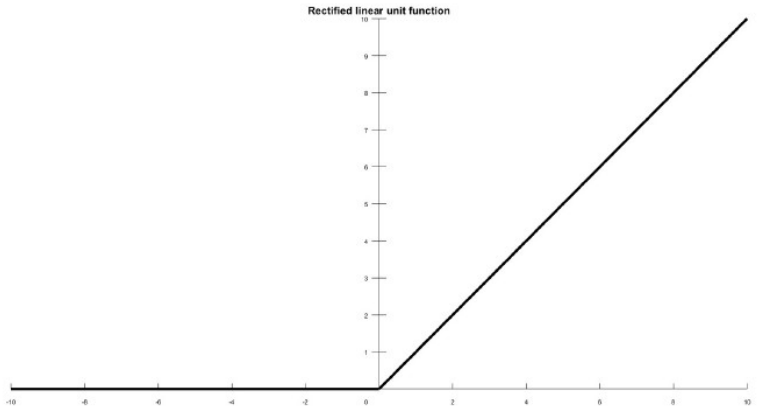


Fig. *75*. Gráfica ReLU

Fuente: Extraído de [52]

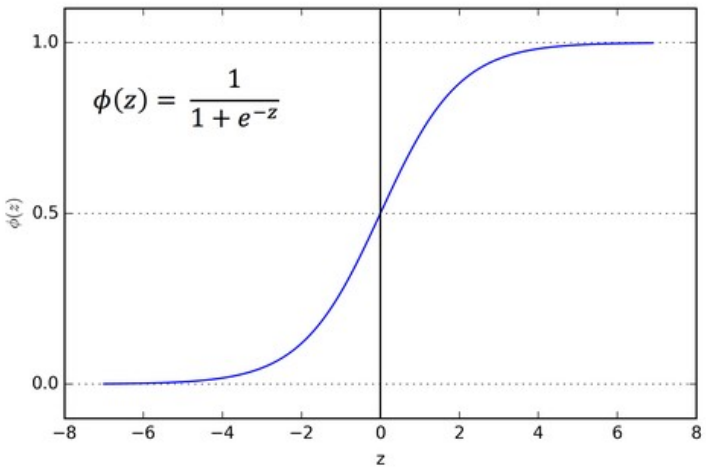


Fig. *76*. Gráfica sigmoidal

Fuente: Extraído de [53]

```
# Inicializar la RNA
classifier = Sequential()
# Añadir las capas de entrada y primera capa oculta
classifier.add(Dense(units = 10, kernel_initializer = "uniform",
                     activation = "relu", input_dim = 19))
# Añadir la segunda capa oculta
classifier.add(Dense(units = 10, kernel_initializer = "uniform",  activation = "relu"))
# Añadir la capa de salida
classifier.add(Dense(units = 1, kernel_initializer = "uniform",  activation =
"sigmoid"))
# Compilar la RNA
classifier.compile(optimizer = "adam", loss = "binary_crossentropy", metrics =
["accuracy"])
# Ajustamos la RNA al Conjunto de Entrenamiento
classifier.fit(X_train, y_train,  batch_size = 10, epochs = 100)
```

Fig. 77. Construcción de la primera RNA

Fuente: Elaboración propia

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
def build_classifier():
  # Inicializar la RNA
  classifier = Sequential()
  # Añadir las capas de entrada y primera capa oculta
  classifier.add(Dense(units = 10, kernel_initializer = "uniform",
                       activation = "relu", input_dim = 19))
  # Añadir la segunda capa oculta
  classifier.add(Dense(units = 10, kernel_initializer = "uniform",  activation = "relu"))
  # Añadir la capa de salida
  classifier.add(Dense(units = 1, kernel_initializer = "uniform",  activation = "sigmoid"))
  # Compilar la RNA
  classifier.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
  # Devolver el clasificador
  return classifier

classifier = KerasClassifier(build_fn = build_classifier, batch_size = 10, nb_epoch = 100)
accuracies = cross_val_score(
  estimator=classifier, X = X_train, y = y_train, cv = 10, n_jobs=-1, verbose = 1)
std= accuracies.std()
```

Fig. 78. Aplicación de 'cross_val_score'

Fuente: Elaboración propia

```python
def build_classifier(optimizer):
  # Inicializar la RNA
  classifier = Sequential()

  # Añadir las capas de entrada y primera capa oculta
  classifier.add(Dense(units = 10, kernel_initializer = "uniform",
                       activation = "relu", input_dim = 19))
  classifier.add(Dropout(rate = 0.1))

  # Añadir la segunda capa oculta
  classifier.add(Dense(units = 10, kernel_initializer = "uniform",  activation = "relu"))
  classifier.add(Dropout(rate = 0.1))

  # Añadir la capa de salida
  classifier.add(Dense(units = 1, kernel_initializer = "uniform",  activation = "sigmoid"))

  # Compilar la RNA
  classifier.compile(optimizer = optimizer, loss = "binary_crossentropy", metrics = ["accuracy"])

  # Devolver el clasificador
  return classifier
```

Fig. 79. Capa de dropout agregada a la RNA

Fuente: Elaboración propia

TABLA V

COMPARACIÓN VALORES DROPOUT

	DROP OUT 0.1	DROP OUT 0.2	DROP OUT 0.3	DROP OUT 0.4	DROP OUT 0.5
Des. Est #1	0.1361	0.0980	0.1392	0.1161	0.0863
Des. Est #2	0.11616	0.12100	0.1012	0.1035	0.1434
Des. Est #3	0.1417	0.1072	0.0807	0.1135	0.0834
Des. Est #4	0.1388	0.1338	0.1178	0.1105	0.0966
Des. Est #5	0.146	0.0816	0.1168	0.1231	0.065

```python
from sklearn.model_selection import GridSearchCV
# build classifier es la arquitectura de la rna propuesta en modo de función
classifier = KerasClassifier(build_fn = build_classifier)
# En un diccionario se colocan todos los parámetros que deseamos comparar para
# encontrar los mejores
parameters = {
    'batch_size' : [10,25,32],
    'nb_epoch' : [100, 500],
    'optimizer' : ['adam', 'rmsprop']
}
# Con la clase GridSearchCV se buscan los mejores parámetros utilizando la validación
# cruzada, el número de 10 es el número de subgrupos en lo que se dividirán los datos
# para ser evaluados
grid_search = GridSearchCV(estimator = classifier,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 10)
# Con el método fit se comienza la busqueda, y podemos observar que se han accedido
# a los atributos de best_params_(mejores parámetros encontrados)
# y best_score_ (mejor métrica encontrada)
grid_search = grid_search.fit(X_train, y_train,verbose=0)
best_parameters = grid_search.best_params_
best_accuracy = grid_search.best_score_
```

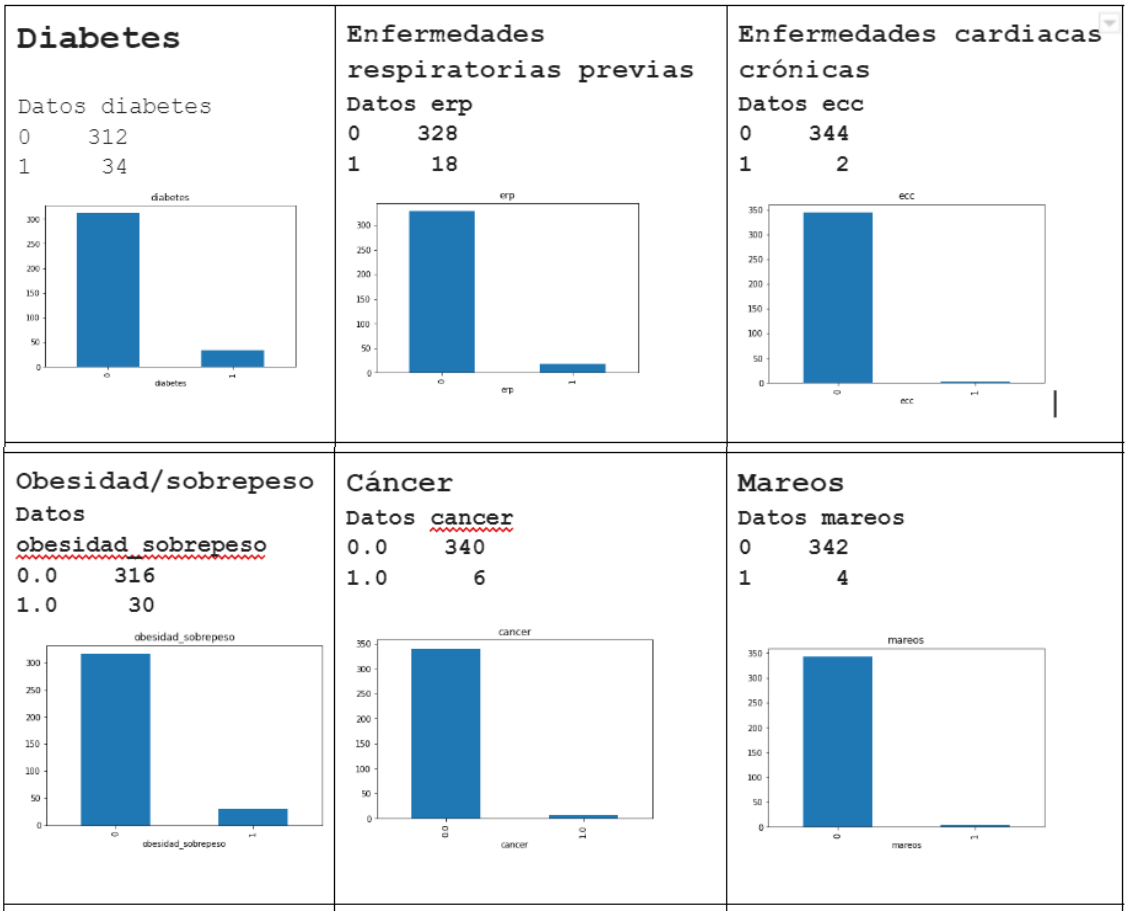
Fig. 80. Aplicación de GridSearchCV

Fuente: Elaboración propia

TABLA VI.

COMPARACIÓN SEGÚN LOS CRITERIOS DE EVALUACIÓN

PARÁMETROS	DESV. EST.	AUROC	EXACTITUD	F-MEASURE
Batch_size = 3 Nb_epoch=250	0.10	0.88	0.83	0.85
Batch_size = 3 Nb_epoch=350	0.14	0.86	0.80	0.81
Batch_size = 5 Nb_epoch=350	0.15	0.84	0.80	0.82
Batch_size = 3 Nb_epoch=450	0.11	0.83	0.78	0.80
Batch_size = 3 Nb_epoch=500	0.12	0.81	0.75	0.77



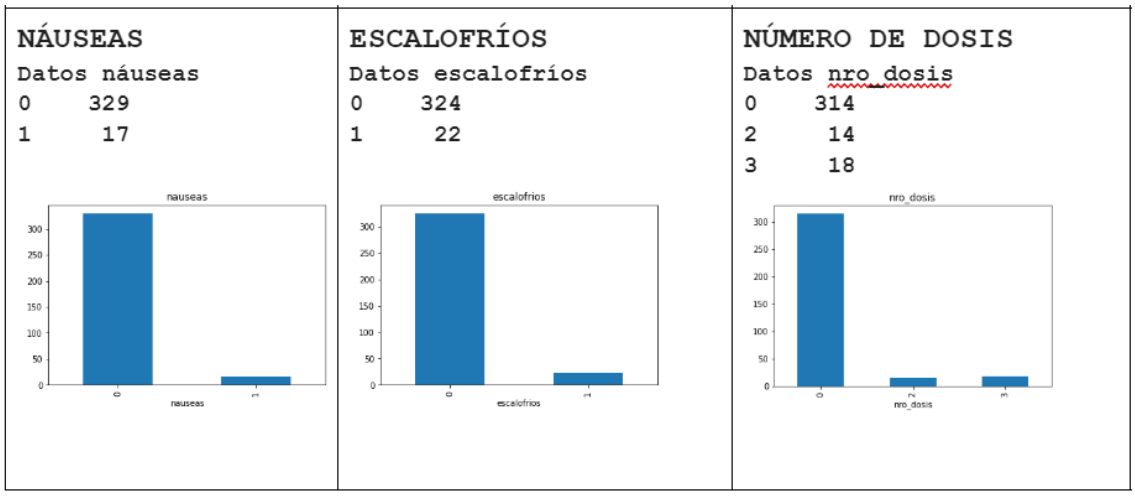


Fig. 81. Variables con menos datos representativos

Fuente: Elaboración propia

TABLA VII.

RESUMEN DE LAS DIFERENTES RNA

PARÁMETROS	NRO. INPUTS	EXACTITUD	DESV. EST.	F-MEASURE	AUROC
DROP_RATE = 0.3 BATCH_SIZE = 5 NB_EPOCHS =350 FIRST_HIDDEN_NEURONS = 10 SECOND_HIDDEN_NEURONS = 10	10	0.8209	0.0948	0.8342	0.89
DROP_RATE = [0.2,0.4] BATCH_SIZE = 5 NB_EPOCHS =350 FIRST_HIDDEN_NEURONS = 10 SECOND_HIDDEN_NEURONS = 10	10	0.8209	0.1244	0.8379	0.89
DROP_RATE = 0.4 BATCH_SIZE = 2 NB_EPOCHS =256 FIRST_HIDDEN_NEURONS = 24 SECOND_HIDDEN_NEURONS = 24	10	0.8271	0.0888	0.8352	0.88
DROP_RATE = [0.2,0.4] BATCH_SIZE = 4 NB_EPOCHS =350 FIRST_HIDDEN_NEURONS = 8 SECOND_HIDDEN_NEURONS = 10	10	0.8148	0.1138	0.8275	0.89

DROP_RATE = 0.2 BATCH_SIZE = 5 NB_EPOCHS =350 FIRST_HIDDEN_NEURONS = 8 SECOND_HIDDEN_NEURONS = 8	13	0.7962	0.1253	0.8114	0.84
DROP_RATE = [0.2,0.4] BATCH_SIZE = 5 NB_EPOCHS =350 FIRST_HIDDEN_NEURONS = 10 SECOND_HIDDEN_NEURONS = 10	16	0.7777	0.0896	0.8105	0.89
DROP_RATE = 0.3 BATCH_SIZE = 4 NB_EPOCHS =350 FIRST_HIDDEN_NEURONS = 16 SECOND_HIDDEN_NEURONS = 16	19	0.80	0.11	0.81	0.83
DROP_RATE = 0.3 BATCH_SIZE = 16 NB_EPOCHS =256 FIRST_HIDDEN_NEURONS = 20 SECOND_HIDDEN_NEURONS = 16	19	0.81	0.12	0.83	0.85
DROP_RATE = 0.3 BATCH_SIZE = 4 NB_EPOCHS =256 FIRST_HIDDEN_NEURONS = 24 SECOND_HIDDEN_NEURONS = 24	19	0.79	0.0995	0.81	0.84

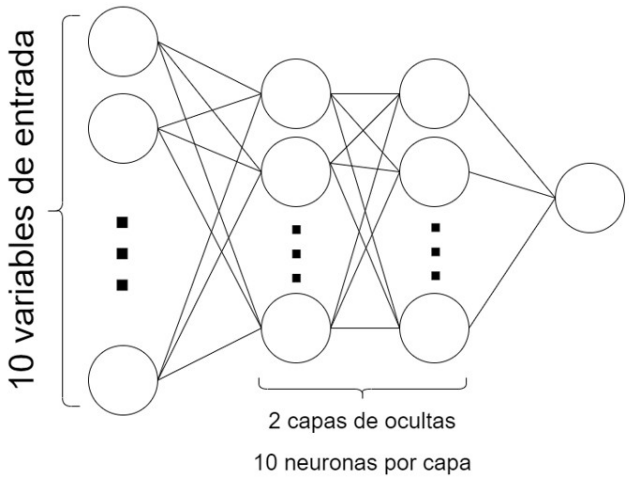


Fig. 82. Mejor arquitectura de RNA obtenida

Fuente: Elaboración propia

ANEXO N° 10. EVALUACIÓN DEL PRIMER MODELO

Datos de entrenamiento

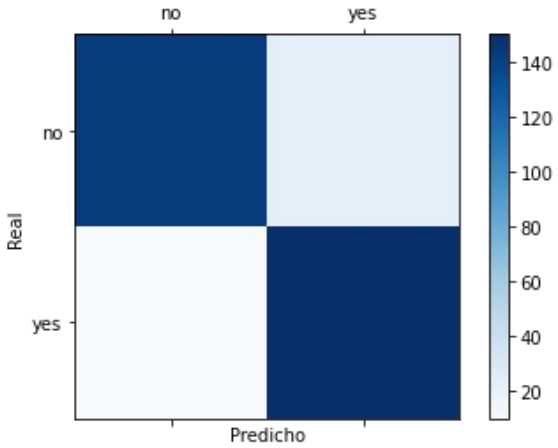


Fig. 83. Matriz de confusión de los datos de entrenamiento

Fuente: Elaboración propia

Matriz de confusión

$$\begin{bmatrix} 143 & 23 \\ 10 & 150 \end{bmatrix}$$

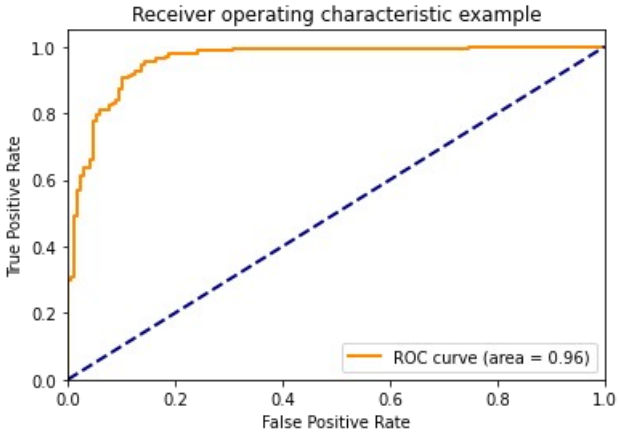


Fig. 84. Gráfico ROC de los datos de entrenamiento

Fuente: Elaboración propia

Datos de prueba

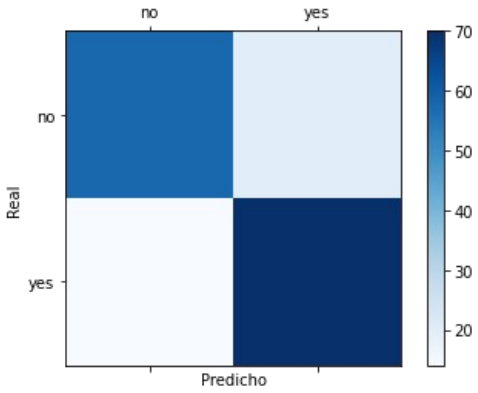


Fig. 85 .Matriz de confusión de los datos de entres de prueba

Fuente: Elaboración propia

Matriz de confusión

$$\begin{bmatrix} 58 & 20 \\ 14 & 70 \end{bmatrix}$$

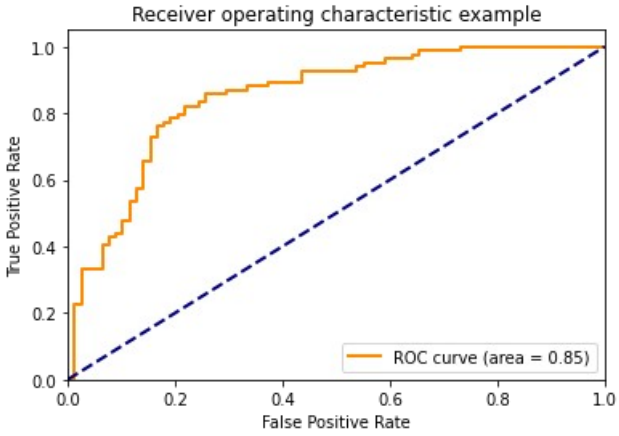


Fig. 86.Gráfico ROC de los datos de prueba

Fuente: Elaboración propia

	Exactitud-Accuracy	Exactitud equilibrada	Precision-Precision	Sensibilidad-Recall	F1Score(precision y sensibilidad)
Train	0.898773	0.899473	0.867052	0.937500	0.900901
Test	0.790123	0.788462	0.777778	0.833333	0.804598

Fig. 87. Comparación entre los datos de entrenamiento y prueba

Fuente: Elaboración propia

ANEXO N° 11. IMPLEMENTACIÓN

```
from joblib import dump
dump(sc_X, ROOT_PATH+'std_scaler.bin', compress=True)

#Para la carga del escalador se usará la función load de la misma librería
from joblib import load
sc_X=load('std_scaler.bin')
```

Fig. 88. Guardar y cargar el escalador unitario

Fuente: Elaboración propia

```
from keras.models import load_model
classifier.save("model10inputs08052022.h5")
# Para cargar la RNA
classifier = load_model("model10inputs08052022.h5")
```

Fig. 89. Guardar y cargar el modelo de RNA

Fuente: Elaboración propia

ANEXO N° 12. INTERFACES DEL SISTEMA INTELIGENTE WEB

Fig. 90. Diseño de la interfaz inicio de sesión

Fuente: Elaboración propia

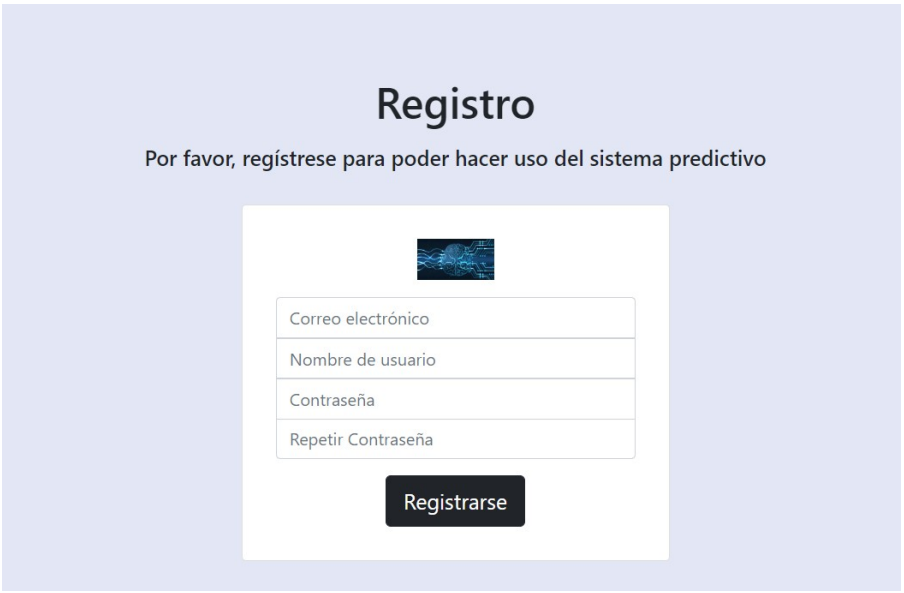


Fig. 91. Diseño de la interfaz registro de usuario

Fuente: Elaboración propia

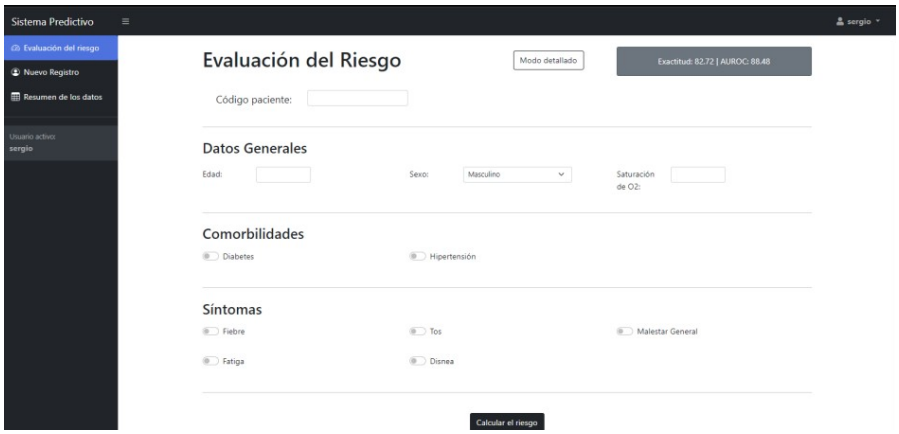


Fig. 92. Diseño de la interfaz de evaluación de riesgo – Modo simple

Fuente: Elaboración propia

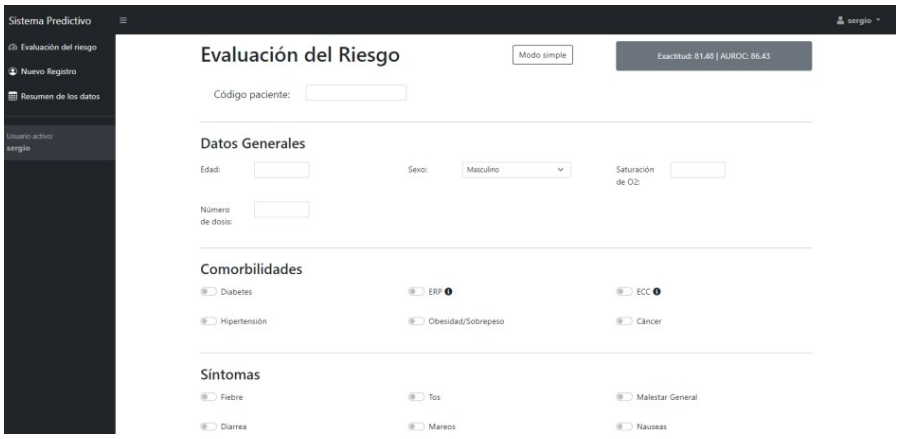


Fig. 93. Diseño de la interfaz de evaluación de riesgo – Modo detallado

Fuente: Elaboración propia

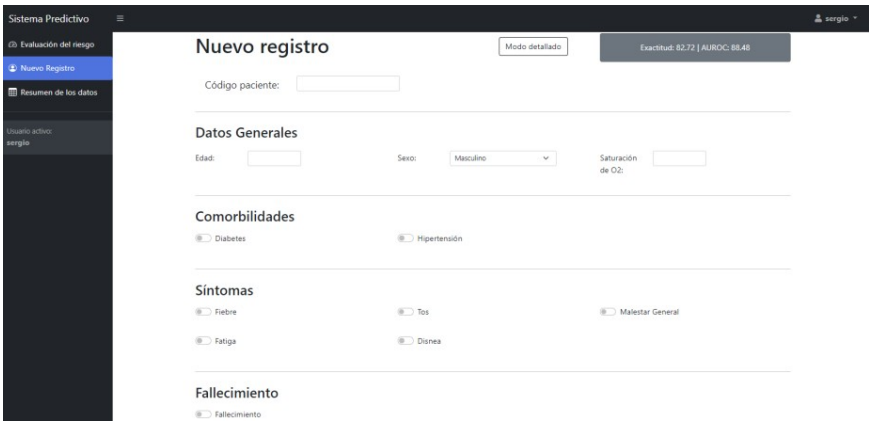


Fig. 94. Diseño de la interfaz de nuevo registro – Modo simple

Fuente: Elaboración propia

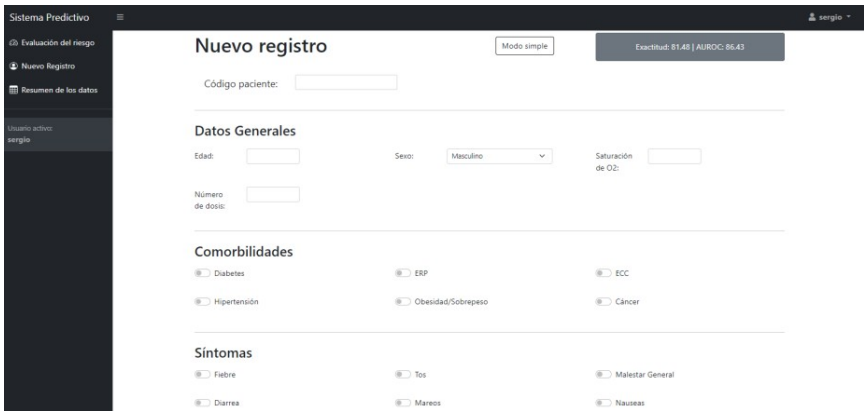


Fig. 95. Diseño de la interfaz de nuevo registro - Modo detallado

Fuente: Elaboración propia

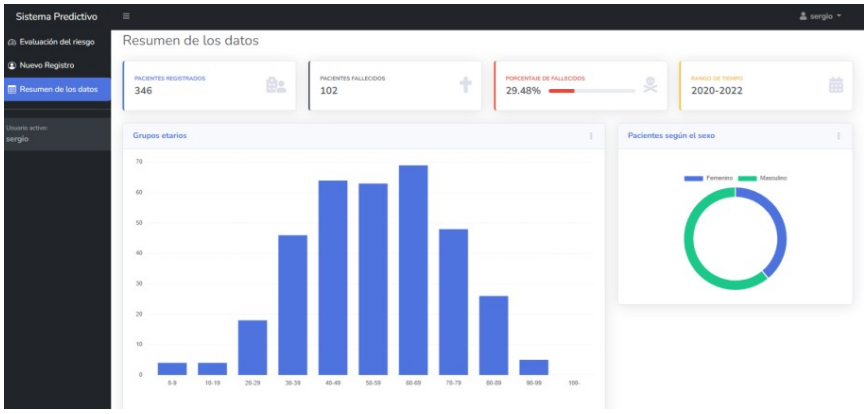


Fig. 96. Diseño de la interfaz de resumen de datos

Fuente: Elaboración propia

ANEXO N° 13. IMPLEMENTACIÓN DEL SISTEMA INTELIGENTE WEB

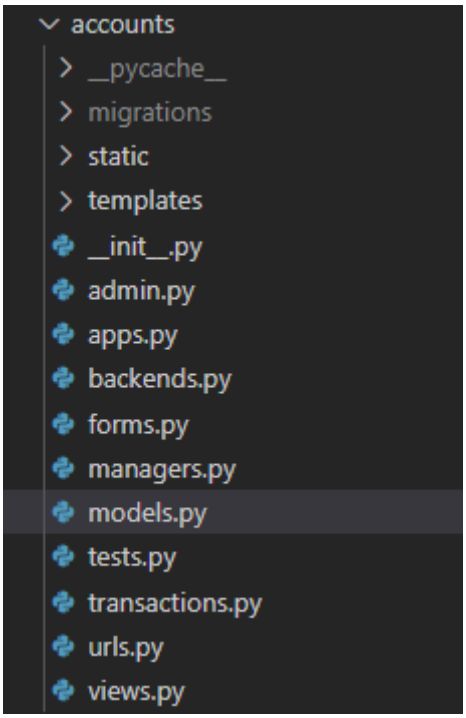


Fig. 97. Organización de archivos de la aplicación de usuarios

Fuente: Elaboración propia

```python
class UserRegisterView(FormView):
    template_name = "accounts/register.html"
    form_class = UserRegisterForm
    success_url = reverse_lazy('accounts_app:result-register')

    def form_valid(self,form):
        user=User.objects.create_user(
            form.cleaned_data['username'],
            form.cleaned_data['email'],
            form.cleaned_data['password'],
        )
        user.save()
        returnVal=super(UserRegisterView,self).form_valid(form)
        return returnVal
```

Fig. 98. Vista UserRegisterView

Fuente: Elaboración propia

```python
def clean(self):
    username = self.cleaned_data['username']
    email = self.cleaned_data['email']
    password=self.cleaned_data['password']
    repeated_password=self.cleaned_data['repeated_password']
    usuario= User.objects.validate_username(username)
    email_valid= User.objects.validate_email(email)
    if usuario:
        raise forms.ValidationError('El nombre de usuario ya existe, ingrese otro')
    if email_valid:
        raise forms.ValidationError('El email ya se encuentra en uso, ingrese otro')
    if validate_password(password) != None:
        raise forms.ValidationError('Ingrese una contraseña')
    if password != repeated_password:
        raise forms.ValidationError('ERROR, Las contraseñas no coinciden')
```

Fig. 99. Función clean de UserRegisterForm

Fuente: Elaboración propia

```python
class LoginUser(FormView):
    template_name = "accounts/login.html"
    form_class = LoginForm
    success_url = reverse_lazy('home_app:home')
    def form_valid(self,form):
        #
        user = authenticate(
            email=form.cleaned_data['email'],
            password=form.cleaned_data['password'],

        )
        login(self.request, user)
        return super(LoginUser,self).form_valid(form)
```

Fig. 100. Vista LoginUser

Fuente: Elaboración propia

```python
def clean(self):
    cleaned_data=super(LoginForm,self).clean()
    email=self.cleaned_data['email']
    password=self.cleaned_data['password']

    if not authenticate(email=email,password=password):
        try:
            if User.objects.get(email=email).is_active == False :
               raise forms.ValidationError(
                 'El usuario no está activo,comuníquese con la institución para la habilitación.'
                       )
        except Exception as e:
            raise forms.ValidationError('Los datos no son correctos, vuelva intentar')
```

Fig. 101. Función clean del LoginForm

Fuente: Elaboración propia

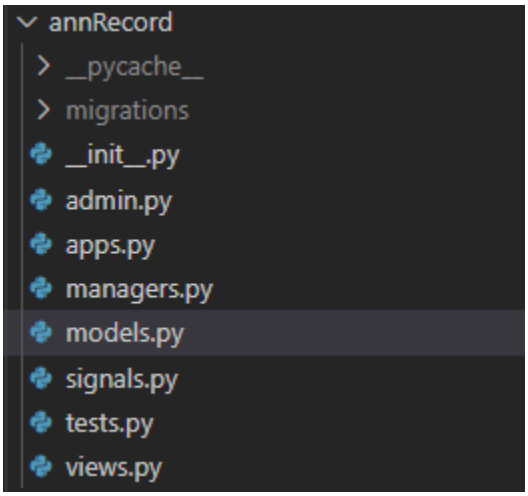


Fig. 102. Archivos de la aplicación carga de RNA

Fuente: Elaboración propia

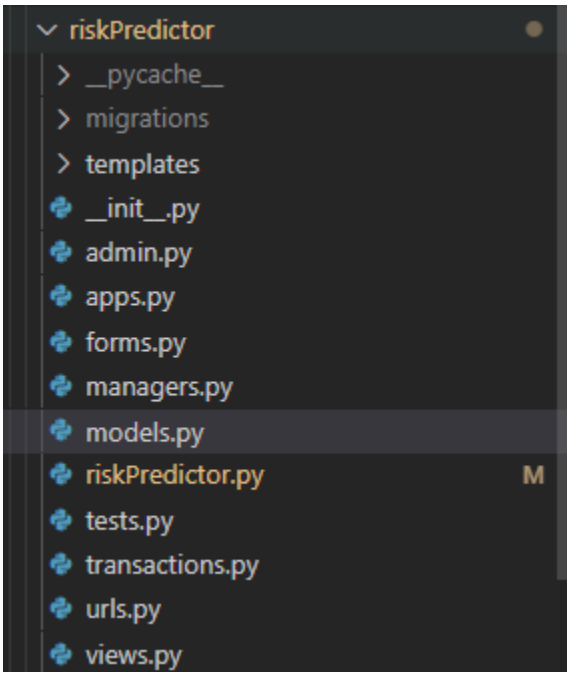


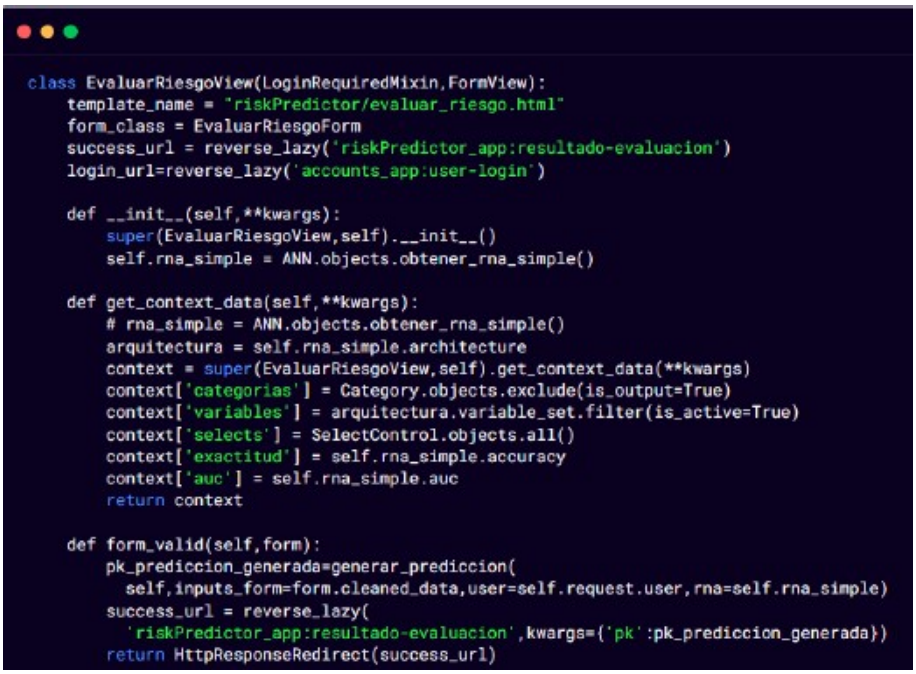
Fig. 103. Archivos de la aplicación Predictor de riesgo

Fuente: Elaboración propia

```python
class Prediction(TimeStampedModel):
    """Prediccion final de la red neuronal"""

    # TODO: Define fields here
    ann = models.ForeignKey(
        ANN, on_delete=models.CASCADE,verbose_name='RNA',blank=True,null=True)
    user = models.ForeignKey(
        User, on_delete=models.CASCADE,verbose_name='Usuario')
    value =models.FloatField(
        'Valor resultado',blank=True,null=True)
    string_value=models.CharField(
        'Resultado',default='',max_length=50
    )
    class Meta:
        """Meta definition for Prediction."""

        verbose_name = 'Predicción'
        verbose_name_plural = 'Predicciones'

    def __str__(self):
        """Unicode representation of Prediction."""
        return f'Predicción: {self.id}'
```

Fig. 104. Clase Prediction

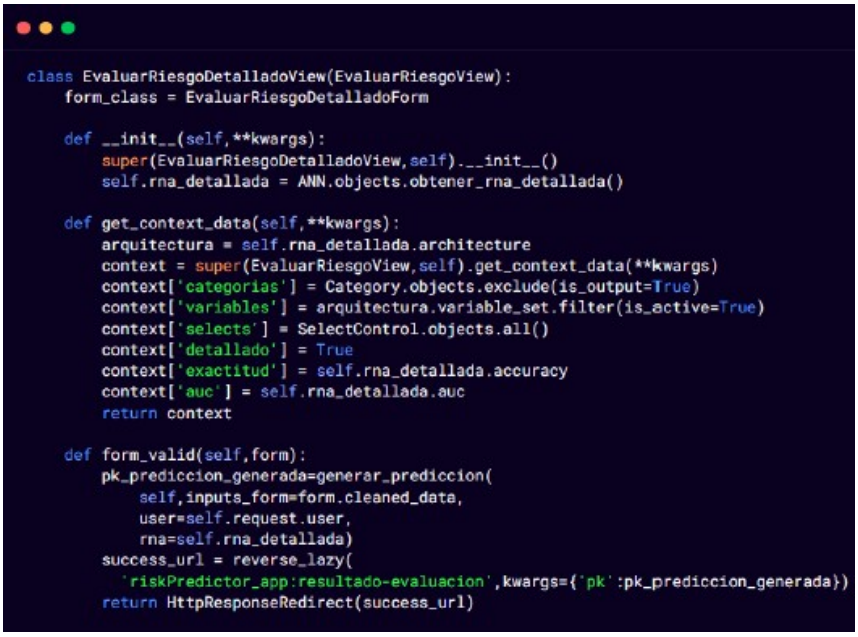
Fuente: Elaboración propia



```python
class EvaluarRiesgoView(LoginRequiredMixin,FormView):
    template_name = "riskPredictor/evaluar_riesgo.html"
    form_class = EvaluarRiesgoForm
    success_url = reverse_lazy('riskPredictor_app:resultado-evaluacion')
    login_url=reverse_lazy('accounts_app:user-login')

    def __init__(self,**kwargs):
        super(EvaluarRiesgoView,self).__init__()
        self.rna_simple = ANN.objects.obtener_rna_simple()

    def get_context_data(self,**kwargs):
        # rna_simple = ANN.objects.obtener_rna_simple()
        arquitectura = self.rna_simple.architecture
        context = super(EvaluarRiesgoView,self).get_context_data(**kwargs)
        context['categorias'] = Category.objects.exclude(is_output=True)
        context['variables'] = arquitectura.variable_set.filter(is_active=True)
        context['selects'] = SelectControl.objects.all()
        context['exactitud'] = self.rna_simple.accuracy
        context['auc'] = self.rna_simple.auc
        return context

    def form_valid(self,form):
        pk_prediccion_generada=generar_prediccion(
          self,inputs_form=form.cleaned_data,user=self.request.user,rna=self.rna_simple)
        success_url = reverse_lazy(
          'riskPredictor_app:resultado-evaluacion',kwargs={'pk':pk_prediccion_generada})
        return HttpResponseRedirect(success_url)
```

Fig. 105. Vista EvaluarRiesgoView

Fuente: Elaboración propia



```python
class EvaluarRiesgoDetalladoView(EvaluarRiesgoView):
    form_class = EvaluarRiesgoDetalladoForm

    def __init__(self,**kwargs):
        super(EvaluarRiesgoDetalladoView,self).__init__()
        self.rna_detallada = ANN.objects.obtener_rna_detallada()

    def get_context_data(self,**kwargs):
        arquitectura = self.rna_detallada.architecture
        context = super(EvaluarRiesgoView,self).get_context_data(**kwargs)
        context['categorias'] = Category.objects.exclude(is_output=True)
        context['variables'] = arquitectura.variable_set.filter(is_active=True)
        context['selects'] = SelectControl.objects.all()
        context['detallado'] = True
        context['exactitud'] = self.rna_detallada.accuracy
        context['auc'] = self.rna_detallada.auc
        return context

    def form_valid(self,form):
        pk_prediccion_generada=generar_prediccion(
            self,inputs_form=form.cleaned_data,
            user=self.request.user,
            rna=self.rna_detallada)
        success_url = reverse_lazy(
          'riskPredictor_app:resultado-evaluacion',kwargs={'pk':pk_prediccion_generada})
        return HttpResponseRedirect(success_url)
```

Fig. 106. Vista EvaluarRiesgoDetalladoView

Fuente: Elaboración propia

```python
def generar_prediccion(self,**params):
    """
    Al calcular la predicción se debe:
    1. Encontrar la variable referida a cada input segun el nombre del form
    2. Asignar al input la variable y el valor determinado
    3. Crear una predicción
    4. Asignar al input la predicción generada
    """
    try:
        with transaction.atomic():
            objAnn= ANN.objects.filter(
                    is_active=True
                ).last()
            diccionario_form=without_keys(params['inputs_form'],'codigo_paciente')
            codigo_paciente = params['inputs_form']['codigo_paciente']
            objPrediction=Prediction.objects.create(
                    user=params['user'],
                    ann=objAnn
                )
            genero= 1 if diccionario_form['genero']=='M' else 0
            diccionario_respuestas=predict(
                rna = objAnn,
                edad=diccionario_form['edad'],
                sexo=genero,
                hipertension=diccionario_form['hipertension'],
                diabetes=diccionario_form['diabetes'],
                fiebre=diccionario_form['fiebre'],
                tos=diccionario_form['tos'],
                malestar_general=diccionario_form['malestar_general'],
                cansancio=diccionario_form['cansancio'],
                disnea=diccionario_form['disnea'],
                sat_oxi=diccionario_form['saturacion'])
            objPrediction.string_value=diccionario_respuestas['string_value']
            objPrediction.value =diccionario_respuestas['value']
            objPrediction.save()
            for key in diccionario_form.keys():
                objVariable=Variable.objects.variableSegunNombreForm(key)
                objInput=Input.objects.create(
                    variable=objVariable,
                    value=diccionario_form[key],
                    patient_code=codigo_paciente,
                    )
                objPrediction.input_set.add(objInput)
            return objPrediction.id
    except Exception as e:
        print("Error al solicitar el generar predicción",e)
```

Fig. 107. Transacción generar_prediccion

Fuente: Elaboración propia

```python
def predict(
    rna,edad,sexo,hipertension,diabetes,fiebre,tos,malestar_general,cansancio,disnea,sat_oxi
    ):
    diccionario_respuestas={}
    rna_file = rna.model.path
    classifier = load_model(rna_file)
    datos=preparar_data(
      rna,edad,sexo,hipertension,diabetes,fiebre,tos,malestar_general,cansancio,disnea,sat_oxi
    )
    y_paciente=classifier.predict(datos)
    diccionario_respuestas['value']=np.round_(y_paciente,decimals=4)
    y_paciente = (y_paciente>0.6)
    if y_paciente[0][0]:
        diccionario_respuestas['string_value']='Riesgo Alto'
    else:
        diccionario_respuestas['string_value']='Riesgo Bajo'
    return diccionario_respuestas
```

Fig. 108. Función predict

Fuente: Elaboración propia

```
def preparar_data(
    rna,edad,sexo,hipertension,diabetes,fiebre,tos,malestar_general,cansancio,disnea,sat_oxi
    ):
    arreglo_paciente=np.array(
      [
          [edad,sexo,hipertension,diabetes,fiebre,tos,malestar_general,cansancio,disnea,sat_oxi]
      ],
      dtype=object
    )
    scalar_file=rna.extra_file.standard_scaler.path
    sc_X=load(scalar_file)
    arreglo_paciente = sc_X.transform(arreglo_paciente)
    return arreglo_paciente
```

Fig. 109. Función preparar_data

Fuente: Elaboración propia

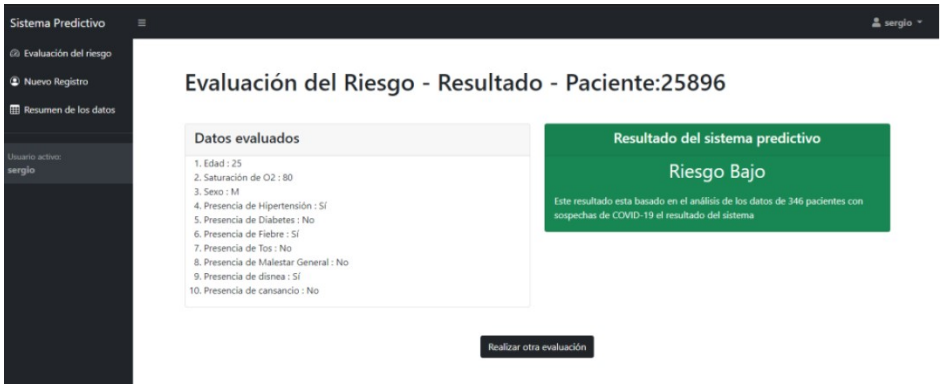


Fig. 110. Interfaz Resultado evaluación del riesgo

Fuente: Elaboración propia

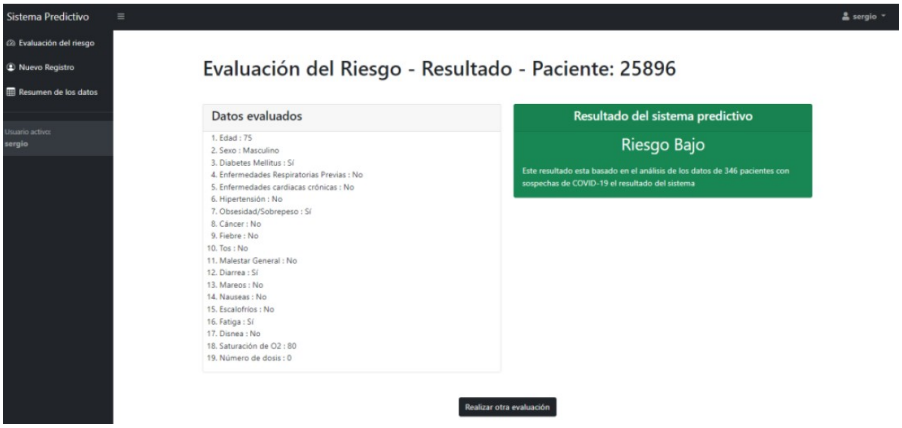


Fig. 111. Interfaz Resultado evaluación de riesgo - Modo detallado

Fuente: Elaboración propia

```python
class ResultadoEvaluacionView(LoginRequiredMixin,DetailView):
    template_name = "riskPredictor/resultado_evaluacion.html"
    login_url=reverse_lazy('accounts_app:user-login')
    model = Prediction
    context_object_name="prediccion"
    def get_context_data(self,**kwargs):
        context = super(ResultadoEvaluacionView,self).get_context_data(**kwargs)
        context['Inputs'] = Input.objects.filter(prediction=self.get_object())
        context['codigo_paciente'] = Input.objects.filter(
                                        prediction=self.get_object()
                                    ).last().patient_code
        return context
```

Fig. 112. Vista ResultadoEvaluacionView

Fuente: Elaboración propia

```python
class Record(TimeStampedModel):

    user = models.ForeignKey(
        User, on_delete=models.CASCADE,verbose_name='Usuario')
    is_used = models.BooleanField(
        '¿Ya se utilizó?',default=False
    )
    class Meta:
        """Meta definition for Record."""
        verbose_name = 'Registro de datos'
        verbose_name_plural = 'Registros de datos'

    def __str__(self):
        """Unicode representation of Record."""
        return f'Registro de datos: {self.id}'
```

Fig. 113. Clase Record

Fuente: Elaboración propia

```python
def registrar_datos(self,**params):
    try:
        with transaction.atomic():
            diccionario_form=without_keys(params['inputs_form'],'codigo_paciente')
            codigo_paciente = params['inputs_form']['codigo_paciente']
            genero= 1 if diccionario_form['genero']=='M' else 0
            objRecord=Record.objects.create(
                    user=params['user']
                )
            for key in diccionario_form.keys():
                objVariable=Variable.objects.variableSegunNombreForm(key)
                objInput=Input.objects.create(
                    variable=objVariable,
                    value=diccionario_form[key],
                    patient_code=codigo_paciente,
                    )
                objRecord.input_set.add(objInput)
            return objRecord.id
```

Fig. 114. Transacción registrar_datos

Fuente: Elaboración propia

```python
class NuevoRegistroView(LoginRequiredMixin,FormView):
    template_name = "riskPredictor/nuevo_registro.html"
    login_url=reverse_lazy('accounts_app:user-login')
    form_class = RegistroNuevoForm

    def __init__(self,**kwargs):
        super(NuevoRegistroView,self).__init__()
        self.rna_simple = ANN.objects.obtener_rna_simple()

    def get_context_data(self,**kwargs):

        arquitectura = self.rna_simple.architecture
        context = super(NuevoRegistroView,self).get_context_data(**kwargs)
        context['categorias'] = Category.objects.all()
        context['variables'] = arquitectura.variable_set.filter(is_active=True)
        context['selects'] = SelectControl.objects.all()
        context['exactitud'] = self.rna_simple.accuracy
        context['auc'] = self.rna_simple.auc
        return context

    def form_valid(self,form):
        pk_registro=registrar_datos(
          self,inputs_form=form.cleaned_data,user=self.request.user,ann=self.rna_simple)
        success_url = reverse_lazy('riskPredictor_app:resultado-registro',kwargs={'pk':pk_registro})
        return HttpResponseRedirect(success_url)
```

Fig. 115. Vista NuevoRegistroView

Fuente: Elaboración propia

```python
class NuevoRegistroDetalladoView(NuevoRegistroView):
    form_class = RegistroNuevoDetalladoForm

    def __init__(self,**kwargs):
        super(NuevoRegistroDetalladoView,self).__init__()
        self.rna_detallada = ANN.objects.obtener_rna_detallada()

    def get_context_data(self,**kwargs):
        arquitectura = self.rna_detallada.architecture
        context = super(NuevoRegistroView,self).get_context_data(**kwargs)
        context['categorias'] = Category.objects.all()
        context['variables'] = arquitectura.variable_set.filter(is_active=True)
        context['selects'] = SelectControl.objects.all()
        context['detallado'] = True
        context['exactitud'] = self.rna_detallada.accuracy
        context['auc'] = self.rna_detallada.auc
        return context

    def form_valid(self,form):
        pk_registro=registrar_datos(
          self,inputs_form=form.cleaned_data,user=self.request.user,ann= self.rna_detallada)
        success_url = reverse_lazy('riskPredictor_app:resultado-registro',kwargs={'pk':pk_registro})
        return HttpResponseRedirect(success_url)
```

Fig. 116. Vista NuevoRegistroDetalladoView

Fuente: Elaboración propia

Fig. 117. Interfaz de resultado del registro de datos – Modo simple

Fuente: Elaboración propia

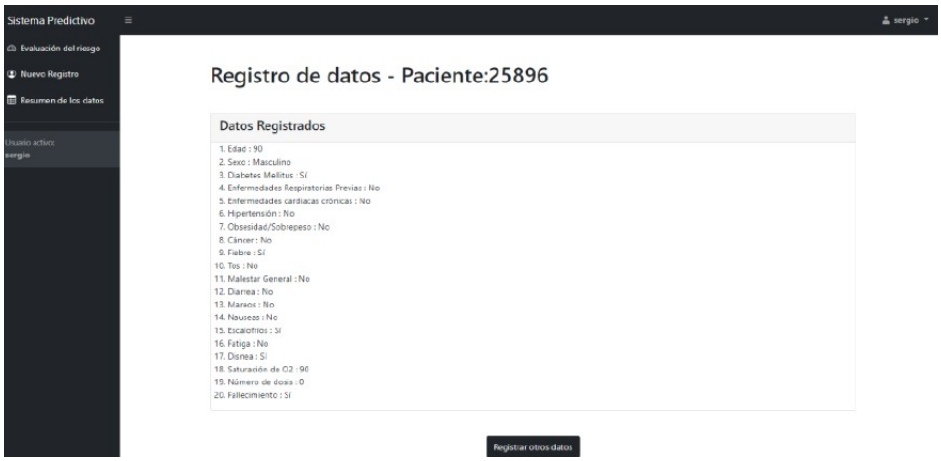


Fig. 118. Interfaz de resultado de registro de datos - Modo detallado

Fuente: Elaboración propia

```python
class ResultadoRegistroView(LoginRequiredMixin,DetailView):
    template_name = "riskPredictor/resultado_registro.html"
    login_url=reverse_lazy('accounts_app:user-login')
    model = Record
    context_object_name="registro"
    def get_context_data(self,**kwargs):
        context = super(ResultadoRegistroView,self).get_context_data(**kwargs)
        context['Inputs'] = Input.objects.filter(record=self.get_object())
        context['codigo_paciente'] = Input.objects.filter(
                                        record=self.get_object()
                                    ).last().patient_code
        return context
```

Fig. 119. Vista ResultadoRegistroView

Fuente: Elaboración propia

```python
    def get_html(self):
        code = f"""
        <p class="fs-2 text-center">{self.options["title"]}</p><canvas id="{self.chart_id}">
        </canvas>"""
        return code

    def get_js(self):
        code = f"""
            var chartElement = document.getElementById('{self.chart_id}').getContext('2d');
            var {self.chart_id}Chart = new Chart(chartElement, {self.get_elements()});
        """
        return code

    def get_presentation(self):
        code = {
            'html':self.get_html(),
            'js': self.get_js(),
        }
        return code
```

Fig. 120. Funciones de obtención de las gráficas

Fuente: Elaboración propia

```python
class DashboardView(LoginRequiredMixin,TemplateView):
    template_name = "staticPlot/dashboard.html"

    def __init__(self, **kwargs):
        super(DashboardView, self).__init__(**kwargs)
        self.dashboard= DashboardData.objects.all().last()
        # Ajustando el numero de vacunados
        lista_vacunacion = Vaccination.objects.all()
        suma_vacunados=0
        for vacunacion in lista_vacunacion:
            suma_vacunados+=vacunacion.patients
        # Pacientes totales
        objDashboard = DashboardData.objects.all().last()
        numero_total_pacientes = objDashboard.patients
        if suma_vacunados != numero_total_pacientes:
            sin_dosis = lista_vacunacion[0]
            sin_dosis.patients+=numero_total_pacientes-suma_vacunados
            sin_dosis.save()

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['dashboard'] = self.dashboard
        context['comorbilidades'] = Comorbidities.objects.comorbilidadesSegunDashboard(
            self.dashboard)
        context['sintomas'] = Symptoms.objects.sintomasSegunDashBoard(
            self.dashboard
        )
        #Gráfico del sexo
        sexo_doughnut = graficar_dona_lista(
            ['Femenino','Masculino'],
            [self.dashboard.female,self.dashboard.male],
            nombre_grafico="Sexo"
            )
        context['chart_sexo']=sexo_doughnut.get_presentation()
        #Gráfico de vacunacion
        lista_vacunaciones = Vaccination.objects.all()
        vacunacion_doughnut = graficar_dona_lista(
            nombre_campos=[x.description for x in lista_vacunaciones],
            values=[x.patients for x in lista_vacunaciones],
            nombre_grafico="vacunacion"
            )
        context['chart_vacunacion']=vacunacion_doughnut.get_presentation()
        #Gráfico de barras
        lista_grupos_etarios = AgeGroup.objects.all()
        grupo_etarios_bar = graficar_barra_lista(
            nombre_campos=[x.description for x in lista_grupos_etarios],
            values=[x.patients for x in lista_grupos_etarios],
            nombre_grafico="grupos_etareos",
            nombre_tooltip="Nro pacientes: "
            )
        context['chart_grupo_etareo']=grupo_etarios_bar.get_presentation()
        return context
```

Fig. 121. Vista DashboardView

Fuente: Elaboración propia

ANEXO N° 14. CONFIGURACIÓN DE LOS SERVIDORES

Fig. 122. Comparación de precios entre máquinas virtuales

Fuente: Extraído de [54]

```
upstream sistema_predictivo_app {
  server unix:/webapps/sistemapredictivo-env/run/gunicorn.sock fail_timeout=0;
}

server {
    server_name sistemapredictivo.akemid.com www.sistemapredictivo.akemid.com;

    access_log /webapps/sistemapredictivo-env/logs/nginx-access.log;
    error_log /webapps/sistemapredictivo-env/logs/nginx-error.log;

    location /static/ {
        alias   /webapps/sistemapredictivo-env/thesis-project/PredictiveSystem/staticfiles/;
    }

    location /media/ {
        alias   /webapps/sistemapredictivo-env/thesis-project/PredictiveSystem/media/;
    }

    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;

        if (!-f $request_filename) {
            proxy_pass http://sistema_predictivo_app;
            break;
        }
  }


}
```

Fig. 123. Archivo configuración NGINX

Fuente: Elaboración propia

```
(sistemapredictivo-env) root@akemidev:/webapps/sistemapredictivo-env# psql -U postgres
Password for user postgres:
psql (12.9 (Ubuntu 12.9-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# CREATE DATABASE db_predictive_system;
```

Fig. 124. Creación de la base de datos

Fuente: Elaboración propia

```
#!/bin/bash

NAME="sistema-predictivo"                                   # Name of the application
DJANGODIR=/webapps/sistemapredictivo-env/thesis-project/PredictiveSystem             # Django project directory
SOCKFILE=/webapps/sistemapredictivo-env/run/gunicorn.sock  # we will communicte using this unix socket
USER=root                                                  # the user to run as
GROUP=root                                                 # the group to run as
NUM_WORKERS=3                                              # how many worker processes should Gunicorn spawn
DJANGO_SETTINGS_MODULE=PredictiveSystem.settings.prod             # which settings file should Django use
DJANGO_WSGI_MODULE=PredictiveSystem.wsgi                     # WSGI module name

echo "Starting $NAME as `whoami`"

# Activate the virtual environment
cd $DJANGODIR
source ../bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

# Create the run directory if it doesn't exist
RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

# Start your Django Unicorn
# Programs meant to be run under supervisor should not daemonize themselves (do not use --daemon)
exec ../bin/gunicorn ${DJANGO_WSGI_MODULE}:application \
  --name $NAME \
  --workers $NUM_WORKERS \
  --user=$USER --group=$GROUP \
  --bind=unix:$SOCKFILE \
  --log-level=debug \
  --log-file=-
```

Fig. 125. Configuración de Gunicorn

Fuente: Elaboración propia

```
[program:sistema-predictivo]
command = /webapps/sistemapredictivo-env/bin/gunicorn_start                    ; Command to start app
user = root                                                                    ; User to run as
stdout_logfile = /webapps/sistemapredictivo-env/logs/gunicorn_supervisor.log   ; Where to write log messages
redirect_stderr = true                                                         ; Save stderr in the same log
environment=LANG=en_US.UTF-8,LC_ALL=en_US.UTF-8                                ; Set UTF-8 as default encoding
~
~
~
~
```

Fig. 126.Configuración de supervisor

Fuente: Elaboración propia

ANEXO N° 15. PRUEBAS DE CAJA NEGRA

HU35-Registro de usuario
CP1: Validación del campo usuario
Descripción:
En la interfaz de registro de usuario se solicitarán los datos de correo electrónico, nombre de usuario, contraseña y la repetición de la contraseña. Así, se creará un usuario del sistema el cual deberá ser activado en una posterior validación de pertenencia de la institución.
ENTRADA
Usuario invalido: <Dato vacío> Contraseña valida: franzo1821 / Contraseña repetida: franzo1821 / Email valido: franzojza2109@outlook.com
Pasos:
1. Ingresar a la página web 2. Seleccionar el botón de registrarse 3. Ingresar los datos 4. Dar clic en registrarse
Resultado esperado:
Mensaje flotante sobre la casilla del dato nombre de usuario: “Completa este campo”
Resultado obtenido:

Fig. 127. Interfaz en el CP1

Fuente: Elaboración propia

HU35-Registro de usuario
CP2: Validación del campo contraseña
Descripción:
En la interfaz de registro de usuario se solicitarán los datos de correo electrónico, nombre de usuario, contraseña y la repetición de la contraseña. Así, se creará un usuario del sistema el cual deberá ser activado en una posterior validación de pertenencia de la institución.
ENTRADA
Usuario: franzo123 Contraseña valida: franzo1821 / Contraseña repetida: franzo181 / Email valido: franzojza2109@outlook.com
Pasos:
1. Ingresar a la página web 2. Seleccionar el botón de registrarse

3. Ingresar los datos 4. Dar clic en registrarse
Resultado esperado:
Mensaje: “ERROR, las contraseñas no coinciden”
Resultado obtenido


Fig. 128. Interfaz en el CP2

Fuente: Elaboración propia

HU35-Registro de usuario
CP3: Validación del campo email
Descripción:
En la interfaz de registro de usuario se solicitarán los datos de correo electrónico, nombre de usuario, contraseña y la repetición de la contraseña. Así, se creará un usuario del

sistema el cual deberá ser activado en una posterior validación de pertenencia de la institución.
ENTRADA
Usuario: franzo123 Contraseña valida: franzo1821 / Contraseña repetida: franzo1821 / Email invalido: franzojza2109outlook.com
Pasos:
1. Ingresar a la página web 2. Seleccionar el botón de registrarse 3. Ingresar los datos 4. Dar clic en registrarse
Resultado esperado:
Mensaje flotante sobre la casilla del dato email: ‘Incluye un signo ‘@’ en la dirección de correo electrónico. La dirección “franzojza209outlook.com” no incluye el signo “@”.
Resultado obtenido:

Fig. 129. Interfaz en el CP3

Fuente: Elaboración propia

HU36 - Inicio de sesión

CP4: Validación del campo correo electrónico
Descripción:
En la interfaz de inicio de sesión se solicitarán los datos de correo electrónico y contraseña. De ser válido los resultados y el usuario esta activo, se identificará si el usuario es administrador o un médico de la institución.
ENTRADA
Correo electrónico inválido: <vacío> Contraseña válida: franzo1821
Pasos:
1. ingresar a la página web 2. Ingresar los datos 3.dar clic en el botón de Ingresar
Resultado esperado:
Mensaje flotante sobre la casilla del dato de correo electrónico:” Complete este campo”
Resultado obtenido:

Fig. 130. Interfaz en el CP4

Fuente: Elaboración propia

HU36 - Inicio de sesión
CP5: Validación del campo contraseña
Descripción:
En la interfaz de inicio de sesión se solicitarán los datos de correo electrónico y contraseña. De ser válido los resultados y el usuario esta activo, se identificará si el usuario es administrador o un médico de la institución.
ENTRADA
Contraseña inválida:<vacía> email válido: sergiomondragondev@gmail.com
Pasos:
1. ingresar a la página web 2. Ingresar los datos 3.dar clic en el botón de Ingresar
Resultado esperado:
Mensaje flotante sobre la casilla del dato de contraseña:” Complete este campo”
Resultado obtenido:

Fig. 131. Interfaz en el CP5

Fuente: Elaboración propia

HU36 - Inicio de sesión
CP6: Credenciales inválidas
Descripción:
En la interfaz de inicio de sesión se solicitarán los datos de correo electrónico y contraseña. De ser válido los resultados y el usuario esta activo, se identificará si el usuario es administrador o un médico de la institución.
ENTRADA
Usuario inválido: sergiomondragondev@gmail.com Contraseña inválida: 123567
Pasos:
1. Ingresar a la página web 2. Ingresar los datos

3.Dar clic en el botón de Ingresar
Resultado esperado:
Mensaje: “Los datos no son correctos, vuelva intentar”
Resultado obtenido:

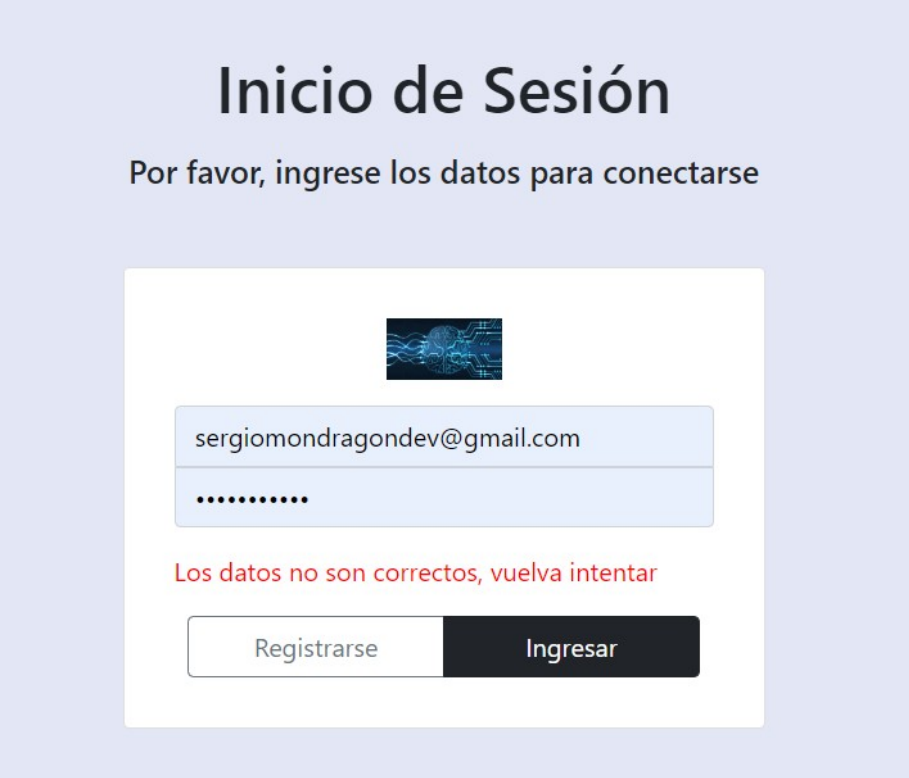


Fig. 132. Interfaz en el CP6

Fuente: Elaboración propia

HU36 - Inicio de sesión
CP7: Intento de logueo usuario no activo.
Descripción:
En la interfaz de inicio de sesión se solicitarán los datos de correo electrónico y contraseña. De ser válido los resultados y el usuario esta activo, se identificará si el usuario es administrador o un médico de la institución.
ENTRADA
Correo electrónico válido: franzo@gmail.com Contraseña válida: franzo1821
Pasos:

1. ingresar a la página web 2. Ingresar los datos 3.dar clic en el botón de Ingresar
Resultado esperado:
Mensaje: “El usuario no está activo, comuníquese con la institución para la habilitación.”
Resultado obtenido:

Fig. 133. Interfaz en el CP7

Fuente: Elaboración propia

HU39 – Evaluar Riesgo
CP8: Evaluación sin desaturación de oxígeno.
Descripción:
En la interfaz de evaluar riesgo se tienen los campos que se definieron para la evaluación

del riesgo de mortalidad del COVID-19 en los pacientes infectados. Así, al darle click en el botón calcular riesgo la red neuronal artificial procesará los datos y mostrará el nivel de riesgo.
ENTRADA
Código paciente: 2234 Edad: 27 Género: Femenino Saturación: <campo vacío> Hipertensión: Check box activado Diabetes mellitus: Check box desactivado Tos: Check box desactivado Malestar general: Check box desactivado Fiebre Check box activado Disnea: Check box activado Fatiga: Check box activado
Pasos:
1. Ingresar a la página web 2. Loguearse con credenciales válidas 3. Seleccionar la opción del menú lateral: ‘Evaluación del Riesgo’ 4. Ingresar los datos 5. Dar clic en el botón de "Calcular el riesgo”
Resultado esperado:
Mensaje flotante sobre la casilla del dato de contraseña:” Complete este campo”
Resultado obtenido:

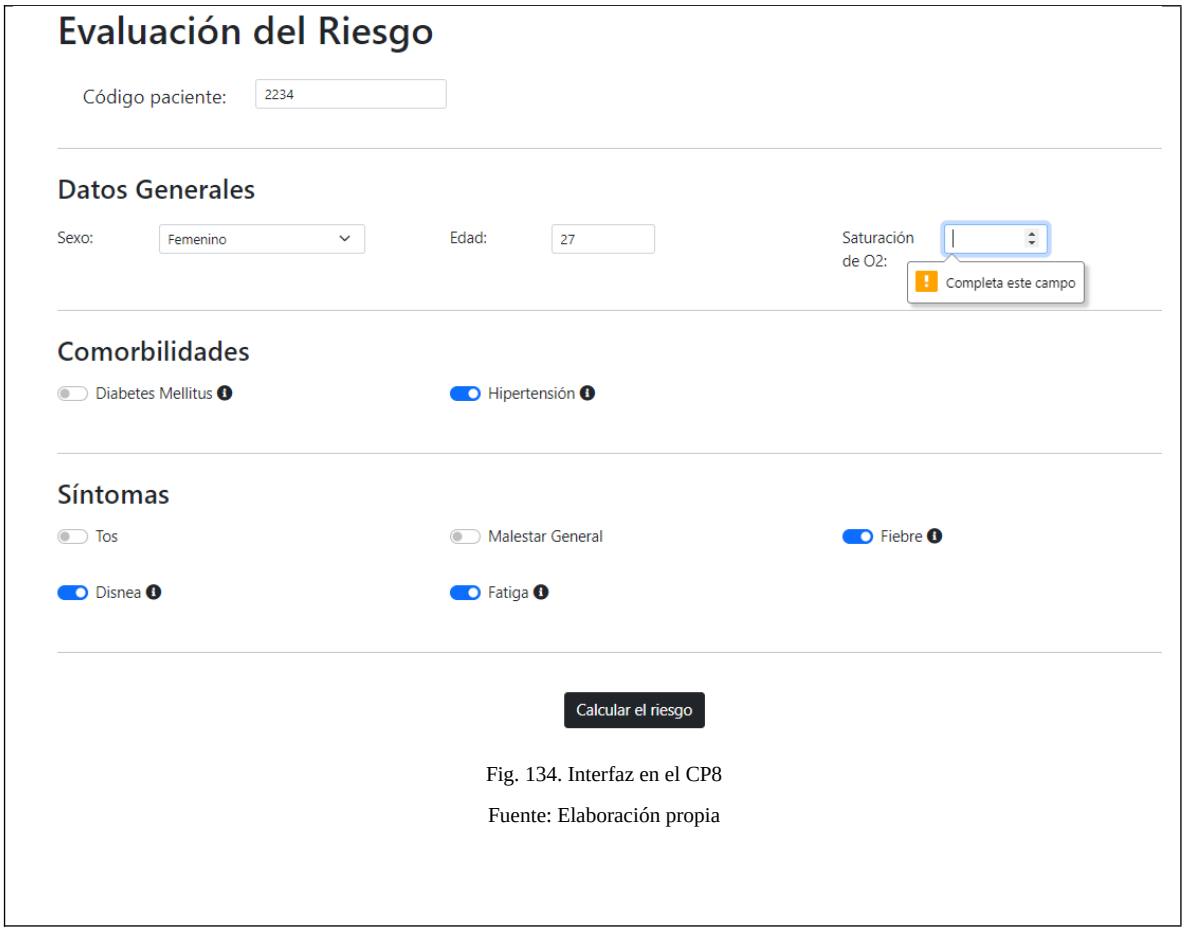


Fig. 134. Interfaz en el CP8

Fuente: Elaboración propia

HU39 – Evaluar Riesgo
CP9: Evaluación del riesgo con edad negativa.
Descripción:
En la interfaz de evaluar riesgo se tienen los campos que se definieron para la evaluación del riesgo de mortalidad del COVID-19 en los pacientes infectados. Así, al darle click en el botón calcular riesgo la red neuronal artificial procesará los datos y mostrará el nivel de riesgo.
ENTRADA
Código paciente: 2234 Edad: -5 Género: Femenino Saturación: 90 Hipertensión: Check box activado Diabetes mellitus: Check box desactivado

Tos: Check box desactivado Malestar general: Check box desactivado Fiebre Check box activado Disnea: Check box activado Fatiga: Check box activado
Pasos:
1. Ingresar a la página web 2. Loguearse con credenciales válidas 3. Seleccionar la opción del menú lateral: ‘evaluación del Riesgo’ 4. Ingresar los datos 5. Dar clic en el botón de "Calcular el riesgo”
Resultado esperado:
Mensaje flotante sobre la casilla del dato de contraseña:” El valor debe ser mayor o igual que 0”
Resultado obtenido:

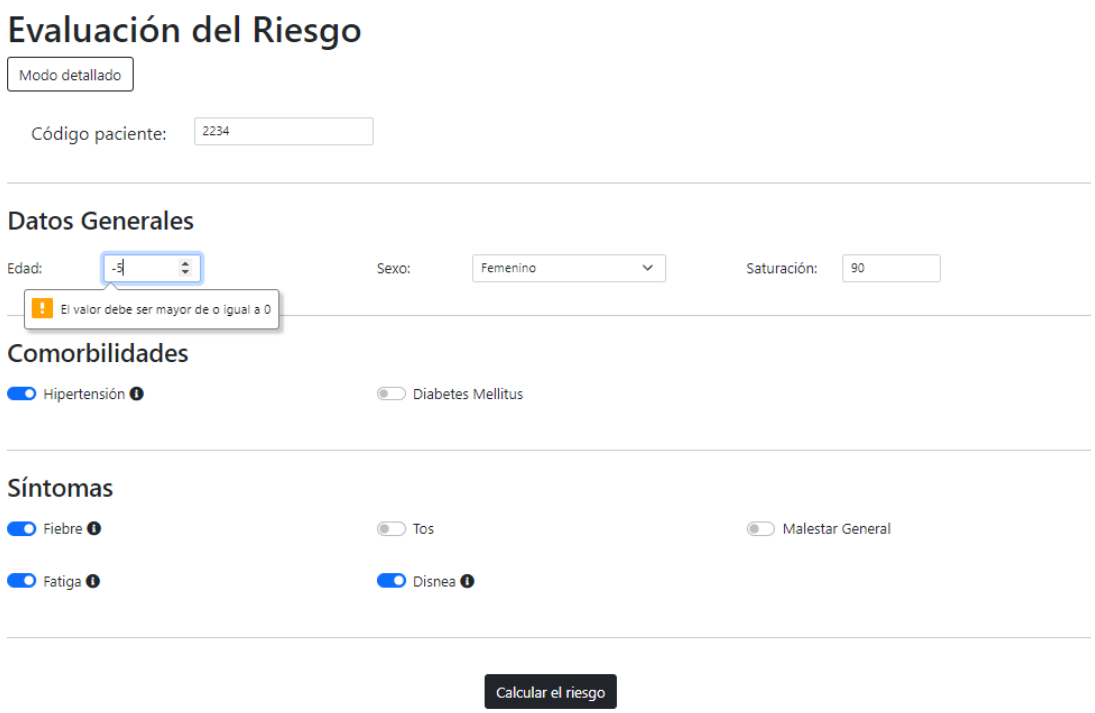


Fig. 135. Interfaz en el CP9

Fuente: Elaboración propia

HU41 – Registro de nuevos datos para la RNA
CP10: Registro de datos sin código de paciente
Descripción:
En la interfaz de registro de datos se tienen las variables dependientes e independientes relacionadas con el riesgo de mortalidad del COVID-19 en los pacientes infectados. Así, al darle clic en el botón registrar datos, se guardarán los registros para el posterior reentrenamiento de la RNA.
ENTRADA
Código paciente: <vacío> Edad: 25 Género: Masculino Saturación: 90 Hipertensión: Check box activado Diabetes mellitus: Check box desactivado Tos: Check box activado Malestar general: Check box desactivado Fiebre Check box desactivado Disnea: Check box desactivado Fatiga: Check box activado Fallecimiento: Check box activado
Pasos:
1. Ingresar a la página web

2. Loguearse con credenciales válidas 3. Seleccionar la opción del menú lateral: ‘Nuevo registro’ 4. Ingresar los datos 5. Dar clic en el botón de "Registrar datos”
Resultado esperado:
Mensaje flotante sobre la casilla del código del paciente:” Completa este campo”
Resultado obtenido:

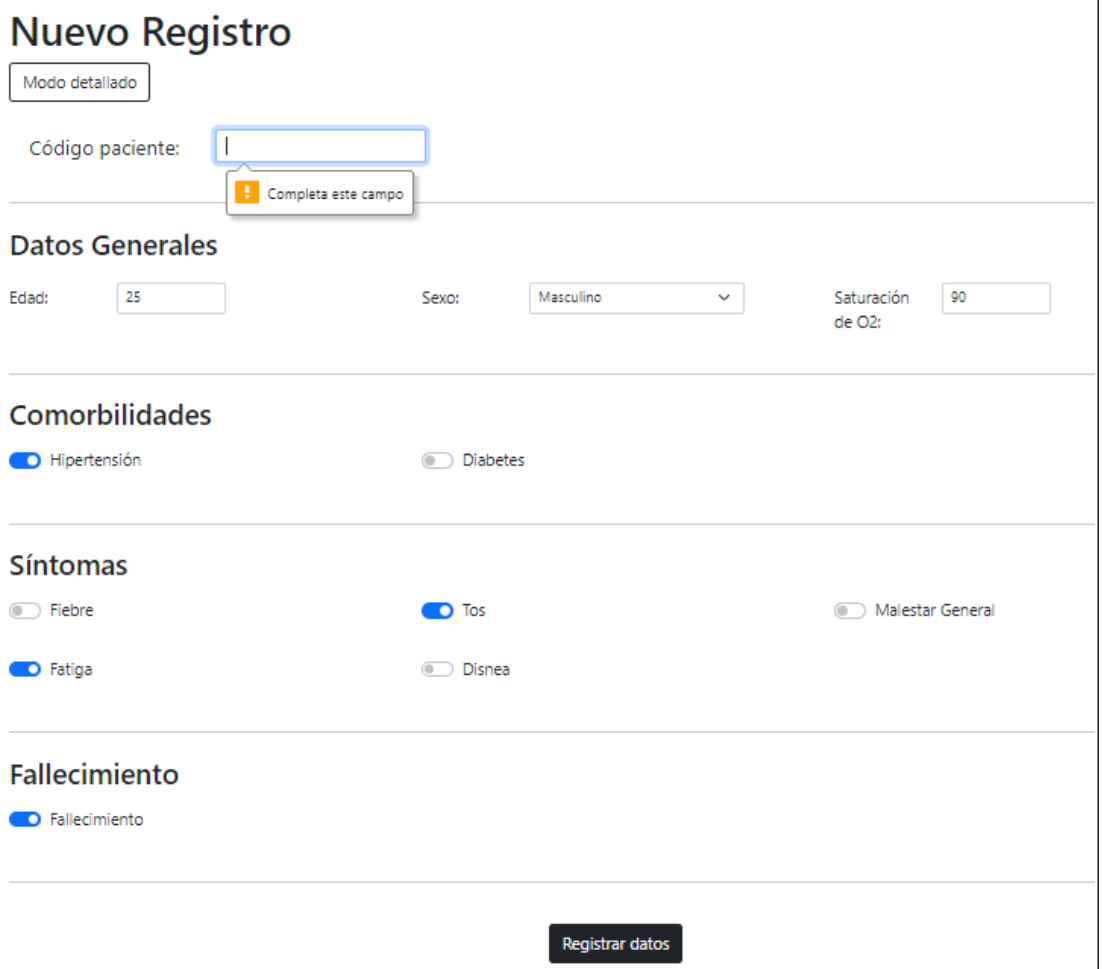


Fig. 136. Interfaz en el CP10

Fuente: Elaboración propia

HU41 – Registro de nuevos datos para la RNA

CP11: Registro de datos con saturación alta

Descripción:

En la interfaz de registro de datos se tienen las variables dependientes e independientes relacionadas con el riesgo de mortalidad del COVID-19 en los pacientes infectados. Así, al darle clic en el botón registrar datos, se guardarán los registros para el posterior reentrenamiento de la RNA.

ENTRADA

Código paciente: 25896
Edad: 25
Género: Masculino
Saturación: 150
Hipertensión: Check box activado
Diabetes mellitus: Check box desactivado
Tos: Check box activado
Malestar general: Check box desactivado
Fiebre Check box desactivado
Disnea: Check box desactivado
Fatiga: Check box activado
Fallecimiento: Check box activado

Pasos:

1. Ingresar a la página web
2. Loguearse con credenciales válidas
3. Seleccionar la opción del menú lateral: ‘Nuevo registro’
4. Ingresar los datos
5. Dar clic en el botón de "Registrar datos”

Resultado esperado:

Mensaje flotante sobre la casilla de la Saturación de Oxígeno:” El valor debe ser inferior o igual a 100”

Resultado obtenido:

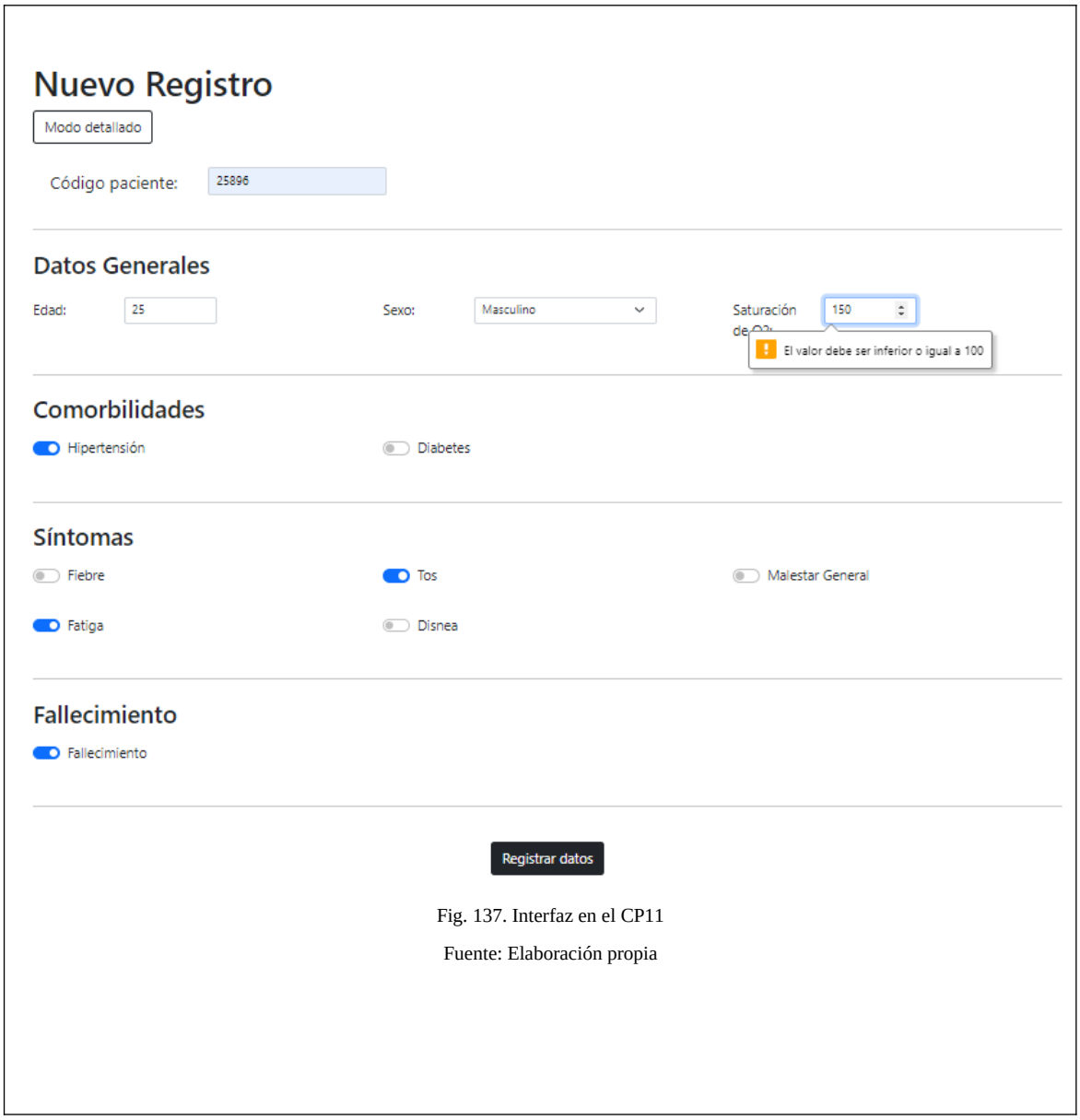


Fig. 137. Interfaz en el CP11

Fuente: Elaboración propia

ANEXO N° 16. PRUEBAS DE CAJA BLANCA

_HU35- Registro de usuario: Función clean del formulario registro de usuario.

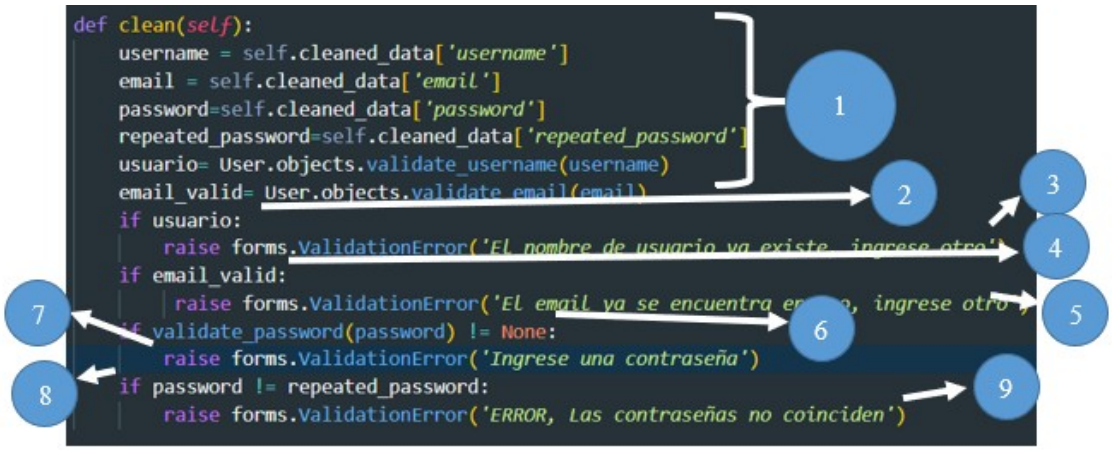


Fig. 138. Código evaluado HU35

Fuente: Elaboración propia

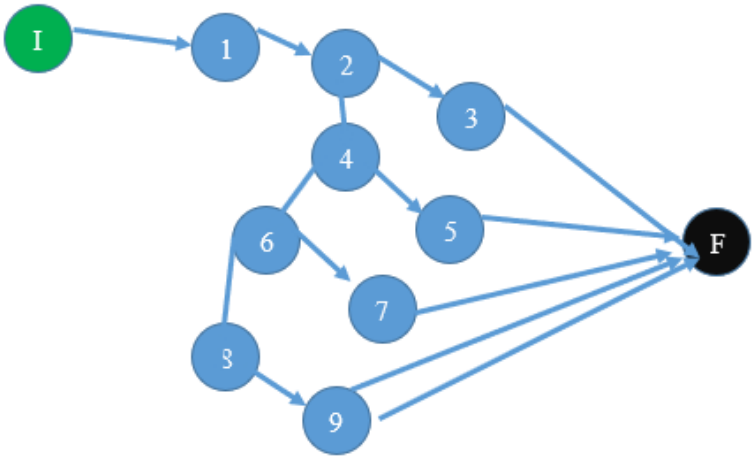


Fig. 139. Grafo de flujo I

Fuente: Elaboración propia

$$V(G) = 14 - 11 + 2 = 5$$

Caminos	Entradas		Salida
I-1-2-3-F	Username	Sergio	Muestra mensaje de error:" El nombre de usuario ya existe, ingrese otro"
	Email	sergiomondragondev@gmail.com	
	password	456478Hy/*	
	repeated_password	456478Hy/*	
I-1-2-4-5-F	Username	franzo	Muestra el mensaje de error: "El email ya se encuentra en uso, ingrese otro"
	Email	franzo123@gmail.com	
	password	456478Hy/*	
	repeated_password	456478Hy/*	

I-1-2-4-6-7-F	**Username**	**alex**	**Muestra el mensaje de error: “Ingrese una contraseña”**
	Email	sergio.a.ms.usat**@gmail.com**	
	password		
	repeated_password	**456478Hy/***	
I-1-2-4-6-8-F	**Username**	**alex**	**Ejecución correcta, usuario registrado**
	Email	sergio.a.ms.usat**@gmail.com**	
	password	**456478Hy/***	
	repeated_password	**456478Hy/***	
I-1-2-4-6-8-9-F	**Username**	**alex**	**Muestra el mensaje de error: “ERROR, Las contraseñas no**
	Email	sergio.a.ms.usat**@gmail.com**	
	password	**456479Hy/***	
	repeated_password	**456478Hy/***	

HU36- Inicio de sesión: Función clean del formulario inicio de sesión.

```
def clean(self):
    cleaned_data=super(LoginForm,self).clean()
    email=self.cleaned_data['email']
    password=self.cleaned_data['password']

    if not authenticate(email=email,password=password):
        try:
            if User.objects.get(email=email).is_active == False :
                raise forms.ValidationError(
                    'El usuario no está activo, comuníquese con la institución para la habilitación.'
                    )
        except Exception as e:
            raise forms.ValidationError('Los datos no son correctos, vuelva intentar')
```

Fig. 140. Código evaluado HU36

Fuente: Elaboración propia

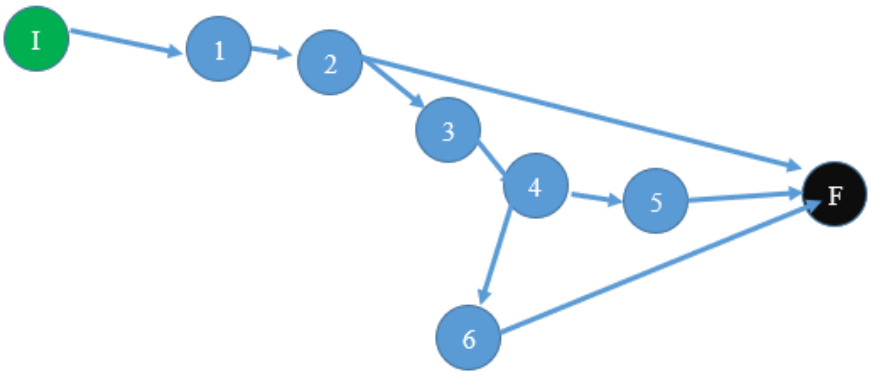


Fig. 141. Grafo de flujo II

Fuente: Elaboración propia

V(G) = Aristas – nodos + 2 =9-8+2=3

Caminos	Entradas		Salida
I-1-2-F	**Email**	sergio.a.ms.usat**@gmail.com**	**Ejecución correcta, usuario logueado**
	password	**456478Hy/***	
I-1-2-3-4-5-F	**Email**	sergiomondragondev**@gmail.com**	**Muestra el mensaje de error: "El usuario no está activo, comuníquese con la institución para la habilitación."**
	password	**456478Hy/***	
I-1-2-3-4-6-F	**Email**	sergio.a.ms.usat**@gmail.com**	**Muestra el mensaje de error: "Los datos no son correctos, vuelva intentar"**
	password	**456477Hy/***	

HU39- Evaluar Riesgo: Función de predicción del riesgo

Fig. 142. Código evaluado de la HU39

Fuente: Elaboración propia

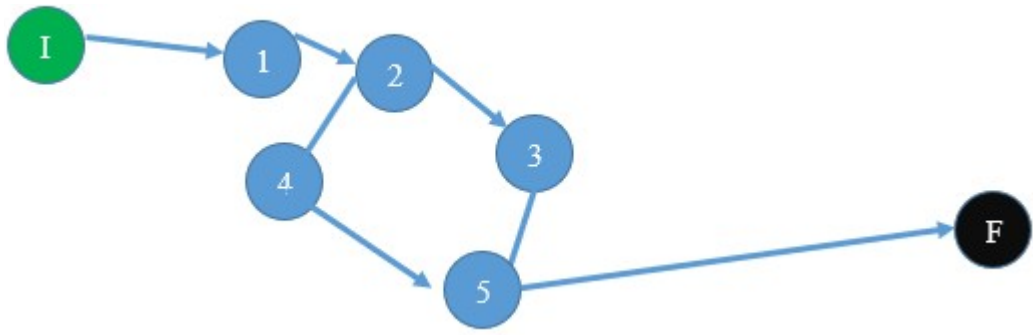


Fig. 143. Grafo de flujo III

Fuente: Elaboración propia

V(G) = Aristas – nodos + 2 =7-7+2=2

Caminos	Entradas		Salida
I-1-2-3-5-F	edad	25	'Riesgo alto'
	sexo	Masculino	
	hipertension	no	
	Diabetes	no	
	Fiebre	sí	
	Tos	sí	
	Malestar_general	sí	
	cansancio	no	
	Disnea	no	
	Sat_oxi	95	
I-1-2-4-5-F	edad	57	'Riesgo alto'
	sexo	Femenino	
	hipertension	no	
	Diabetes	no	
	Fiebre	no	
	Tos	no	

	Malestar_general	**sí**	
	cansancio	**no**	
	Disnea	**no**	
	Sat_oxi	**79**	

HU41- Registro de nuevos datos para la RNA: Función de preparación de los datos registrados para la actualización de la RNA:

Precondición: Existen registros con variables dependientes e independientes

Fig. 144. Código evaluado de la HU41

Fuente: Elaboración propia

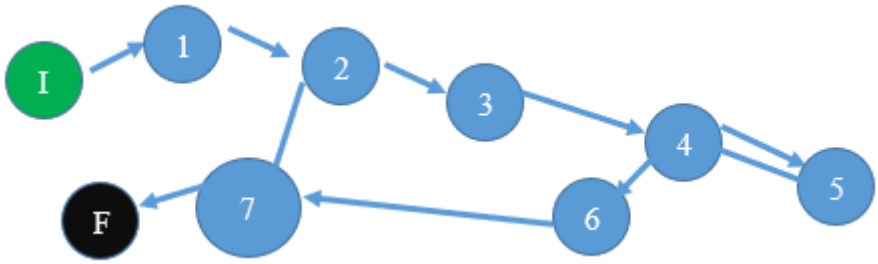


Fig. 145. Grafo de flujo IV

Fuente: Elaboración propia

V(G) = Aristas – nodos + 2 =10-9+2=3

Caminos	Entradas		Salida
I-1-2-3-4-6-7-F	**Listado de registros**	**[<QuerySet [<Record: Registro de datos: 18>]>]**	**X_train** = array ([[-2.01063711, -1.38228589, 3.46987031, -0.47977982, -1.0633527, 0.59388307, -1.68383316, 2.56542304, -1.60699022, 8.04546674]]) **y_train =**array ([1] **Lista_registros_no_entrenados**=<QuerySet [<Record: Registro de datos: 18>]>
I-1-2-7-F	**Listado de registros**	**<QuerySet []>**	**X_train**= array ([], dtype=object **y_train**= array ([], dtype=object **Lista_registros_no_entrenados**=<QuerySet []>
I-1-2-3-4-5-4-6-7-F	**Listado de registros**	**[<Record: Registro de datos: 18>, <Record: Registro de datos: 17>, <Record: Registro de datos: 11>]**	**X_train**=array ([[-2.01063711, -1.38228589, 3.46987031, -0.47977982, -1.0633527, 0.59388307, -1.68383316, 2.56542304, -1.60699022, 8.04546674], [-2.01063711, -1.38228589, -0.28819521, -0.47977982, -1.0633527, -1.68383316, -1.68383316, -0.38979926, -1.60699022, 1.57056573], [-2.18937655, -1.38228589, -0.28819521, -0.47977982, -1.0633527, -1.68383316, -1.68383316, -0.38979926, -1.60699022, 1.57056573]]) **y_train**=array ([1, 0, 0]) **Lista_registros_no_entrenados** = <QuerySet [<Record: Registro de datos: 18>, <Record: Registro de datos: 17>, <Record: Registro de datos: 11>]>)

ANEXO N° 17. ARQUITECTURA DEL SISTEMA INTELIGENTE WEB

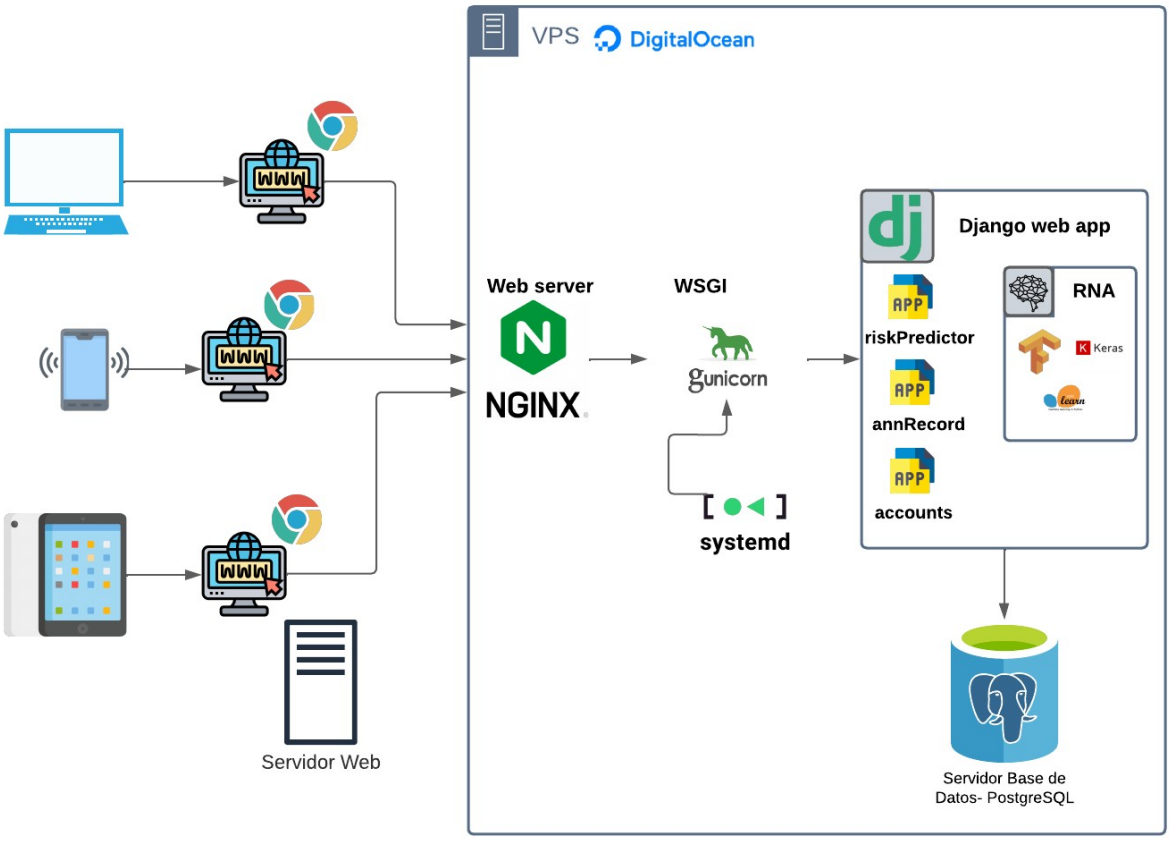


Fig. 146. Arquitectura del sistema inteligente web

Fuente: Elaboración propia