UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN

Trabajo Monográfico Para optar al título de

Ingeniero en Computación

Tema

Implementación de modelo de machine learning para el diagnóstico del cáncer de mama

Autor (es)

Br. Christhofer Gutiérrez Acevedo

Br. Sara Iris Bobadilla Moreno

Tutor

M.Sc. Narciso Javier Aguilera Centeno

Diciembre 2023

Managua, Nicaragua

Agradecimientos

En primer lugar, deseamos expresar nuestro sincero y profundo agradecimiento a Dios, quien ha sido nuestra fuente de vida, inspiración y fortaleza a lo largo de este viaje académico y que nos ha brindado la sabiduría y la perseverancia para llevar a cabo nuestra tesis de grado.

También queremos agradecer a todas las personas e instituciones que han contribuido de manera fundamental al éxito de nuestra investigación.

En primer lugar, agradecemos a la Universidad Nacional de Ingeniería por habernos brindado el privilegio de cursar una carrera que consideramos de alto nivel y sumamente apasionante.

Agradecemos al Centro de Mujeres IXCHEN en Managua, así como a todo su personal, por habernos dado la oportunidad de llevar a cabo este proyecto monográfico en beneficio de la salud de las mujeres nicaragüenses.

No podemos dejar de mencionar y agradecer la invaluable ayuda proporcionada por las especialistas radiólogas del centro médico durante la fase de pruebas; su conocimiento y experiencia resultaron fundamentales para la culminación exitosa de este proyecto.

Este logro no solo es nuestro, sino también de todos aquellos que nos han apoyado a lo largo de este camino. A todos ustedes, ¡nuestro más profundo agradecimiento!

Resumen

Durante los últimos 5 años, el cáncer de mama ha estado entre las 5 principales causas de muerte por tumoraciones malignas, en Nicaragua. Los decesos por esta enfermedad, en mujeres adultas, ocuparon la cuarta posición en 2022, 2019 y 2018; se ubicó segunda en 2021 y quinta en 2020 [1]. Como una contribución en la lucha contra este flagelo, en esta monografía, se explica el proceso seguido en la implantación de un proyecto de adaptación tecnológica de modelos de aprendizaje automático, como apoyo a la detección oportuna del cáncer de mama. El conjunto de datos utilizado se creó a partir de ecografías de mama previamente diagnosticadas por especialistas médicos y clasificadas de acuerdo con tales criterios [2].

Se partió de los resultados alcanzados en cuatro proyectos anteriores, ejecutados por investigadores de instituciones académicas y científicas en América Latina, Estados Unidos y Egipto [3] [4] [5] [6] [7] [8]. Se implementó una solución utilizando técnicas de Inteligencia Artificial que combinan modelos de segmentación y clasificación de imágenes basadas en redes neurales convolucionales.

La herramienta implementada en este trabajo fue puesta a prueba por personal médico especializado en radiología y en el análisis de imágenes de ultrasonido para el diagnóstico médico de cáncer de mama utilizando ecografías. La retroalimentación brindada en esa etapa sirvió para corregir, mejorar y poner a punto la implementación definitiva del nuevo modelo. Esto incluyó, ajustar apropiadamente el conjunto de datos utilizado por la máquina de aprendizaje automático que finalmente fue puesta en producción. Para facilitar el acceso al mismo, a través de Internet, el recurso desarrollado se instaló en una plataforma en la nube. Esto contribuirá a facilitar y aprovechar, con mayor eficiencia, la disponibilidad de esta herramienta por parte de personal médico y operadores de servicios de salud que lo necesiten.

Índice de contenido

Índice de Figuras

Índice de Tablas

Índice de Anexos

1. Introducción

En esta monografía se describe el trabajo realizado para optar al título de Ingeniero en Computación. Se expone la problemática existente en Nicaragua con respecto a las muertes por cáncer de mama, y cómo el desarrollo de hardware capaz de ejecutar algoritmos de aprendizaje automático ha permitido la detección oportuna de dicha enfermedad, en las últimas décadas.

Mediante el uso de un conjunto de herramientas (lenguajes de programación [10], librerías [11], capacidades de hardware, etc.), se realizó una adaptación tecnológica tomando como referencia varios trabajos investigativos sobre el uso de machine learning para diagnóstico de cáncer de mama. Estos se realizaron en universidades y/o centros de investigación de varios países [3] [4] [5] [6] [7] [8]. Para su selección, se usaron criterios como: similitudes culturales y étnicas de pacientes y profesionales involucrados; semejanza en los objetivos presentados en dichos estudios con los planteados en este proyecto; uso de herramientas open source; calidad en la elaboración; entre otros.

También se replicaron versiones de pruebas de los modelos de aprendizaje automático empleados en esos trabajos. A partir de ellos, se seleccionaron los criterios a tener en cuenta para implementar un modelo alternativo, a ser utilizado por el personal médico del Centro de Mujeres IXCHEN[1], como herramienta auxiliar en el diagnóstico de la patología. En la fase de prueba, los datos de los usuarios fueron suministrados al modelo a través de un formulario en un sitio web de Internet. Se recomienda esta opción para su puesta en producción permanente.

Este modelo, también podría ponerse a disposición de otros operadores de servicios de salud, para beneficio de sus pacientes. La finalidad de este trabajo es servir como un estudio predictivo de diagnóstico de sospecha de cáncer de mama y contribuir a un diagnóstico oportuno de este padecimiento en mujeres que presenten algún tipo de sintomatología en sus primeras etapas.

[1] En lo sucesivo utilizaremos solamente IXCHEN para referirnos al Centro de Mujeres IXCHEN

2. Antecedentes

Las técnicas y modelos de machine learning están posicionándose rápidamente en muchas áreas del quehacer cotidiano, entre las cuales, es oportuno destacar la medicina. Por ejemplo, se han desarrollado, desde modelos aplicables en el diagnóstico y pronóstico de enfermedades, hasta el desarrollo de fármacos y epidemiología. Así pues, esta disciplina tiene un potencial significativo de transformar el panorama médico [12].

El primer estudio realizado sobre el diagnóstico temprano del cáncer de mama data del año 1989 [13]. En el mismo, se desarrolló un modelo para determinar las probabilidades de desarrollar cáncer de mama en mujeres blancas que accedían a ser examinadas anualmente. El modelo utilizaba parámetros de antecedentes familiares y personales para determinar la probabilidad de desarrollo de esa enfermedad. Parte de los parámetros incluyen: edad de primera menstruación, edad de la paciente, numero de biopsias previas y numero de parentescos en primer grado con familiares que habían sufrido antes este padecimiento.

Investigaciones posteriores realizadas sobre la enfermedad brindan una gran cantidad de información que marca un punto de entrada para el desarrollo de modelos de aprendizaje automático que nos ayudan al diagnóstico de la enfermedad. Entre ellas, se destaca el conjunto de datos creado por la Universidad de Wisconsin [5] en el cual se representan las características computadas a partir de imágenes digitalizadas de un aspirado con aguja fina de una masa mamaria. A partir de este conjunto de datos se han realizado varias investigaciones [14] [15] [16] [17].

Estos trabajos se enfocaron en aspectos específicos del análisis del conjunto de datos, tales como: uso de diferentes técnicas de aprendizaje automático para el diagnóstico de cáncer de mama [16] [17] y métodos para elegir una buena configuración del algoritmo KNN (K-vecinos más cercanos, por sus siglas en inglés), en la detección de cáncer de mama [18].

Un uso aplicativo de los modelos de aprendizaje para el diagnóstico del cáncer de mama se puede ver en “Modelo en Machine Learning para el diagnóstico de cáncer de mama” [3] donde se desarrolla un software que muestra los resultados de distintos algoritmos de aprendizaje según el conjunto de datos introducido. Parte de los algoritmos de aprendizaje utilizados incluyen: regresión logística, arboles de decisión y máquinas de soporte vectorial, por sus siglas en inglés SVM.

Actualmente, en Nicaragua, se han desarrollado trabajos dentro del campo de la inteligencia artificial, mayormente enfocados en robótica [19] [20]. Sin embargo, no se encontró bibliografía de trabajos de inteligencia artificial aplicados al área de la medicina y el diagnóstico temprano de enfermedades en este país. Por lo tanto, se puede presumir que este sería el primer proyecto en el que se realizará una adaptación tecnológica parcial, con la finalidad de construir modelos de aprendizaje automático para la detección temprana de cáncer de mama.

Se procedió entonces a buscar trabajos sobre este mismo tópico, realizados a nivel de la región del Caribe, América Central y del Sur. Específicamente, se realizó la búsqueda de información sobre trabajos de investigación o innovación donde se abordan, con el uso de técnicas y/o modelos de machine learning, el diagnóstico de cáncer de mama. Entre ellos se destacan:

1. Proyecto de Investigación “Modelo en machine learning para el diagnóstico del cáncer de mama” [3]. Se publicó en el año 2020 en la Universidad Distrital Francisco de Caldas, Colombia. En este trabajo, se aplicaron modelos de machine learning para el diagnóstico del cáncer de mama utilizando un conjunto de datos elaborado por investigadores de la Universidad de Wisconsin [5]. Además, desarrollaron un aplicativo llamado BreastApp, el cual les permite realizar entrenamiento y predicciones por medio de un modelo de servicios.
2. Tesis para obtener grado de maestría “Análisis de la implementación de Machine Learning en el diagnóstico por imágenes” [4]. Publicado en el año 2019 en la Universidad de San Andrés, Argentina. Este trabajo tiene por objetivo realizar un análisis de los principales beneficios y barreras

asociados a la implementación de Machine Learning en el Diagnóstico por Imágenes poniendo un especial foco en la eficacia y eficiencia obtenida, los aspectos éticos, regulatorios y el impacto en el rol del médico especialista y en el ecosistema de startups.

3. Proyecto de Investigación “Artificial Intelligence System Reduces False-Positive Findings in the Interpretation of Breast Ultrasound Exams” [6]. Publicado en el año 2021 en Nature. En este trabajo, se presenta un sistema de inteligencia artificial que logra una precisión a nivel de radiólogo para identificar el cáncer de mama en imágenes de ultrasonido. El artículo también se aborda el tema de cómo los sistemas basados en inteligencia artificial reducen la tasa de falsos-positivos en la interpretación de exámenes de ultrasonido de mama. Para su desarrollo, utilizaron un conjunto de datos [7] consistente de 288,767 exámenes de ultrasonido de 143,203 pacientes examinados en centro NYU Langone Health, entre el año 2012 y 2019. Durante las pruebas realizadas, el sistema de inteligencia artificial logra un valor de la métrica AUROC (Area Under the Receiver Operating Curve) de 0.976.
4. Proyecto de Investigación “U-Net: Convolutional Networks for Biomedical Image Segmentation” [8]. Publicado en el año 2015 como un artículo de conferencia para la “Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015”. En este trabajo, se presenta una arquitectura y estrategia de entrenamiento basada en el fuerte uso de las técnicas de “Data Augmentation”, para usar las muestras anotadas disponibles de manera más eficiente. Se propone una solución al problema planteado por la premisa de que el entrenamiento exitoso de redes neuronales profundas (redes de aprendizaje profundo o deep learning) requiere muchos miles de muestras de entrenamiento anotadas.
5. Proyecto de Investigación: “Dataset of Breast Ultrasound Images” [2]. Publicado en el año 2020 como un artículo del periodo “Data in Brief”. En el artículo, se detalla el proceso de creación de un conjunto de datos de imágenes de medicas de cáncer de mama usando escáneres de

ultrasonido. El conjunto de datos esta categorizado en 3 clases: normal, benigno y maligno. La finalidad de este es el de ser utilizado en tareas que permitan la clasificación, detección y segmentación de cáncer de mama en conjunto con técnicas de machine learning.

3. Justificación

En Nicaragua, el cáncer de mama es la segunda causa de defunciones en mujeres con edades entre 40 y 44 años [21]. La tasa de mortalidad es del 23%, es decir, fallecen 23 de cada 100 mujeres diagnosticadas con la enfermedad, siendo superado solo por el cáncer cervicouterino [21].También es la segunda causa de muerte en la región de las Américas [22].

El mapa nacional de la salud en Nicaragua presenta estadísticas generales de padecimientos de enfermedades desde el año 2017 a agosto del año 2020 [1]. En este se muestra un aumento de defunciones por tumores malignos en glándulas mamarias para el año 2018, con 225 fallecimientos, y un total de 206 y 172 muertes registradas para los años 2019 y de enero a agosto del año 2020, respectivamente.

La OPS/OMS trabaja con los ministerios de salud en América Latina y el Caribe para prevenir el cáncer, detectarlo de forma temprana y mejorar el tratamiento [22]. Sin embargo, todo esto bajo técnicas estándares, que incluyen: mamografía, ecografía mamaria, resonancia magnética, entre otras. Dichas técnicas son realizadas en dependencia de la situación y síntomas específicos del paciente.

En los últimos años, el surgimiento de GPUs (**Graphics Processing Unit)**, facilitó una mejora significativa en el desempeño de las computadoras y el desarrollo de software especial, siendo posible trabajar con Big Data y algoritmos de machine learning [23]. Este campo de estudio brinda a las computadoras la capacidad de aprender sin estar explícitamente programadas [24]. Se centra en desarrollar modelos de aprendizaje que se ajustan en base a los datos que consume.

Por lo antes expuesto, en este proyecto se realizó un proceso de adaptación tecnológica parcial, tomando como referentes, trabajos sobre machine learning y deep learning, desarrollados en otros países [3] [2] [3]. El mismo fue complementado con aportes propios.

El modelo implementado es accesible a través de una página web con un formulario para la entrada de datos. Por este medio, los médicos podrán hacer uso del mismo para el diagnóstico oportuno del cáncer de mama, tratar a tiempo la enfermedad y reducir los índices de mortalidad entre las pacientes afectadas.

4. Objetivos

4.1 Objetivo General

Implementar un modelo de machine learning para el diagnóstico de cáncer de mama, como parte de un proceso de adaptación tecnológica.

4.2 Objetivos Específicos

1. Identificar los orígenes de datos pertinentes y modelos factibles para su implementación.
2. Procesar la data recopilada para su uso en la implementación del modelo.
3. Crear un modelo de aprendizaje automático adecuado para la determinación del carácter de una enfermedad mediante el examen de sus signos.
4. Implementar el modelo creado para que sirva de apoyo al diagnóstico médico de cáncer de mamas.

5. Marco Teórico

5.1 ¿Qué es el cáncer?

Las células constituyen unidades básicas que conforman los tejidos y órganos del cuerpo humano. Estas crecen y se dividen para producir nuevas, a medida que el cuerpo las necesita. Por lo general, las células mueren cuando envejecen demasiado o son dañadas. Luego, las nuevas toman su lugar.

La palabra cáncer proviene del griego karkinoma y del latín cáncer (cangrejo, tumor). Se define así a un grupo de enfermedades caracterizadas por el crecimiento excesivo y descontrolado de células que invaden y dañan tejidos y órganos, provocando finalmente la muerte del individuo. [25]

La enfermedad se desarrolla a partir de la acumulación y selección sucesiva de alteraciones genéticas y epigenéticas, que permiten a las células sobrevivir, replicarse y evadir mecanismos reguladores de apoptosis, proliferación y del ciclo celular [26]. Las células comienzan a crecer de manera descontrolada y pueden formar una masa llamada tumor. Este puede ser canceroso o benigno. Un tumor canceroso es maligno, lo que significa que puede crecer y diseminarse a otras partes del cuerpo. Un tumor benigno significa que el tumor puede crecer, pero no se diseminará, aunque el simple hecho de su crecimiento podría generar afectaciones a la salud del paciente.

5.2 Tipos de cáncer

Los médicos dividen el cáncer en distintos tipos, en función del lugar donde empieza. Los cuatro tipos de cáncer principales son:

- **Carcinomas:** comienza en la piel o el tejido que cubre la superficie de los órganos internos y las glándulas. Estos, por lo general forman tumores sólidos y son el tipo más frecuente de cáncer. Ejemplos de carcinomas incluyen: cáncer de próstata, cáncer de mama y cáncer de pulmón.

- **Sarcomas:** comienza en los tejidos que sostiene y conectan el cuerpo. Puede desarrollarse en la grasa, los músculos, los nervios, los tendones, las articulaciones o los huesos.
- **Leucemias:** es un cáncer en la sangre. En este tipo, las células sanguíneas sanas cambian y proliferan sin control. Algunos tipos son: leucemia linfocítica aguda, leucemia linfocítica crónica, leucemia mieloide aguda y leucemia mieloide crónica.

5.2.1 Cáncer de mama

Este padecimiento es desarrollado cuando las células de la mama empiezan a cambiar y crecer sin control, y forman un conglomerado de células que se denomina tumor. Este se puede observar en una radiografía o se puede palpar como una masa o bulto. Las células cancerígenas se pueden diseminar a otras partes del cuerpo ya sea porque crecen en esa parte del cuerpo o se desplazan a través de los vasos sanguíneos o linfáticos. A esto, se le denomina metástasis.

5.2.2 Detección

Se han desarrollado y continúan desarrollando pruebas que ayudan a encontrar este tipo de cáncer antes de la aparición de signos o síntomas. Teniendo en cuenta que el desarrollo de nuevas técnicas de detección de este padecimiento es un área activa, se considera oportuno destacar:

Exploración clínica de la mama: Un médico observa y palpa la mama para detectar cambios en su tamaño y forma.

Ultrasonido de mama: Estudio a través de un transductor que envía ondas sonoras de alta frecuencia y recibe ondas del eco, sirve para detectar cambios en el aspecto y función de los órganos, tejidos y vasos con el fin de detectar masas anormales o tumores en las mamas.

Mamografía: Es un tipo de radiografía específicamente diseñada para ver los tejidos mamarios. Esta prueba crea imágenes en las cuales se pueden mostrar tumores o irregularidades en las mamas.

Biopsias: Es un estudio citológico de tejidos extraídos de nódulos o masas sospechosas en las mamas y que permite clasificar el tipo de afectación presente en el tejido mamario.

Resonancia magnética (RM): Con frecuencia se utiliza en mujeres que hayan sido diagnosticadas con la enfermedad y ayuda a medir el tamaño del tumor e identificar otros tumores. Cabe destacar que esta técnica de detección se hace con menor frecuencia en Nicaragua.

5.3 Machine Learning (Aprendizaje Automático)

Es un subcampo de la inteligencia artificial que se encarga de encontrar patrones a través de los datos. Adicionalmente, se puede definir como el arte o la ciencia de programar computadoras de tal forma que tengan la habilidad de aprender de los datos.

Una definición formal se obtiene de [27]:

Se dice que una computadora aprende de la experiencia E con respecto a algún tipo de tareas T y una medida de desempeño P, si su desempeño en las tareas de T, medidas por P, mejora con la experiencia E.

Esta disciplina rompe con el enfoque tradicional de desarrollo de software, basada en manejar con datos de entrada según reglas ya establecidas. En el enfoque de aprendizaje automático, tenemos la suficiente cantidad de datos, pero no las reglas que asocian estos datos. El proceso de "**entrenar**" un modelo de aprendizaje, consiste en aprender estas reglas de asociación.

Este campo está dividido en diferentes tipos [24], en dependencia de: si es o no entrenado con supervisión humana (supervisado, no supervisado, semi supervisado y aprendizaje por reforzamiento), si puede o no aprender incrementalmente (aprendizaje en línea y por lotes) y si trabajan comparando datos nuevos con datos ya conocidos en vez de detectar patrones en los datos de entrenamiento (aprendizaje basado en instancias y basado en modelos). En este trabajo solo nos enfocaremos en el primer tipo.

5.3.1 Aprendizaje supervisado

En este tipo de aprendizaje la solución deseada es parte del conjunto de datos de entrenamiento, esta es también llamada la “etiqueta”. Los problemas que soluciona este tipo de aprendizaje se pueden agrupar en problemas de clasificación y regresión [24].

Clasificación: en este tipo de problemas, la variable a predecir es una categoría. Por ejemplo, predecir si un correo es spam o no.

Regresión: es cuando la variable a predecir es una cantidad numérica. Por ejemplo, el precio que tendrá un producto en cierta época del año.

A pesar de esta separación de enfoques, en ciertas tareas y algoritmos, ambos enfoques son utilizados. El algoritmo de regresión logística es usualmente utilizado en tareas de clasificación, ya que su salida es una probabilidad de que un valor corresponda a una categoría. Por ejemplo, en la clasificación de un correo como spam o no spam, la salida del algoritmo podría ser una probabilidad del 20% de que sea spam, por lo tanto, este será clasificado como no spam.

5.3.2 Aprendizaje no supervisado

Los datos de entrenamiento no están etiquetados. El principal objetivo del aprendizaje no supervisado es encontrar la estructura detrás de un conjunto de datos de tal forma que podamos aprender de los datos [24]. Los problemas que soluciona este tipo de aprendizaje pueden ser divididos en: agrupamiento y asociación.

Agrupamiento: se descubren grupos de datos similares. Por ejemplo, agrupar usuarios por tipos de comportamiento de compras.

Asociación: el objetivo es encontrar relaciones entre atributos de un conjunto de datos. Un ejemplo común es en los supermercados, donde se descubre que dos productos son generalmente comprados juntos y una acción se deberá tomar al respecto.

5.3.3 Aprendizaje reforzado

En el contexto, el sistema de aprendizaje es llamado un agente [24], el cual se sitúa en un ambiente, puede tomar decisiones y obtener recompensas, ya sea positivas o negativas. Por lo tanto, el sistema debe de construir la mejor estrategia para obtener las mayores recompensas positivas a medida que pase el tiempo.

Su uso es frecuente en el desarrollo de sistemas que aprenden a jugar [28]. Para una persona, es muy difícil proveer a un algoritmo con los atributos necesarios para entrenar y evaluar un modelo con un conjunto de datos de entrenamiento. Por el contrario, se le hace saber al agente si perdió o ganó y usar esta información para aprender una función de evaluación que luego nos dará una probabilidad de ganar en cada posición posible.

5.4 Algoritmos de machine learning (Aprendizaje Automático)

A continuación, abordaremos algunos algoritmos que son comúnmente usados en tareas de clasificación o regresión dentro del aprendizaje supervisado y no supervisado. Sin embargo, a pesar de la gran variedad de algoritmos, es frecuente invertir mucho tiempo y dinero en el desarrollo de algoritmos. Estudios sugieren que los datos importan más que los algoritmos en problemas más complejos, e incluso se comportan de una forma idéntica cuando se les alimenta con la suficiente cantidad de datos [29].

5.4.1 Máquinas de Vectores de Soporte (SVM2)

Una definición formal [30] nos dice que:

El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. "De la mejor forma posible" implica el hiperplano con el margen más amplio entre las dos clases … el margen se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores.

[2] Por sus siglas en inglés: Support Vector Machine

Este algoritmo es particularmente utilizado en tareas de clasificación para datos complejos, pero también para datos de mediana escala. Existen distintas implementaciones de este modelo [24], pero todas se basan en una idea fundacional, la cual puede ser explicada con la siguiente figura:

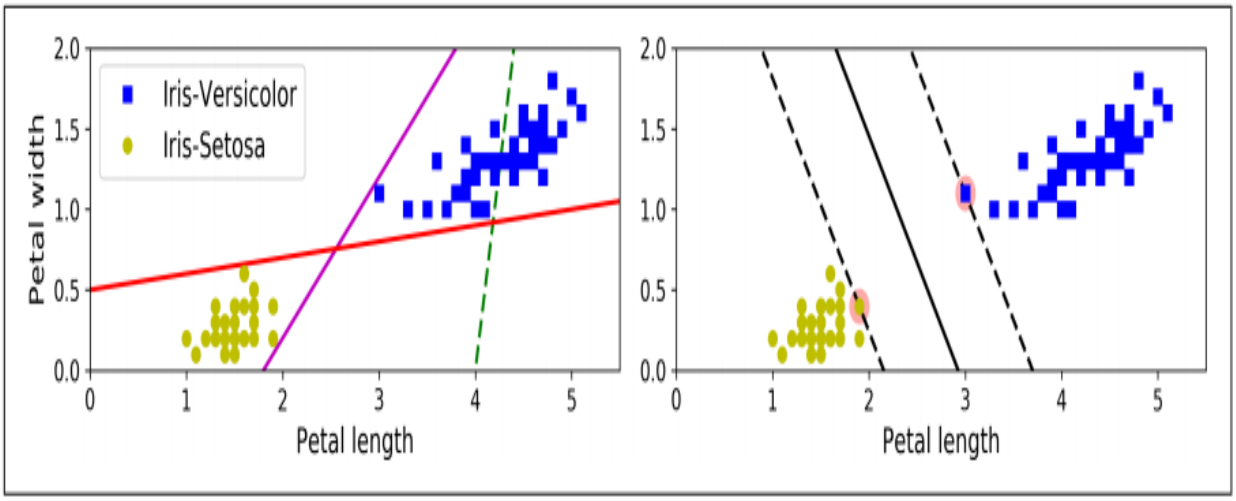


Figura 1: Clasificación de márgenes largos [24]

En esta imagen se puede observar una clasificación de flores de iris utilizando el algoritmo SVM aplicado al conjunto de datos de especies de iris ofrecido por el UCI Machine Learning Repository [31]. Dos clases son claramente separadas por medio de una línea recta (están linealmente separadas). En la imagen de la izquierda se muestran límites de decisión de 3 posibles clasificaciones lineales. Las líneas de decisión en la imagen derecha no solo separan las 2 clases si no que, se mantienen tan lejos de la instancia de entrenamiento más cercana como sea posible.

5.4.2 K-Vecino más Cercano (KNN)

Algoritmo de aprendizaje no supervisado utilizado para clasificación. Este mide la distancia de la nueva instancia de datos, con los datos de clasificación y elige el k-más cercano. Luego, se encuentra la clase que tenga la mayoría en ese grupo y se lo asigna a la nueva instancia de datos.

En la siguiente figura se muestra un ejemplo de una clasificación basada en el k-vecino más cercano:

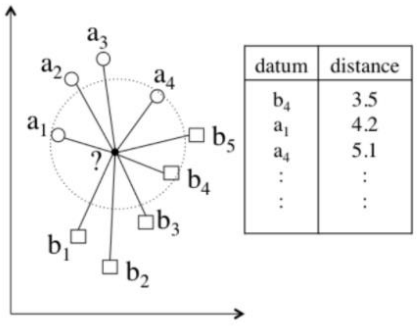


Figura 2: Ejemplo del uso del algoritmo KNN [18]

En este ejemplo se muestra una nueva instancia de datos (representado por el punto negro), la cual será clasificada como tipo a o tipo b. Se calcula la distancia del nuevo dato con los demás y se ordena para encontrar los 3 más cercanos. En la figura se muestra que 2 de las instancias de datos están clasificados como “a” y una instancia como “b”, así que la nueva instancia será clasificada como “a”.

5.4.3 Regresión Lineal

Usado como enfoque de aprendizaje supervisado. Es generalmente usado para modelar variables continuas y hacer predicciones [32]. En su forma más simple se ajusta una línea recta con el conjunto de datos. Esto es posible cuando la relación entre las variables es lineal. La principal ventaja de este algoritmo es que evita sobre ajustar el modelo, pero se vuelve difícil manejar relaciones no lineales.

En la siguiente figura se muestra un ejemplo de regresión lineal:

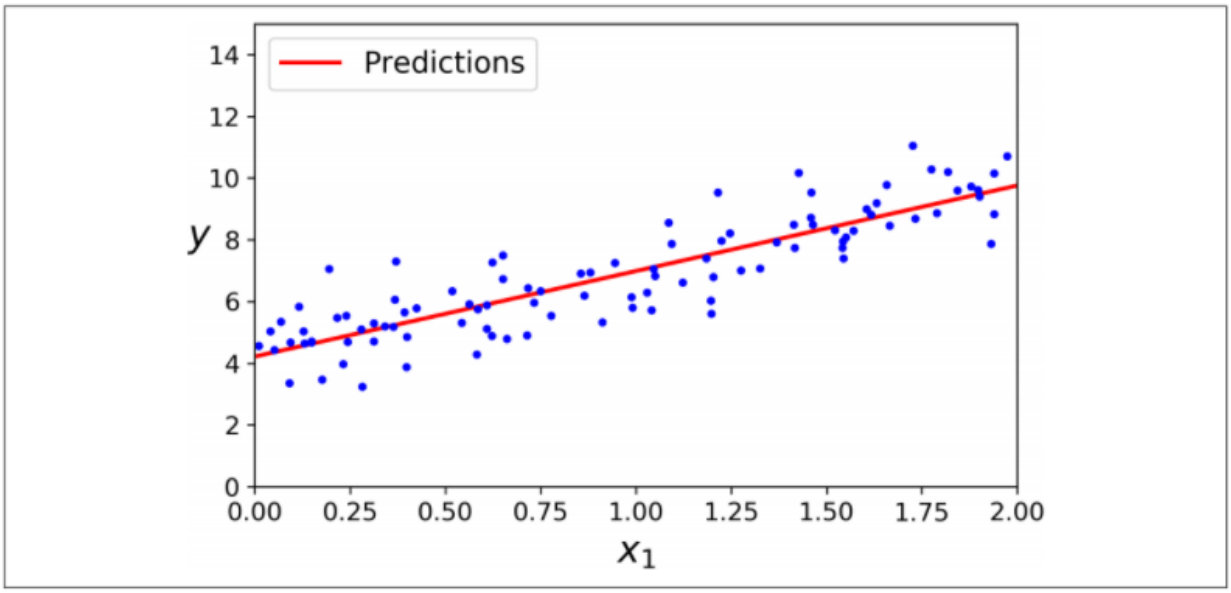


Figura 3: Predicciones de modelo de Regresión Lineal [24]

Este es un diagrama de dispersión que asocia las variables X1 e Y, esta asociación es representada por los puntos azules. A partir del entrenamiento del modelo de aprendizaje, se ajusta una línea recta con las predicciones (línea roja). Utilizando algoritmos de optimización se puede minimizar el error del modelo en los datos de entrenamiento. El proceso básico incluye, determinar la distancia entre predicción del modelo y la variable Y, ajustando la línea recta de manera que se minimice esta diferencia.

5.4.4 Regresión Logística

Como su nombre indica, es un algoritmo de regresión, pero que es mayormente utilizado para determinar la probabilidad de que una instancia de datos pertenezca a una clase en particular o no [24]. En dependencia de la cantidad de clases, la salida de este algoritmo puede ser binomial o multinomial. Un ejemplo de una salida binomial es la probabilidad de que un tumor sea maligno o benigno.

Parte de su funcionamiento es similar al de un algoritmo de regresión lineal, pero en vez de devolver la salida directa, a como lo hace este algoritmo, la regresión logística devuelve una salida logística del resultado. Esta “logística” no es más

que una función sigmoide que retorna un numero entre 0 y 1, generando la siguiente representación:

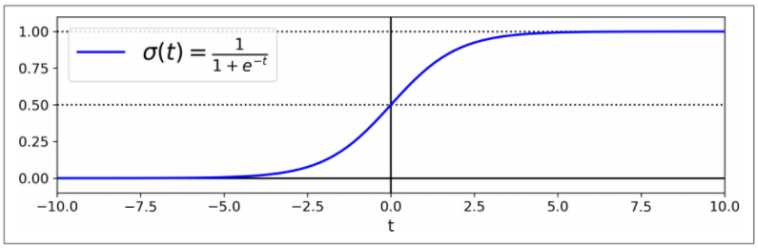


Figura 4: Regresión logística [24]

5.4.5 Redes Neuronales Artificiales

Su concepto es derivado de la definición biológica de las neuronas. Para entender una red neuronal artificial, debemos entender cómo funciona una neurona. Esta consiste principalmente en 4 partes: dendritas, núcleo, soma y axón, a como se muestra en la siguiente figura:

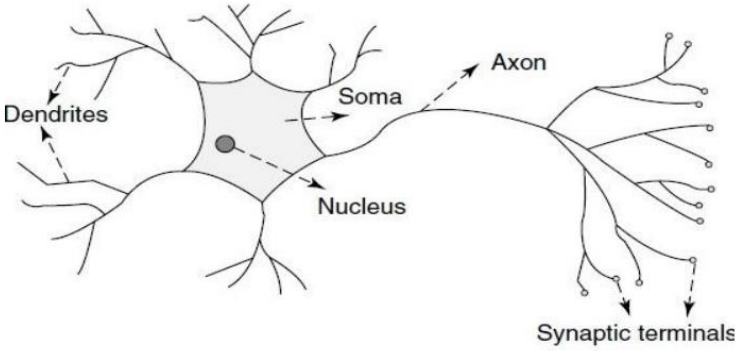


Figura 5: Neurona humana [33]

Las dendritas reciben señales eléctricas. El soma procesa estas señales y la salida es llevada por el axón a las dendritas terminales. Aquí la salida se manda a la siguiente neurona. El núcleo es el corazón de la neurona. Esta interconexión es llamada una red neuronal donde los impulsos eléctricos viajan alrededor del cerebro [34].

Las redes neuronales artificiales se comportan de la misma manera. Su estructura básica está compuesta de 3 capas: la de entrada (toma los datos de entrada), la oculta (procesa los datos) y de salida (envía la salida calculada). Basados en su patrón de conexión (arquitectura), una red neuronal puede ser agrupada en 2 categorías: redes de avance (el grafico no tiene bucles) y redes recurrentes (en las cuales los bucles ocurren debido a las conexiones de retroalimentación) [35].

El proceso de aprendizaje puede ser visto como un problema de actualizar la arquitectura interna de la red. Esto se refiere a renovar los “pesos” de las conexiones, de manera que esta pueda realizar una tarea eficientemente. Este proceso iterativo permite mejorar el desempeño de la red a medida que pasa el tiempo.

5.5 Deep learning y su ubicación dentro del machine learning

El aprendizaje automático convencional está limitado al dominio del problema y la habilidad para diseñar características que sean representativas del problema que se está tratando de resolver [36]. En este contexto, deep learning es un área del aprendizaje automático que permite aprender de datos sin procesar de tal forma que la red descubre las representaciones necesarias para detectar o clasificar.

El aprendizaje representativo del deep learning son métodos los cuales permiten obtener múltiples niveles de representación compuestos de módulos no lineales que transforman la representación en un nivel superior, un poco más abstracto [36].

5.6 Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo de red neuronal que se utiliza comúnmente en detección de objetos, reconocimiento y segmentación de imágenes [37]. En esta sección, se explorarán los elementos de las redes neuronales convolucionales, incluyendo el kernel, strides, padding, funciones de activación y capas de pooling y como se estructuran las redes neuronales para tareas de clasificación de imágenes.

La idea básica de las redes convoluciones y lo que hace las diferencia a arquitecturas básicas de perceptrón multicapa (MLP, MultiLayer Perceptron, por sus siglas en inglés), es que las CNN utilizan capas de convolución y pooling para extraer características relevantes de las imágenes de entrada. Estas capas están diseñadas para reducir la dimensionalidad de la imagen y preservar información espacial importante [38].

Además, las redes convolucionales tienen menos parámetros que las MLP debido a que utilizan filtros compartidos en las capas de convolución, lo que las hace más eficientes y fáciles de entrenar. Las MLP, por otro lado, tienen capas completamente conectadas, donde cada perceptrón está conectado con todos los demás perceptrones, lo que puede resultar en un gran número de parámetros y dificultades para entrar el modelo [39].

A continuación, se detallan algunos elementos de las redes convoluciones y que deberán de ser ajustados manualmente dentro del proceso de ajuste de hiperparámetros:

Kernel: es un filtro que se utiliza en la convolución para detectar patrones en la imagen. El tamaño del kernel y sus valores se determinan durante el entrenamiento de la CNN, y se utilizan para convolucionar (operar con) la imagen de entrada y obtener una imagen de salida [40]. La convolución es el proceso de desplazar el kernel por la imagen de entrada y calcular el producto punto a punto entre los valores del kernel y los valores de los píxeles de la imagen de entrada.

Strides: determinan el desplazamiento del kernel durante la convolución, y, por lo tanto, afectan el tamaño de la imagen de salida. Si los strides son pequeños, la imagen de salida será más grande y se podrán detectar patrones más detallados [40]. Si los strides son grandes, la imagen de salida será más pequeña y se podrán detectar patrones más generales.

Padding: es una técnica que se utiliza para mantener el tamaño de la imagen de entrada y evitar la pérdida de información en los bordes [41]. Consiste en agregar ceros alrededor de la imagen de entrada antes de aplicar la convolución.

Funciones de activación: se utilizan en las capas de la CNN para agregar no linealidad a la red [41]. Las funciones de activación más comunes son la función ReLU, la función sigmoid y la función tanh.

Capas de pooling: se utilizan para reducir el tamaño de la imagen y disminuir el costo computacional de la red [41]. Las capas de max pooling y average pooling son las más utilizadas, y consisten en reducir la dimensión de la imagen al tomar el valor máximo o promedio de un conjunto de píxeles.

Estos elementos de las redes convolucionales sirven como un proceso que extrae las características de las imágenes usando las capas convoluciones y pooling. Luego, utilizamos estas características para clasificar imágenes, para eso se pueden utilizar una arquitectura de red neuronal típica, como una MLP [40].

En problemas de clasificación de imágenes, se sigue una arquitectura compuesta de 2 partes, una con elementos de las redes convolucionales para extraer de forma automática las características de la imagen y luego otra que se encarga de, una vez extraídas, clasificar las imágenes en una de varias de clases [40].

En la siguiente imagen se ilustra esta arquitectura:

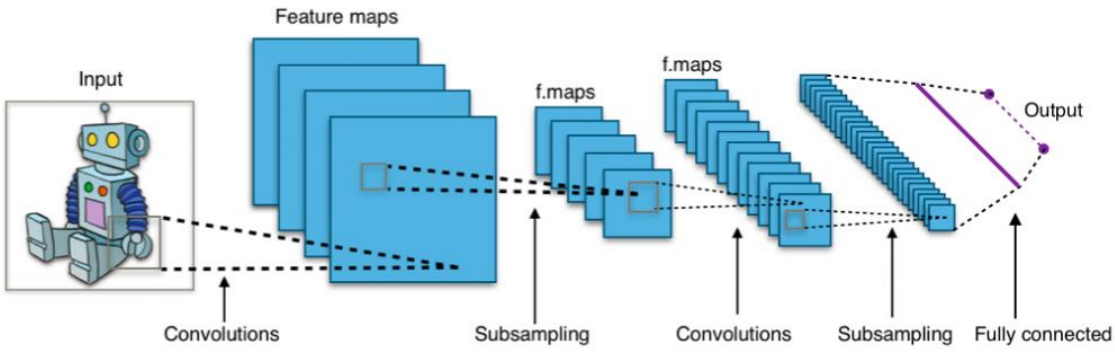


Figura 6: Arquitectura típica de una red neuronal convolucional [42]

En ella se observan algunos de los elementos que componen a las redes convoluciones (convolutions y feature maps) y el elemento final para clasificar la imagen (fully connected).

Además de los elementos de las redes neuronales convolucionales, también se presentarán arquitecturas comunes de redes neuronales convolucionales

utilizadas en la segmentación de imágenes, como la arquitectura VGG, la arquitectura ResNet y la arquitectura U-Net.

En resumen, esta sección presenta los elementos fundamentales de las redes neuronales convolucionales, desde el kernel hasta las capas de pooling, y su importancia en la clasificación de imágenes.

5.6.1 Segmentación semántica

La segmentación semántica es una técnica de procesamiento de imágenes que consiste en etiquetar cada píxel de una imagen con una categoría correspondiente. En otras palabras, la segmentación semántica permite dividir una imagen en diferentes regiones con significado semántico, como personas, objetos, fondos, etc. En este capítulo se explorará en detalle esta técnica, sus principales enfoques y algoritmos utilizados en la actualidad.

La segmentación semántica se divide en dos enfoques principales: basados en píxeles y basados en regiones. Los enfoques basados en píxeles se centran en etiquetar cada píxel individualmente en la imagen, mientras que los enfoques basados en regiones se centran en segmentar la imagen en regiones y luego etiquetar cada región.

Dentro de los enfoques basados en píxeles, existen diferentes algoritmos utilizados en la segmentación semántica, como los basados en redes neuronales convolucionales, como la arquitectura U-Net. La U-Net es una red neuronal convolucional diseñada específicamente para la segmentación semántica, que utiliza una arquitectura de "codificador-decodificador" para aprender y reconstruir la imagen de salida.

Por otro lado, los enfoques basados en regiones utilizan algoritmos de segmentación por agrupamiento, como la segmentación por felzenszwalb, que agrupan píxeles similares en regiones y luego etiquetan cada región con una categoría semántica.

Además, la segmentación semántica también puede ser evaluada utilizando diferentes métricas de evaluación, como la precisión, el recall y la medida F. La precisión se refiere a la proporción de píxeles correctamente clasificados en la imagen, mientras que el recall se refiere a la proporción de píxeles relevantes que han sido correctamente identificados.

En resumen, esta sección presenta los principales enfoques y algoritmos utilizados en la segmentación semántica, desde los enfoques basados en píxeles y regiones hasta la arquitectura U-Net y la segmentación por agrupamiento. Además, se presentan las métricas de evaluación comúnmente utilizadas para medir el desempeño de la segmentación semántica.

5.7 Selección y evaluación de modelos de aprendizaje

La evaluación del desempeño de un modelo de aprendizaje es crucial para determinar su capacidad para realizar tareas específicas. En este capítulo se explorarán las medidas de desempeño más utilizadas en modelos de aprendizaje, incluyendo tanto métricas de clasificación como de segmentación.

Para modelos de clasificación, una de las medidas de desempeño más comúnmente utilizada es la exactitud (accuracy), que se refiere a la proporción de predicciones correctas realizadas por el modelo en relación con el total de predicciones. Otras medidas de desempeño incluyen la precisión (precision), que se refiere a la proporción de predicciones positivas que son verdaderas en relación con el total de predicciones positivas, y el recall (recuperación), que se refiere a la proporción de verdaderos positivos que han sido identificados en relación con el total de casos positivos en los datos.

En modelos de segmentación, una de las medidas de desempeño más comúnmente utilizada es la métrica IoU (Intersección sobre Unión), que mide la similitud entre la predicción del modelo y la segmentación de la imagen de referencia. La métrica IoU se calcula dividiendo la intersección de la predicción del modelo y la segmentación de referencia por la unión de estas dos regiones. Además, también se utilizan medidas como el coeficiente de correlación de

Pearson (PCC) y el coeficiente de correlación de Spearman (SCC) para evaluar la similitud entre las predicciones del modelo y las segmentaciones de referencia.

Otras medidas de desempeño utilizadas en modelos de aprendizaje incluyen el área bajo la curva (AUC) para modelos de clasificación binaria, y la pérdida logarítmica (log loss) y el error cuadrático medio (MSE) para modelos de regresión.

En conclusión, esta sección presentó las medidas de desempeño más utilizadas en modelos de aprendizaje, incluyendo tanto métricas de clasificación como de segmentación. Estas medidas son esenciales para evaluar la capacidad de un modelo para realizar tareas específicas y mejorar su rendimiento.

5.8 Transfer learning

Transfer learning es una técnica de aprendizaje automático en la que el conocimiento aprendido de una tarea se reutiliza para mejorar el rendimiento en una tarea relacionada. Esta técnica permite a los científicos de datos beneficiarse del conocimiento adquirido de un modelo de aprendizaje automático previamente utilizado para una tarea similar.

Un modelo pre-entrenado es un modelo que ya ha sido entrenado en un conjunto de datos grande y generalmente disponible públicamente, como ImageNet. En lugar de entrenar un modelo desde cero, que puede ser costoso en términos de tiempo y recursos, se puede utilizar un model pre-entrenado como punto de partida para entrenar un modelo en una tarea relacionada.

Un ejemplo de transfer learning es en el campo de la visión por computadora, es común utilizar modelos pre-entrenados en ImageNet para tareas de clasificación de imágenes. ImageNet es un conjunto de datos grande y diverso que contiene millones de imágenes etiquetadas en miles de categorías diferentes. Al utilizar un modelo pre-entrenado en ImageNet, se puede aprovechar el conocimiento adquirido por el modelo al aprender a reconocer patrones y características en las imágenes. En transfer learning, se utilizan las capas tempranas y medias y solo se vuelven a entrenar las ultimas capas.

Transfer learning tiene varios beneficios, pero las principales ventajas son el ahorro de tiempo de entrenamiento, el mejor rendimiento de las redes neuronales (en la mayoría de los casos) y no necesitar una gran cantidad de datos. Con transfer learning, se puede construir un modelo sólido de aprendizaje automático con relativamente pocos datos de entrenamiento porque el modelo ya está pre-entrenado.

En Anexos 1 se muestra un ejemplo en pseudo código donde se carga un modelo pre-entrenado en ImageNet y se congelan sus capas para que se actualicen durante el entrenamiento. Luego, se agregan nuevas capas al final del modelo base para adaptarlo a la nueva tarea de clasificación de imágenes. Finalmente, se compila y entrena el nuevo modelo en el conjunto de datos de entrenamiento.

6. Metodología

El desarrollo de una solución de aprendizaje automático requiere mucho más que solo seleccionar un modelo, entrenarlo y aplicarlo a datos nuevos. Su entrenamiento, es solo una pequeña parte del proceso. El proceso CRISP-DM (siglas en inglés de: Cross-Industry Standard Process for Data Mining), o proceso estándar entre industrias para la minería de datos, es una de esas técnicas y marcos de trabajos que nos ayudan a organizar proyectos de aprendizaje automático de tal manera que no se salgan de control. De acuerdo con el proceso CRISP-DM, los proyectos de aprendizaje automático consisten en 6 etapas:

1. Entendimiento del negocio
2. Entendimiento de los datos
3. Preparación de los datos
4. Modelado
5. Evaluación
6. Despliegue

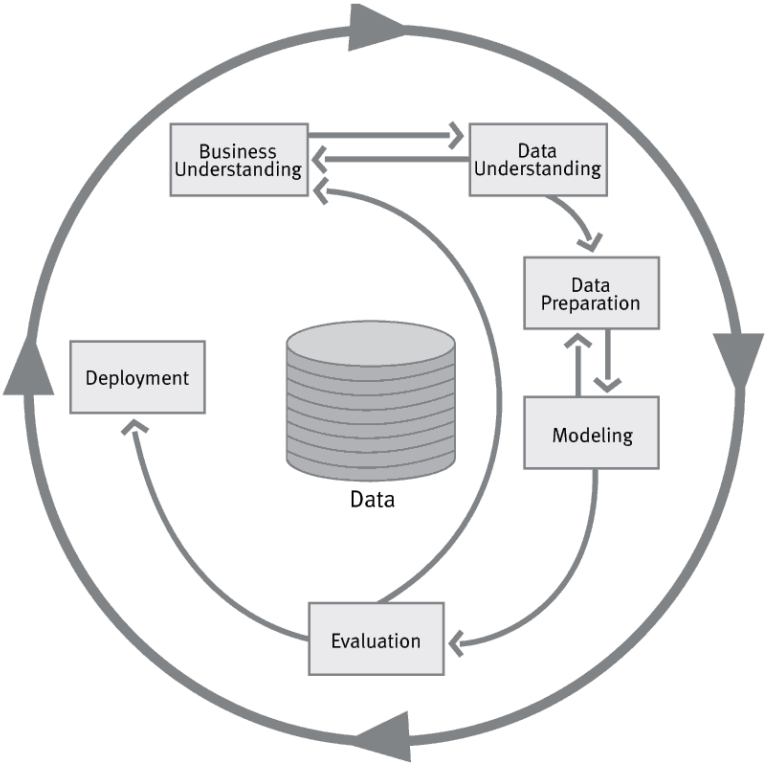


Figure 7: Fases de la metodología CRISP-DM [43]

6.1 Entendimiento del negocio

En esta etapa, primero se define el problema a resolver. Luego, se identifican y analizan las posibles soluciones existentes. Finalmente, se trata de determinar si, aplicando modelos de machine learning, es posible obtener algún beneficio [44].

Al implantar esta etapa, en el presente proyecto, se incluyó varias tareas. La primera de ellas fue:

1. Identificar el objetivo del modelo y como se relaciona con los objetivos de esta tesis monográfica.

Una forma de lograrlo es realizando preguntas cuyas respuestas coadyuven a establecer la relación entre el objetivo del modelo aplicado y los objetivos del proyecto monográfico. Por ejemplo:

- ¿Cuál es el propósito del modelo?
- ¿Se trata de mejorar la precisión del diagnóstico?
- ¿Acelerar el proceso de detección?
- ¿Reducir los costos?

En este caso, se determinó que el objetivo principal es implementar un modelo de machine learning, para el diagnóstico de cáncer de mama, como parte de un proceso de adaptación tecnológica. No se trata de mejorar la precisión del diagnóstico, pero sí, lograr que este sirva como un recurso adicional y de utilidad en la lucha contra este flagelo. En principio, estará a disposición del personal médico del centro donde se ha llevado a cabo el proyecto (IXCHEN – Managua). La idea es que sirva de apoyo en el diagnóstico y detección de patologías asociadas al cáncer de mama en las usuarias atendidas. Adicional a esto, se plantean los siguientes objetivos:

- Identificar los origines de datos pertinentes y modelos factibles para su implementación.
- Procesar la data recopilada para su uso en la implementación del modelo.

- Crear un modelo de aprendizaje automático adecuado para la determinación del carácter de una enfermedad mediante el examen de sus signos.
- Implementar el modelo creado para que sirva de apoyo al diagnóstico médico del cáncer de mama.

La segunda tarea por realizar es:

2. Comprender el contexto en el que se utilizará el modelo.

Esto incluye consideraciones como:

- ¿Qué tipo de datos se utilizarán como entrada (por ejemplo, imágenes de mamografía o resultados de análisis de sangre, etc.)?
- ¿Quién tendrá acceso al modelo (podrían ser radiólogos, médicos generales, usuarios), y cómo se integrará el modelo en la práctica clínica?

En IXCHEN-Managua se utiliza el ultrasonido o ecografía, como uno de los exámenes principales para detectar el cáncer de mama en una usuaria. Si los resultados indican dudas o certezas sobre la presencia de la enfermedad, adicional a este, se pueden realizar otros exámenes, como, por ejemplo, biopsias por aspiración con aguja fina. Sin embargo, las muestras son analizadas por un laboratorio externo y IXCHEN solo recibe los resultados.

Debido a ello, en la implementación de este proyecto, se tuvo, como principal recurso disponible para el diagnóstico, imágenes de ultrasonidos de mamas. De manera que, el modelo a implementarse recibe esta información como la primera entrada de datos a utilizar. Así también, el estudio se vio condicionado por los métodos de aprendizaje automático que fueron utilizados para el desarrollo del modelo de aprendizaje. El conjunto de datos que se empleó será detallado en una etapa posterior.

La máquina de aprendizaje automático resultante es utilizada por las radiólogas de IXCHEN, en sus sucursales de la Sede Central (Managua) y la ubicada en Villa Libertad. El modelo es accedido por las facultativas a través de una interfaz web.

En la etapa de despliegue, se darán más detalles de cómo fue desarrollado e implementado, de tal forma que sea accesible para el personal médico vía web.

La tercera tarea por realizar se resume en:

3. Analizar el mercado y la competencia.

Esto incluyó tareas de indagación sobre qué otros modelos o herramientas de diagnóstico del cáncer de mama están disponibles en el mercado, cuáles son sus ventajas y desventajas, y cómo se diferencia de ellos, el modelo que se está desarrollando con el apoyo de IXCHEN.

Actualmente, existen otros trabajos que se han realizado sobre el uso de técnicas modernas de inteligencia artificial para la detección del cáncer de mamas. Sin embargo, en este caso, se tiene la restricción de utilizar las imágenes de ultrasonido, disponibles en IXCHEN, para el logro de los objetivos del proyecto. Esta restricción se debe a que el centro utiliza imágenes de ultrasonido como principal método de detección de cáncer de mama.

Por ello, se consideró como una referencia importante, el artículo titulado: “Artificial Intelligence System Reduces False-Positive Findings in the Interpretation of Breast Ultrasound Exams” [6]. En esta publicación científica, los autores proponen nuevas técnicas para reducir la tasa de falsos positivos. Complementario a este artículo, los autores también escribieron otro artefacto sobre el proceso que ellos siguieron para recolectar los datos de ultrasonido necesarios [7]. Este presenta un reporte del conjunto de datos de ultrasonidos de New York University, consistiendo en 288,767 exámenes de ultrasonidos de mama con un total de 5,442,907 imágenes adquiridas de 143,203 pacientes examinadas entre 2012 y 2019 en el centro NYU Langone Health. En este reporte, se resumen estadísticas del conjunto de datos, el proceso de recolección de las imágenes y el procedimiento de procesamiento. A lo largo de este documento, se abordará cómo fue aplicado en este trabajo lo aprendido de estas publicaciones.

Adicionalmente, se consideraron otros trabajos [45] publicados por investigadores de instituciones científicas de Latinoamérica, en las cuales utilizan inteligencia

artificial aumentativa en imagenología diagnóstica. Se destaca la solución brindada por la plataforma **Indiria** [46]**,** la cual es fue desarrollada por **Indigo Technologies** para el reconocimiento de 25 patologías complejas como COVID 19, diversos tipos de cáncer, enfermedades cardiovasculares y trastornos de las vías respiratorias.

Como cuarta tarea se plantea:

4. Establecer un marco de referencia.

Esto incluye la definición de los criterios que se utilizarán para medir el éxito del modelo (por ejemplo, tasa de precisión, tasa de falsos positivos, etc.), así como establecer las metas y objetivos específicos que se espera alcanzar con el mismo. Específicamente, en este proyecto, se definió la implantación de un modelo híbrido que combina la segmentación de imágenes con un modelo de clasificación. Las métricas utilizadas, toman en cuenta las características de ambos combinados.

Para el modelo de segmentación, se utiliza como entrada una imagen de ultrasonido de mama. El resultado obtenido es una imagen segmentada en la cual se muestran las zonas donde el modelo detecta que puede haber una lesión, basado en los datos de entrenamiento. En la Figura 7, se muestran dos imágenes. A la izquierda, está la imagen de ultrasonido utilizada como entrada de datos a procesar. A la derecha, se muestra otra imagen donde es posible apreciar el resultado de la segmentación producida por el modelo de aprendizaje.

Para medir la precisión de este modelo, en este proyecto, se implementó un mecanismo de retroalimentación. En este caso, un equipo médico, conformado por personal experto en radiología, mediante un proceso de observación empírica y de acuerdo con la experiencia profesional de sus integrantes, manualmente, seleccionan cual es la precisión de la segmentación. Se insiste en hacer notar que en esta parte del proceso es de gran importancia los criterios y experiencia del equipo de facultativos. De esta manera, el personal médico puede elegir un valor de precisión en el sub-rango [0 – 100]. El valor aportado, se promedia a continuación, con el total de exámenes en el periodo de pruebas.

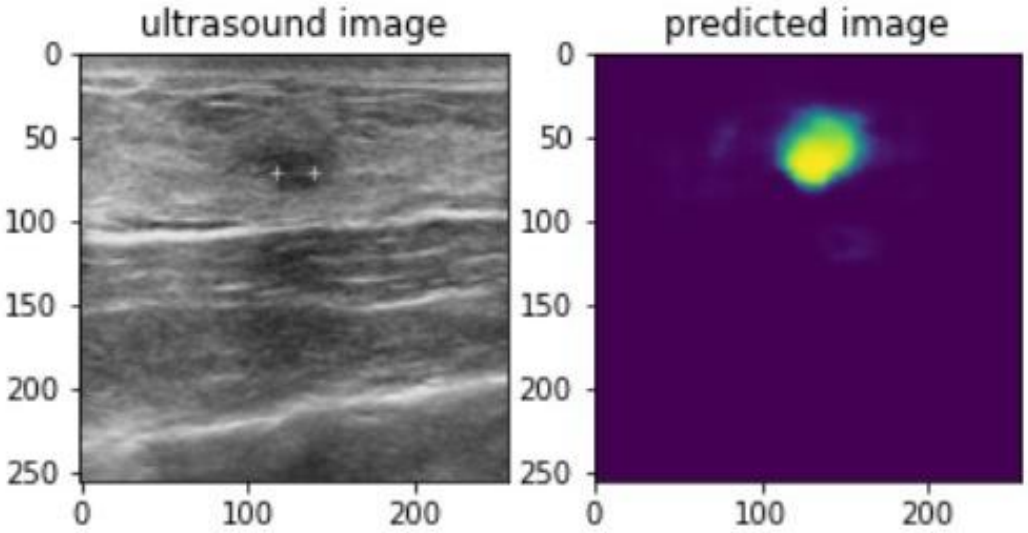


Figura 7: Comparación entre imagen de ultrasonido y la segmentación producida. Fuente: Imagen propia.

Cabe aclarar que cuando hablamos de promedio, nos referimos a la media aritmética, la cual es la calculada de la siguiente manera: se suman todas las precisiones reportadas en el sistema, el total es dividido por el total de registros obtenidos. Como resultado, obtenemos la media de precisión en las primeras pruebas con el sistema.

En modelos de clasificación, para datos médicos, es de interés maximizar las métricas que tengan una alta sensibilidad para detectar cáncer de tipo maligno [6]. Se profundizará sobre este tópico más adelante, al abordar la etapa de pruebas.

Como resumen, se puede decir que, la etapa de entendimiento del negocio en el modelo CRISP-DM, se refiere, principalmente, a la recopilación y análisis de información relevante sobre el problema y su contexto en el negocio (trabajo a realizar), para establecer un marco de referencia en el desarrollo y evaluación del modelo de aprendizaje automático implementado.

6.2 Entendimiento de los datos

Se refiere a la compilación, limpieza y análisis de los datos que se utilizarán como entrada para el modelo de aprendizaje automático [44]. Aplicado al presente trabajo investigativo, esta etapa podría incluir tareas como:

1. Recopilar los datos necesarios para entrenar y evaluar el modelo.

En casos como este, se podrían incluir imágenes de mamografía, resultados de análisis de sangre, resultados de biopsias con aguja fina y otros datos clínicos de pacientes con y sin cáncer de mama. En este proyecto, se optó por el uso de ecografías debido a la disponibilidad de ellas en IXCHEN. Cabe señalar que al iniciar el mismo, no se disponía del servicio de mamografía en el Centro.

Los datos utilizados para desarrollar, tanto el modelo de segmentación, como el de clasificación, provienen de dos fuentes: la primera [2] se utilizó para entrenar el modelo; la segunda, es el conjunto de datos que se ha creado a lo largo de este proyecto con la colaboración de IXCHEN. Cabe destacar, que ambos conjuntos de datos son imágenes ultrasonográficas donde se pueden observar nódulos y masas, a diferencia de mamografías que usan rayos X para visualizar imágenes quísticas o microcalcificaciones.

La fuente de datos de entrenamiento consiste en imágenes en formato PNG, las cuales vienen en pares con su “máscara” de segmentación y clasificadas en benigno, maligno o normal. Este conjunto de datos fue recolectado en una afiliación de la Facultad de Computación e Inteligencia Artificial de la Universidad del Cairo, Egipto, y el Instituto Nacional del Cáncer de la Universidad del Cairo, Egipto [2].

Los datos de entrenamiento fueron recolectados en mujeres con edades entre 25 y 75 años, en el año 2018, en el Hospital Baheya [2], con un total de 600 pacientes. Las imágenes recolectadas consisten en 780 imágenes y su par de máscara de segmentación, con un promedio de tamaño de 500x500 pixeles. En Anexos 2, se

muestran ejemplos de cómo lucen estas imágenes, una vez descargadas en la estación de trabajo local.

Al momento de empezar a trabajar en este proyecto, IXCHEN solo contaba con dispositivos de ultrasonidos para el diagnóstico por imágenes a usuarias que llegan al centro solicitando atención médica por dolencias que hacían sospechar sobre un posible padecimiento de cáncer de mama; o bien, por solicitud expresa de ellas. Este fue el principal motivo para elegir este conjunto de datos sobre otros, como los recolectados con exámenes de mamografía, biopsias o pruebas de sangre.

Para el caso del conjunto de datos de prueba, las imágenes de ecografía son extraídas de un dispositivo de ultrasonido y convertidas en el mismo formato de las imágenes del conjunto de datos de entrenamiento. Estos datos fueron reunidos en un periodo de tiempo comprendido entre los meses de marzo de 2022 y enero del 2023 en IXCHEN-Managua.

El método utilizado para extraer los datos del dispositivo de ultrasonido consistió en filtrar los exámenes pertenecientes a usuarios que se realizaron ecografías de mamas, mismos que luego fueron exportados a una memoria USB. En este proceso, se crea una carpeta para cada estudio realizado, conteniendo las imágenes resultantes de los ultrasonidos efectuados en ese estudio. En Anexos 3 se muestra un ejemplo de cómo se exportan los exámenes en la memoria USB, también se muestran las imágenes individuales de los exámenes que se exportan sin formato.

Adicional a los DICOM, se exportaban los documentos con el diagnóstico del médico radiólogo a partir de las imágenes de ultrasonido.

Un segundo paso por realizar en esta etapa es:

2. Limpiar y preparar los datos para su uso en el modelo.

En este caso, se incluyeron tareas como: eliminar datos incompletos o inconsistentes, normalizar los datos para que tengan un formato consistente, y

seleccionar las características relevantes de los datos que se utilizarán como entrada para las pruebas del modelo.

En el caso del conjunto de datos de entrenamiento, estos se obtuvieron limpios y preparados para su preprocesamiento y posterior uso en el modelo. En cuanto al conjunto de datos de prueba, se hizo necesario, primero, convertir las imágenes al mismo formato de los datos de entrenamiento. De esta manera, el dataset estará listo para su procesamiento y uso en el modelo. Cabe destacar que, inicialmente, los datos de prueba estaban codificados en el formato estándar de los dispositivos de ultrasonidos utilizados (DICOM: Digital Imaging and Communications in Medicine), aunque se exporten sin formato.

El estándar DICOM, es comúnmente utilizado para guardar datos de imagenología en estudios de CT, MRI, PET, ultrasonidos y otros tipos de escaneos médicos [47]. Para leer este tipo de archivos en una computadora, existen distintos programas que muestran el contenido. Inicialmente, en este proyecto, se utilizó una librería de Python, llamada Pydicom, pudiendo así leer las etiquetas (tags), de los archivos y transformarlo al formato requerido por los datos de entrenamiento (en este caso, PNG). Posteriormente, en este mismo proyecto, se utilizó el módulo Input/Output de Tensorflow, el cual permite abrir la imagen en un formato específico. Cabe señalar, que estas funciones también utilizan internamente Pydicom. A continuación, se muestra un segmento del código necesario para hacer esto:

```
1  def cargar_dicom_file(image_bytes):
2      lossy_image = tfio.image.decode_dicom_image(image_bytes.getvalue(), scale='auto', on_error='lossy', dtype=tf.uint8)
3      numpy_lossy_image = np.squeeze(lossy_image.numpy())
4      return numpy_lossy_image
```

Figura 8: Función para obtener la representación numérica de archivo DICOM. Fuente: Imagen propia.

El parámetro “image_bytes” de la función “cargar_dicom_file” es el objeto que se recibe cuando se sube el archivo a la aplicación “streamlit”. La función,

“tfio.image.decode_dicom_image” recibe como parámetros el archivo dicom (DCM) en su representación de bytes, “scale”, escala la imagen automáticamente, ”on_error”, carga la imagen con perdida, es decir, en formato jpg, y luego el tipo de datos en que será decodificada, en este caso “int” de 8 bits.

Las imágenes están estructuradas. En su forma individual, están compuestas de 3 dimensiones: alto, ancho y canales de colores. Donde los canales de colores pueden ser RGB (3 canales: rojo, verde y azul) o blanco y negro (1 canal). Los valores que pueden tener van desde 0 a 255, donde cada valor representa la intensidad del canal de color e intensidades de blanco y negro para imágenes en escala de grises. Por lo tanto, sólo se necesitan campos de 8 bits en la representación numérica de las imágenes.

Luego, sobre el valor retornado por la función “tfio.image.decode_dicom_image”, se elimina cualquier dimensión extra que se haya podido agregar a su representación de Numpy array. Para ello, es utilizada la función “squeeze” de la librería Numpy.

Finalmente, esta representación numérica es utilizada en etapas posteriores como una imagen png, que es el formato original de las imágenes que se utilizaron para entrenar los modelos de aprendizaje.

A continuación, se procede a:

3. Analizar los datos para comprender su contenido y su distribución.

Esto podría incluir tareas como calcular estadísticas básicas (por ejemplo, media, desviación estándar), así como visualizar los datos utilizando gráficos y diagramas. También se debe realizar pruebas estadísticas para evaluar la calidad y la confiabilidad de los datos. Debido a la naturaleza no estructurada de estas imágenes, no es factible calcular este tipo de estadísticas básicas. Sin embargo, en este trabajo también se extrajo estadísticas que permiten saber cómo están distribuidas las imágenes para cada categoría. Esto permite considerar que, las clases con un mayor número de imágenes pueden tener mayor peso al momento de entrenar y hacer predicciones utilizando el modelo de clasificación.

El conjunto de datos base, sin incluir las imágenes de máscaras utilizadas en el modelo de segmentación, consiste en 437 imágenes para la clase benigno, 133 normal y 210 maligno. En la imagen a continuación, se observa la diferencia (en cuanto a la cantidad de registros), entre las 3 clases:

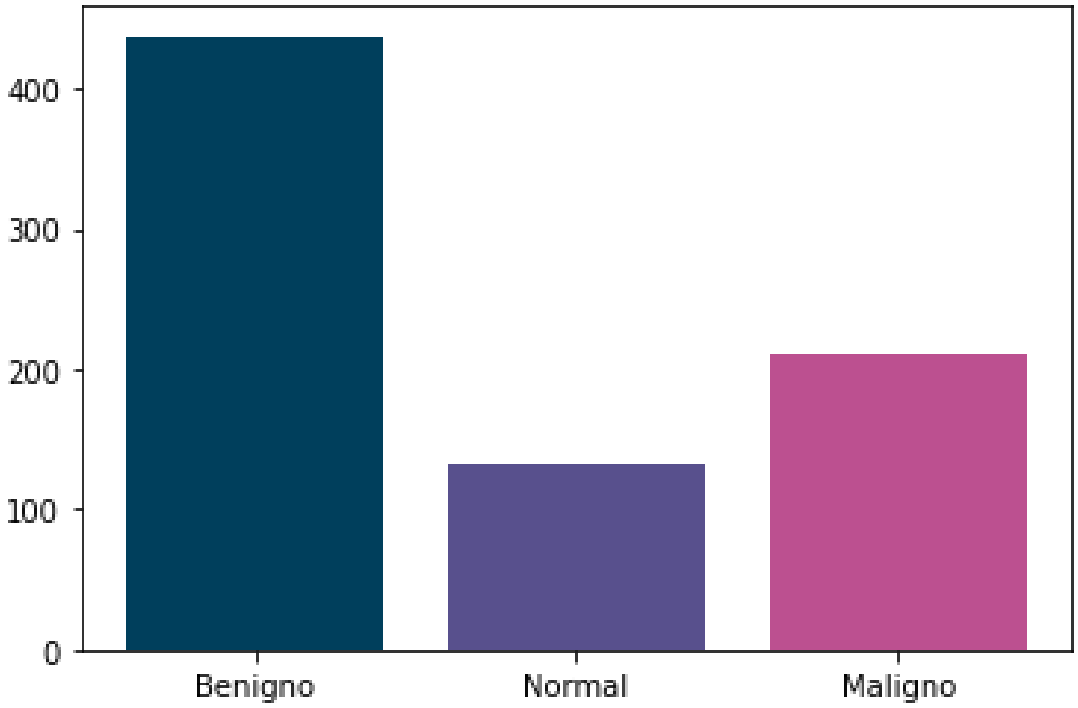


Figura 9: Cantidad de elementos por clases. Fuente: Imagen propia.

Por lo tanto, esto tiene como principal consecuente que nuestro modelo tenga una mayor sensibilidad a reconocer la clase benigno en el modelo de clasificación. Los resultados del modelo inicial se discuten en etapas posteriores del modelo CRISP-DM.

Esta proporción de cantidad de elementos en cada clase también se debe de reflejar al momento dividir el conjunto de datos en datos de entrenamiento, evaluación y prueba.

Una vez actualizado el conjunto de datos base con los casos añadidos luego de la primera fase de pruebas, la proporción de elementos en cada clase término de la siguiente manera:

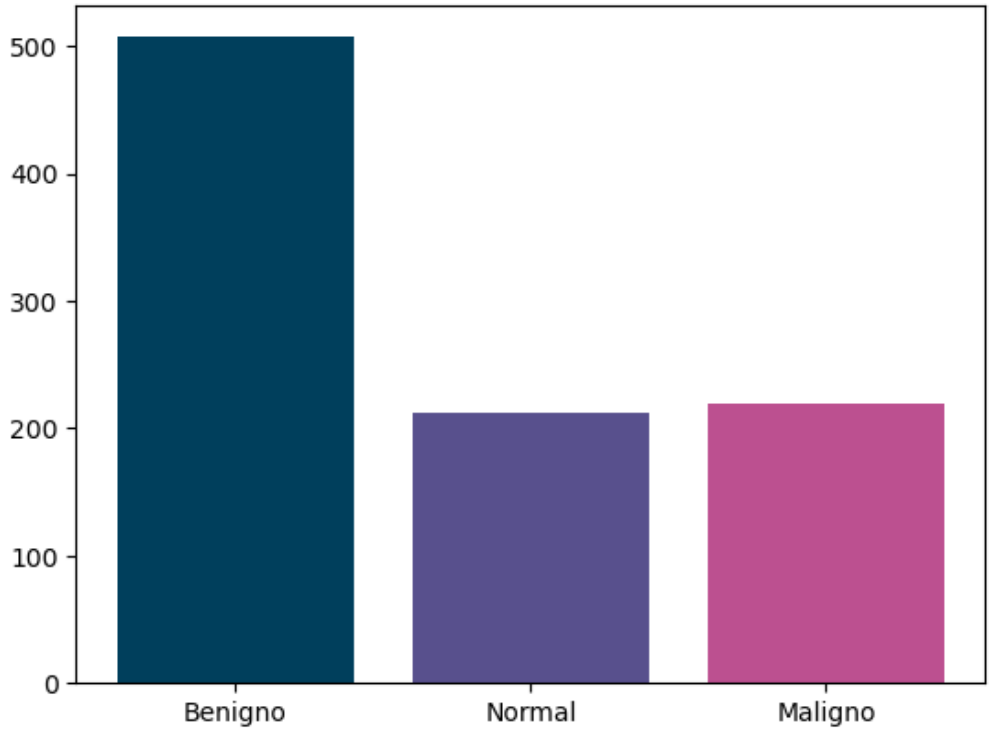


Figura 10: Cantidad de elementos por clases de conjunto de datos actualizado. Fuente: Imagen propia.

Donde podemos observar que hay un crecimiento de la clase menos representada, normal, pasando de tener 133 a 212 registros, de 437 a 507 registros para la clase benigno y de 210 a 219 registros para la clase maligno.

También es necesario:

4. Identificar posibles problemas o desafíos con los datos.

En este paso se realizó la detección de datos atípicos (o outliers), investigar posibles fuentes de sesgo o variabilidad en los datos, y determinar si se necesitan más datos (o si es necesario recopilar más información) para entrenar y evaluar el modelo de manera adecuada.

Algunos desafíos con los datos utilizados están relacionados con la forma en que se convirtieron los archivos de ultrasonidos en imágenes en formato PNG. Es posible que las técnicas y herramientas utilizadas no sea la misma que la empleada en este trabajo, y, por lo tanto, pudo darse algún tipo de perdida de información en el proceso.

También puede existir variabilidad en la representación de los datos. Por ejemplo, podría haber inconsistencias entre las representaciones de las características o ítems de información del conjunto de datos usado para el entrenamiento y el creado para la fase de prueba. En estos casos, se sugiere tener mucho cuidado en cuanto a asegurar la consistencia entre los formatos de las representaciones de características de los datasets utilizados.

Al momento de guardar las imágenes en formato DICOM en los contenedores de Azure, los archivos primero se deben de “serializar”, esto es, convertir las imágenes en bytes para que puedan ser enviadas por la red. Al momento de obtener las imágenes, se debe de seguir el mismo proceso, serializar los archivos que están en los contenedores y luego convertirlos al formato DICOM.

El conjunto de datos original está estructurado por las 3 clases: benigno, maligno y normal, a como se muestra en el Anexos 2. Sin embargo, este aún debe de ser estructurado para que pueda ser utilizado por los 2 modelos, clasificación y segmentación. El proceso que se llevó a cabo para formatear los datos correctamente se explica en el paso 2 de la etapa de preparación de los datos.

En resumen, la etapa de entendimiento de los datos en el modelo CRISP-DM, se refiere a la recopilación, limpieza y análisis de los datos que se utilizan como entrada para el modelo de aprendizaje automático, con el objetivo de comprender su contenido y su distribución, e identificar posibles problemas o desafíos.

6.3 Preparación de los datos

La etapa de preparación de los datos en el modelo CRISP-DM, se refiere a las tareas que se llevan a cabo para transformar y manipular los datos de entrada en un formato adecuado para su uso en el modelo de aprendizaje automático [43]. En el caso de un modelo de diagnóstico del cáncer de mama, esta etapa podría incluir primera tarea a realizar:

1. Seleccionar las características relevantes de los datos.

Se refiere a la selección de las variables más relevantes, a tomar en cuenta para el diagnóstico del cáncer de mama (por ejemplo, los resultados de un análisis de sangre, las características visuales de las imágenes de mamografía, etc.), y eliminar las características irrelevantes o redundantes.

Lo anterior, no aplica en este proyecto, debido a la naturaleza no estructurada de los datos. Sin embargo, una característica importante es la identificación de la zona afectada en la imagen de ultrasonido. Los datos de entrenamiento constan de un par (x, y). Donde x son las características utilizadas para determinar el valor de la variable objetivo: y. En este caso x, son las imágenes de entrada, e y, es la correspondiente segmentación objetivo.

Podría decirse que, las características relevantes son las imágenes del ultrasonido. Más específicamente, la zona en la imagen de ultrasonido correspondiente a la mama, evitando así las áreas relacionadas a la información complementaria que pudiese aparecer en las imágenes, como el nombre, anotaciones de los médicos, etc., ya que estas crean ruido al momento de realizar las predicciones.

Acto seguido, se debe:

2. Transformar los datos para mejorar su calidad.

Esto incluyó acciones como: normalizar los datos, para que tengan un rango de valores consistente; aplicar técnicas de procesamiento de señales, para mejorar la calidad de las imágenes de ecografía; o utilizar técnicas de ingeniería de

características, para generar nuevas propiedades a partir de las peculiaridades preexistentes.

Una transformación comúnmente utilizada en el procesamiento de imágenes es, por ejemplo, normalizar su rango de valores, de tal manera que, en lugar de tener un rango de valores enteros de 0 a 255 para cada canal, se utilice un rango numérico entre 0 y 1. Esto beneficia al proceso de entrenamiento, específicamente, en la etapa de optimización, ya que la misma incluye el cálculo del gradiente de la función de pérdida. Si se calcula utilizando un rango de valores más reducido, el proceso se acelera y el entrenamiento converge más rápido [48].

A continuación, será necesario:

3. Dividir los datos en conjuntos de entrenamiento, validación y pruebas.

La importancia de esta tarea radica en que posibilita entrenar el modelo de manera adecuada y evaluar su desempeño de manera justa y precisa [49]. Por lo general, se utiliza una división de los datos en un conjunto de entrenamiento (utilizado para entrenar el modelo), un conjunto de validación (utilizado para ajustar los parámetros del modelo), y un conjunto de pruebas (utilizado para evaluar el desempeño final del modelo).

En la primera etapa de este proyecto se siguió un enfoque tradicional, a como se mencionó en el párrafo anterior, de dividir el total del conjunto de datos en: datos de entrenamiento, validación y prueba. Esto lo podemos hacer con las utilidades del módulo Keras de Tensorflow, en Anexos 4 se muestra el código necesario para crear los conjuntos de datos.

Primeramente, se siguió una partición de forma aleatoria, es decir, sin tomar en cuenta el porcentaje de cada clase sobre el total del conjunto de datos. El conjunto de datos de prueba no se obtuvo a partir del conjunto de datos total, en su lugar las pruebas se realizaron a partir de nuevos casos de IXCHEN.

Finalmente, luego de hacer las pruebas y obtener los resultados de estas, se integró el conjunto de datos de pruebas al conjunto de datos total para reentrenar el modelo y así obtener una versión mejorada de este.

En etapas posteriores, luego de preparado el conjunto de datos para el reentrenamiento, se sigue un particionamiento “estratificado”, esto es, se seleccionan cada conjunto de datos de tal manera que la distribución original (el porcentaje de cada clase), es reflejado en cada conjunto de datos (entrenamiento y validación), añadiendo el dataset original de la universidad de Egipto y los casos recolectados en IXCHEN.

Para hacer esto, del conjunto de datos total (original e IXCHEN), se dividió en base al porcentaje de cada clase por respecto al total, el código completo para hacer esta división estratificada para los datos de entrenamiento y validación se muestra en Anexos 5.

Las pruebas finales de los modelos reentrenados se llevaron a cabo a partir de datos recolectados luego del periodo de prueba inicial, esto para asegurarnos de que registros utilizados durante el entrenamiento no sean recurrentes durante las pruebas y por lo tanto nos den resultados incorrectos.

Para preparar los datos para la prueba final, se siguió un proceso el cual consiste en los siguientes pasos:

- Obtener los archivos DICOM para cada usuario junto con su diagnóstico proveído por el medico radiólogo.
- Registrar cada archivo y su correspondiente diagnóstico.
- Realizar las predicciones para ambos modelos (AlexNet y ResNet-50).
- Registrar las predicciones para cada archivo.

De los anteriores pasos, cabe destacar que para obtener cada la clase verdadera (benigno, maligno o normal), el documento de diagnostico nos dice la categoría BI-RADS a la que pertenece, luego se traduce esta categoría a una de las 3 clases. En Anexos 6 se muestra la tabla utilizada para hacer esta traducción y la cual fue facilitada por médicos radiólogos de IXCHEN.

En resumen, la etapa de preparación de los datos en el modelo CRISP-DM, se refiere a las tareas que se llevan a cabo para transformar y manipular los datos de entrada en un formato adecuado para su empleo en el modelo de aprendizaje

automático, con el objetivo de mejorar su calidad y prepararlos para su uso en el entrenamiento y la evaluación del modelo.

6.4 Modelado

La etapa de modelado en CRISP-DM, se refiere a las tareas que se llevan a cabo para entrenar y evaluar el modelo de aprendizaje automático [43]. En el caso de un modelo de diagnóstico del cáncer de mama, esta etapa incluye, como primera tarea:

1. Seleccionar el tipo de modelo de aprendizaje automático que se utilizará.

Resulta conveniente indagar sobre diferentes tipos de modelos que pudieran ser utilizados (por ejemplo, modelos de clasificación, de regresión, basados en árboles de decisión, etc.) y seleccionar el que mejor se adapte a los datos disponibles para la solución del problema.

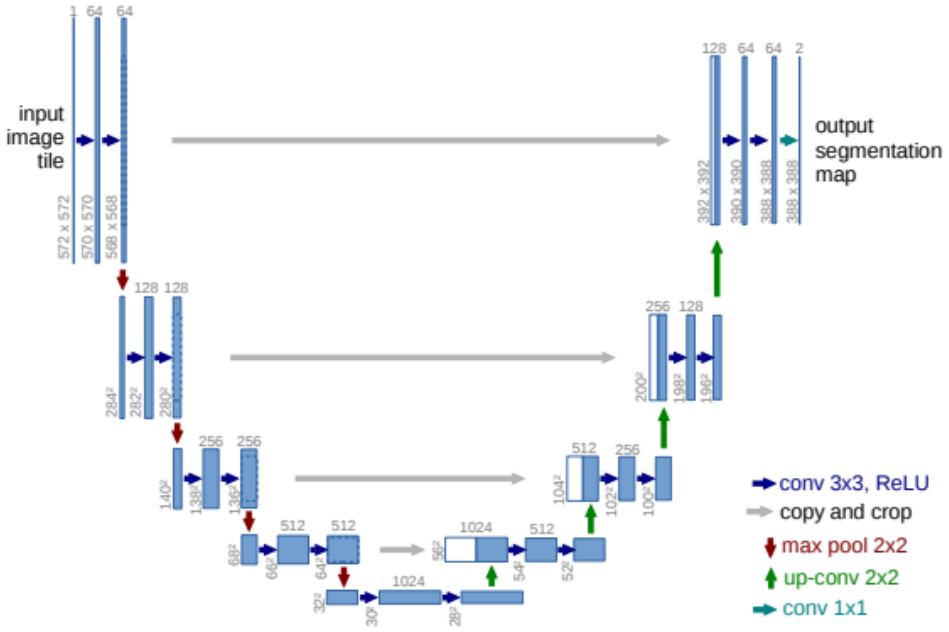


Figura 11: Arquitectura modelo U-Net [8]

En el caso del modelo de segmentación, este sigue la arquitectura U-Net, la cual se consideró de utilidad para segmentación semántica. Esta consiste en un

camino de contracción para capturar el contexto y un camino simétrico de expansión que permite una localización precisa del objeto que se trata de segmentar. En [8], se menciona que la red puede ser entrenada, de inicio a fin, con pocas imágenes (como en este caso de estudio) y obtener resultados satisfactorios. En la imagen anterior (figura 11), se muestra una descripción genérica de la arquitectura utilizada. En ella se puede apreciar, el camino inicial de contracción, y luego, el camino de expansión.

En la primera iteración del modelado se construyó la arquitectura U-Net con TensorFlow y Keras. En Anexos 7, se muestra el código necesario para poder implementar esta arquitectura. Sin embargo, diferentes backends y librerías de deep learning ofrecen una “plantilla” para poder utilizar esta arquitectura y en ocasiones nos dan la posibilidad de obtener una versión pre-entrenada de esta arquitectura con un conjunto de datos de acceso libre.

A continuación, el trabajo se enfoca en dar una explicación más a fondo de la arquitectura U-Net y cada uno de sus componentes. En este caso, puede ser vista como un tipo de autoencoder convolucional con algunas modificaciones a una arquitectura de autoencoder estándar. Un autoencoder consiste en dos partes: un codificador y decodificador. El codificador asigna los datos de entrada a una representación dimensional inferior, mientras que el decodificador asigna la representación dimensional inferior de vuelta a los datos originales. En el caso de U-Net, la parte del codificador es el camino de contracción y la parte del decodificador es el camino de expansión.

En un autoencoder estándar, el objetivo es reconstruir los datos originales con la mayor precisión posible, por lo que la función objetivo es minimizar la pérdida de reconstrucción entre los datos de entrada y la salida del decodificador. En U-Net, el objetivo no es solo reconstruir la imagen original, sino que también realizar una segmentación semántica, por lo que la capa final de la red produce un mapa de segmentación donde a cada píxel se le asigna una clase.

La arquitectura U-Net también usa conexiones de salto (skip connections), que no están presentes en los autoencoders estándares [50]. Estas conexiones de salto

ayudan a preservar las características de alto nivel de la ruta de contracción durante el proceso muestreo ascendente, lo que permite que el modelo aprenda detalles más finos en la imagen.

De esta manera, mientras que U-Net toma prestada la estructura general de un autoencoder, está diseñado para manejar la tarea especifica de segmentación semántica y usa una función objetivo diferente y modificaciones adicionales como las conexiones de salto.

Una conexión de salto (skip connection) es un atajo que permite que la señal de gradiente pase por alto una o más capas en la red y conecte directamente la entrada a la salida, o viceversa. En la arquitectura U-Net, las conexiones de salto se utilizan para pasar información desde la ruta de contracción de la red (la parte del codificador) a la ruta de expansión (la parte del decodificador). La ruta de contratación reduce la resolución de la imagen de entrada para extraer características, pero al mismo tiempo, también reduce la resolución espacial de los mapas de características. La ruta de expansión, por otro lado, aumenta la muestra de los mapas de características para aumentar su resolución espacial y producir el resultado final.

Al agregar las conexiones de salto, U-Net puede preservar la información espacial perdida durante el proceso de submuestreo y usarla para informar el proceso de muestreo ascendente en la ruta de expansión. Esto permite que la red recupere detalles específicos en la salida que, de otro modo, se perderían.

A continuación, se presenta los pasos a seguir para implementar conexiones de salto en proyectos similares, donde podremos preservar la información espacial en toda la red, son los siguientes:

- Identificar las capas de la red donde se desea preservar la información espacial.
- Crear conexiones de salto entre estas capas para proporcionar un atajo para que fluya la señal del gradiente.
- Entrenar la red para que aprenda relaciones más complejas entre la entrada y la salida.

En resumen, las conexiones de salto ayudan a preservar la información espacial en toda la red y proporcionan un atajo para que fluya la señal de gradiente, lo que permite que la red aprenda relaciones más complejas entre la entrada y la salida. Esto da como resultado un mejor rendimiento en tareas como la segmentación de imágenes y la segmentación semántica.

Para la segunda iteración, en el caso del modelo se segmentación, debido a los resultados obtenidos durante las pruebas se decidió no hacer un re-entrenamiento de este modelo. En su lugar se optó por seguir obteniendo más registros, de tal manera que el aumento de la precisión en la segmentación sea considerable.

Para el caso del modelo de clasificación, se mencionó anteriormente, que está compuesto por 2 partes: un extractor de características y un clasificador que utiliza las características anteriormente obtenidas para clasificar una imagen en benigno, maligno o normal.

En la primera iteración, se desarrolló una red neuronal la cual sigue un modelo secuencial que consta de varias capas. Primero, se aplica una capa de aumento de datos para mejorar el rendimiento del modelo. Luego, se aplica una capa de re-escalado para normalizar los valores de los pixeles de la imagen. Después, hay tres capas convolucionales con funciones de activación ReLU y capas de agrupación máxima entre ellas para reducir la dimensionalidad de los datos. Luego, se aplica una capa de dropout para prevenir el sobreajuste. Después, se aplana la salida y se una capa densa con función de activación ReLU. Finalmente, se aplica otra capa densa para producir la salida del modelo.

A continuación, se muestra el código de este modelo:

```
model = tf.keras.Sequential([
  data_augmentation,
  tf.keras.layers.Rescaling(1./255),
  tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes)
])
```

Figura 12: Arquitectura inicial modelo de clasificación. Fuente: Imagen propia.

En siguientes iteraciones se sigue una arquitectura AlexNet, esta fue diseñada por Alex Krizhevsky. AlexNet compitió en el Desafio de Reconocimiento Visual a Gran Escala de ImageNet (ImageNet Large Scale Visual Recognition Challenge) el 30 septiembre del 2012. La red logro un error top-5 del 15.3%, más de 10.8 puntos porcentuales por debajo del segundo lugar [51].

Esta arquitectura consta de ocho capas en total, de las cuales las primeras cinco son capas convolucionales y las últimas tres son completamente conectadas (fully connected). Las primeras dos capas convolucionales están conectadas a capas de agrupación máximas (max pooling) superpuestas para extraer el máximo número de características [52]. Al final de cada capa, se realiza la activación ReLU, excepto en la última, que se produce con una softmax con una distribución sobre las 1000 etiquetas de clase, sin embargo, para el caso de este proyecto la cantidad de etiquetas es de 3.

Aquí se muestra una imagen de la arquitectura:

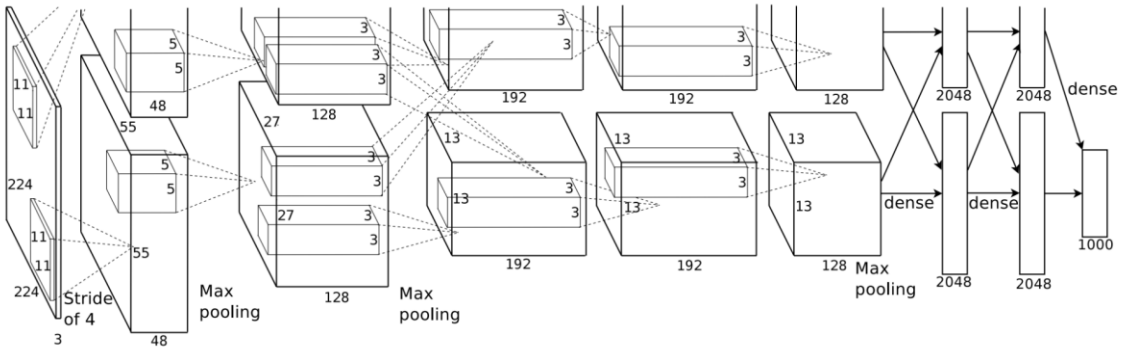


Figura 13: Arquitectura AlexNet [51]

Adicional a la arquitectura AlexNet, también se hicieron pruebas con la arquitectura ResNet-50, la cual es una red neuronal convolucional de 50 capas que utiliza un marco de aprendizaje residual profundo para permitir el entrenamiento de redes muy profundas con cientos de capas. Esta utiliza un diseño de cuello de botella para el bloque de construcción, que reduce el número de parámetros y multiplicación de matrices mediante el uso de convoluciones 1x1, conocidas como “cuellos de botella” [53]. Esto permite un entrenamiento mucho más rápido de cada capa.

En la siguiente imagen se puede ver un esquema de esta arquitectura:

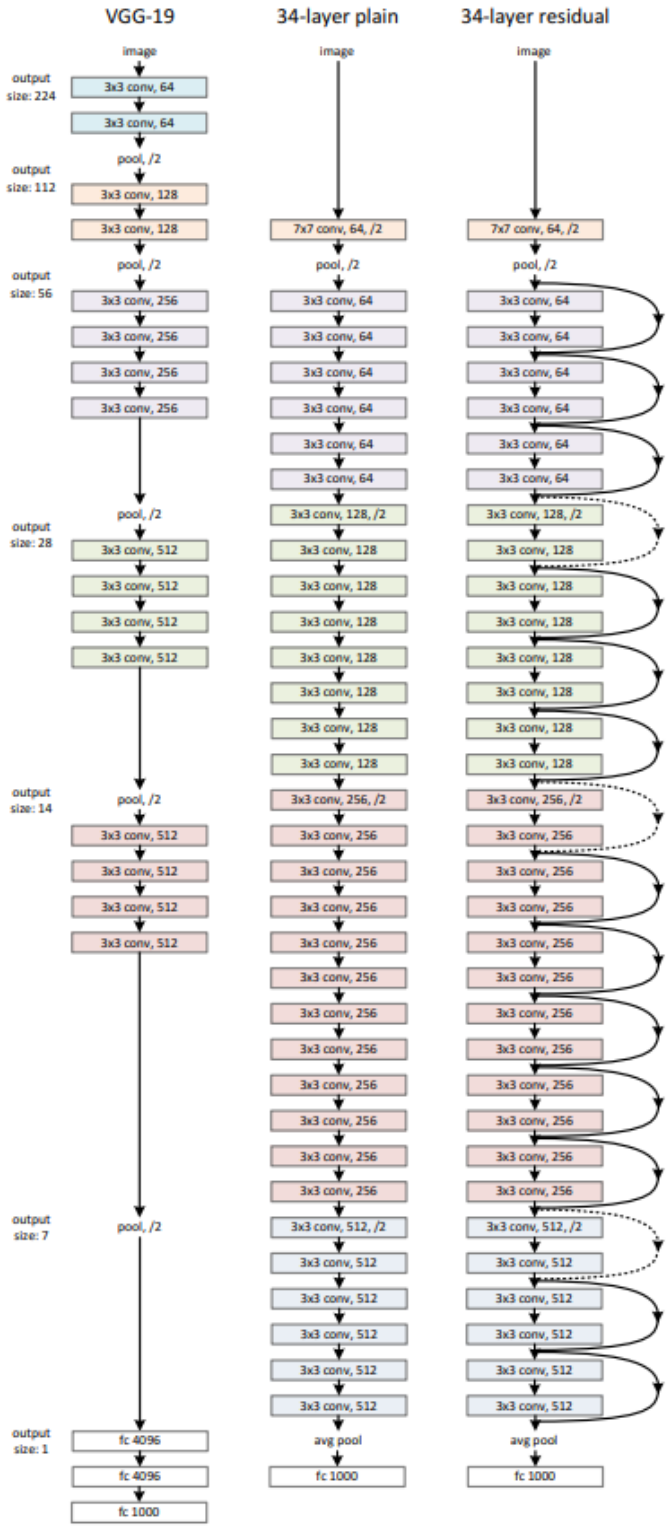


Figura 14: Arquitectura ResNet-50 [50]

Las siguientes tareas por realizar son:

2. Configurar y entrenar el modelo.

En este punto, se podría incluir la definición de los parámetros del modelo (por ejemplo, la tasa de aprendizaje, la complejidad del modelo, etc.). También se utiliza el conjunto de entrenamiento para entrenar el modelo, y ajustar los parámetros del modelo utilizando el conjunto de validación.

El entrenamiento de un modelo de aprendizaje automático es un proceso iterativo que busca minimizar el error de predicción. Este proceso se realiza ajustando los parámetros del modelo mediante técnicas de optimización, como el descenso de gradiente, utilizando un conjunto de datos de entrenamiento.

Aplicado a nuestro proyecto, esto incluye el ajuste de los hiperparámetros necesarios para configurar el entrenamiento y desempeño de los modelos desarrollados. Estos hiperparámetros son diferentes para ambos tipos de modelos desarrollados, ya que estos dependen de la arquitectura especifica a implementar.

En ambos modelos se pueden modificar los siguientes hiperparámetros, en caso de que solo aplique a un tipo de modelo, esto será especificado:

- Tamaño de filtro: valores más grandes pueden capturar mayor contexto, mientras que filtros más pequeños pueden ser computacionalmente más eficientes.
- Numero de filtros: esto afecta a la capacidad de la red para aprender representaciones complejas.
- Numero de capas: el número de capas en las porciones de codificador y decodificador de la red se puede ajustar.
- Tamaño stride: valores más grandes reducen la resolución espacial de los mapas de características, mientras que valores de stride más pequeños pueden preservar más información espacial. En este trabajo se escogió un valor de 1.

- Tamaño agrupación (pooling): el tamaño de la operación de agrupación en la parte del codificador de la red se puede ajustar. Esto afecta la reducción de la resolución y la invariancia espaciales de las características.
- Funciones de activación no lineales: se refiere a la función de activación utilizada en cada capa. Las siguientes son comúnmente utilizadas: ReLU, leaky ReLU y sigmoide. En este trabajo fue utilizada la función de activación ReLU.
- Función de perdida: para el modelo de segmentación, cada píxel debe de clasificarse como perteneciente a una de dos clases, segmentado o no. Por lo tanto, tenemos una clasificación binaria y en estos casos, la función de perdida por defecto es llamada binary cross-entropy. En el caso del modelo de clasificación, cada imagen debe clasificarse en una de 3 clases posibles, y es comúnmente utilizada la función categorical cross-entropy.
- Optimizador: en este trabajo se utilizó el optimizador Adam, el cual es comúnmente utilizado y da buenos resultados.
- Tamaño de lote (batch size): tamaños de lote más largos puede incrementar la rapidez de entrenamiento, mientras que tamaños más pequeños puede conducir a un entrenamiento más estable.
- Numero de etapas: una etapa se refiere a una iteración completa del conjunto de datos de entrenamiento durante el entrenamiento del modelo, más épocas pueden dar como resultado un mejor rendimiento, pero también incrementa el riesgo de sobreajuste. El modelo de segmentación fue entrenado por 50 etapas y el modelo de clasificación por 20 etapas.

Para la segunda fase, en el modelo de clasificación se siguió un proceso de fine-tuning utilizando para esto la librería FastAI la cual nos provee de varias utilidades para poder importar conjuntos de datos en diferentes formatos, al igual que arquitecturas previamente entrenadas.

En Anexo 8 se muestra el código necesario para hacer el proceso de fine-tuning sobre ambas arquitecturas. Este proceso se puede reducir en los siguientes pasos:

- Se importan las librerías necesarias, incluyendo pathlib, fastai.vision.all y torch.
- Se vacía la memoria caché CUDA para liberar espacio en la GPU.
- Se crea un objeto ImageDataLoaders a partir de una carpeta que contiene imágenes de entrenamiento y validación. Las imágenes se redimensionan a 256 pixeles y se utiliza un tamaño de lote de 8.
- Se crea un objeto vision_learner utilizando el modelo pre-entrenado AlexNet y ResNet-50 y se especifican las métricas a utilizar (error_rate y accuracy).
- Se realiza un fine-tuning del modelo durante 10 iteraciones.
- Se muestran los resultados del modelo en 20 imágenes.
- Finalmente, se exporta el modelo entrenado a un archivo con nombre especifico de la arquitectura utilizada en formato pkl.

Este proceso se repitió para ambas arquitecturas: AlexNet y ResNet-50.

3. Evaluar el desempeño del modelo.

Resulta factible utilizar el conjunto de pruebas para evaluar el desempeño del modelo (por ejemplo, calcular la tasa de precisión, la tasa de falsos positivos, etc.). Se compara el desempeño del modelo con el de otros modelos o herramientas de diagnóstico del cáncer de mama, y se analiza los resultados de este con diferentes subgrupos de datos (por ejemplo, en función de la edad de los usuarios).

En este proyecto, en la primera fase, se evaluó el desempeño del modelo en varios pasos. Primero, como parte del proceso de entrenamiento, internamente, el algoritmo compara el resultado producido por el modelo con el que debería de ser y lo ajusta los parámetros internos en base al resultado de esta comparación. Una estrategia comúnmente utilizada es dividir el total del conjunto de datos en 3:

- Datos de entrenamiento: utilizado para ajustar los parámetros internos del modelo.
- Datos de validación: utilizado para ver el comportamiento del algoritmo durante el entrenamiento. Sin embargo, el resultado del error de las

predicciones no es utilizado para ajustar los parámetros del algoritmo, en su lugar, es utilizado para, una vez terminado el entrenamiento, cambiar el valor de los hiperparámetros de tal forma que las predicciones mejoren en este conjunto de datos.

- Datos de prueba: habiendo terminado el entrenamiento y ajustado los hiperparámetros para obtener mejores predicciones en los datos de validación. Realizamos una prueba final sobre los datos de prueba que fueron realizadas sobre datos de usuarios nuevos.

Generalmente, es una buena práctica el dividir el conjunto de datos en las siguientes porciones: 80% para datos de entrenamiento, 10% para datos de validación y el 10% restante para los datos de prueba. Sin embargo, un problema que debió valorarse es la poca cantidad de datos disponible. Por lo tanto, al dividir el conjunto de datos se evitó que el modelo tenga una mayor base de conocimiento de la cual aprender.

En la fase inicial, se evaluó el modelo de aprendizaje sobre los datos generados por IXCHEN, Luego, por un proceso que se detallará posteriormente, se recibió retroalimentación por parte del personal de radiología encargado de probar el modelo con nuevos usuarios del centro.

En la segunda fase, durante el proceso de fine-tuning para el modelo AlexNet se obtuvo una precisión del 84% sobre los datos de validación y del 83.5% para el modelo ResNet-50, con un tiempo de ajuste menor para arquitectura AlexNet. A pesar de que la arquitectura ResNet-50 cuenta con un mayor número de parámetros, hay varias razones por las que AlexNet podría haber tenido un mejor rendimiento que ResNet-50 en nuestros datos.

Una posible razón puede ser que el conjunto de datos puede ser más adecuado para una arquitectura más simple como AlexNet. Por ejemplo, nuestro conjunto de datos tiene menos de 1000 elementos, lo cual es relativamente pequeño y las características pueden ser fáciles de extraer y una arquitectura más simple como AlexNet podría ser suficiente para lograr un buen rendimiento [54].

Una razón para que el proceso de fine-tuning para la arquitectura AlexNet haya sido más rápido que ResNet-50 podría deberse a que AlexNet tiene menos capas y parámetros que ResNet-50, lo que hace que el proceso de fine-tuning sea más rápido. Sin embargo, en general, ResNet-50 puede tener un mayor rendimiento en otro tipo de tareas con un número mayor de datos.

4. Optimizar el modelo.

En este caso, se busca como mejorar el desempeño del modelo (por ejemplo, utilizando técnicas de regularización, añadiendo más datos para entrenar el modelo, o utilizando técnicas de enriquecimiento de datos), y validar los resultados obtenidos por el modelo optimizado, utilizando un nuevo conjunto de pruebas.

Luego de la fase de pruebas iniciales, el modelo es ajustado y modificado. Para ello, se utilizan los datos que fueron usado para pruebas. En resumen, la etapa de modelado en CRISP-DM, se refiere a las tareas que se llevan a cabo para entrenar y evaluar el modelo de aprendizaje automático, con el objetivo de obtener una versión mejorada u optimizada del mismo, de manera que se logre un mejor desempeño, en términos de la tarea que se está realizando (en este caso, el diagnóstico del cáncer de mama).

6.5 Evaluación

La etapa de evaluación en el modelo CRISP-DM se refiere a las tareas que se llevan a cabo para examinar y evaluar el desempeño del modelo de aprendizaje automático en relación con los objetivos y las metas establecidos en la etapa de entendimiento del negocio [43]. En el caso de un modelo de diagnóstico del cáncer de mama, esta etapa podría incluir tareas como:

1. Calcular medidas de desempeño del modelo.

Se incluyen acciones como el cálculo de la tasa de precisión del modelo (es decir, la proporción de diagnósticos correctos), la tasa de falsos positivos (es decir, la

proporción de diagnósticos erróneos), y otras medidas que se hayan definido en la etapa de entendimiento del negocio.

Para el caso del modelo de segmentación, el modelo se entrenó por 50 iteraciones dando un resultado de 98% de precisión y una perdida cercana a 0, a como se muestra en la figura 15.

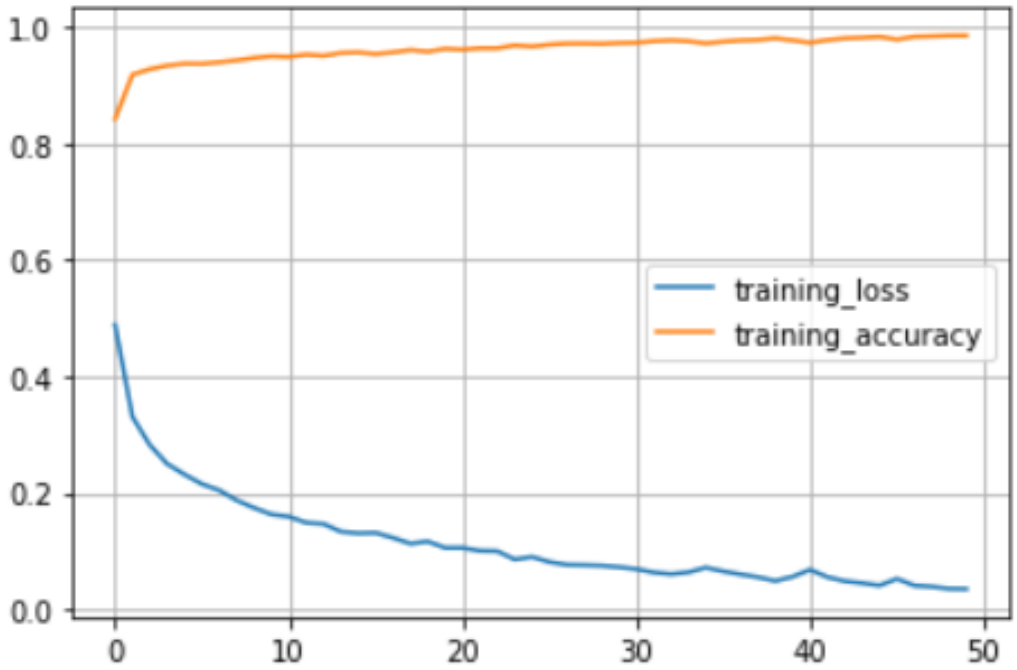


Figura 15: Resultados del entrenamiento de modelo de segmentación. Fuente: Imagen propia.

Durante el periodo de pruebas, se obtuvo una precisión del 85%, siendo esta calculada por medio de la media aritmética de todas las precisiones individuales ajustadas por el medico radiólogo, de un total de 256 pruebas.

Para el modelo de clasificación, se entrenó por 15 iteraciones y se obtuvo una precisión del 71% y del 72% en los datos de validación. En la figura 16 se muestra una imagen con los resultados obtenidos durante el entrenamiento.

En la siguiente figura, se observa como al continuar el entrenamiento se aumenta la precisión y se reduce la perdida, de igual manera se observan fluctuaciones en

estas medidas, lo cual es un indicio de que el modelo puede estar sobre ajustando (overfitting) el modelo a los datos de entrenamiento, por lo tanto, una estrategia es detener el proceso de entrenamiento de forma temprana y ver cómo se comporta durante los datos de prueba.

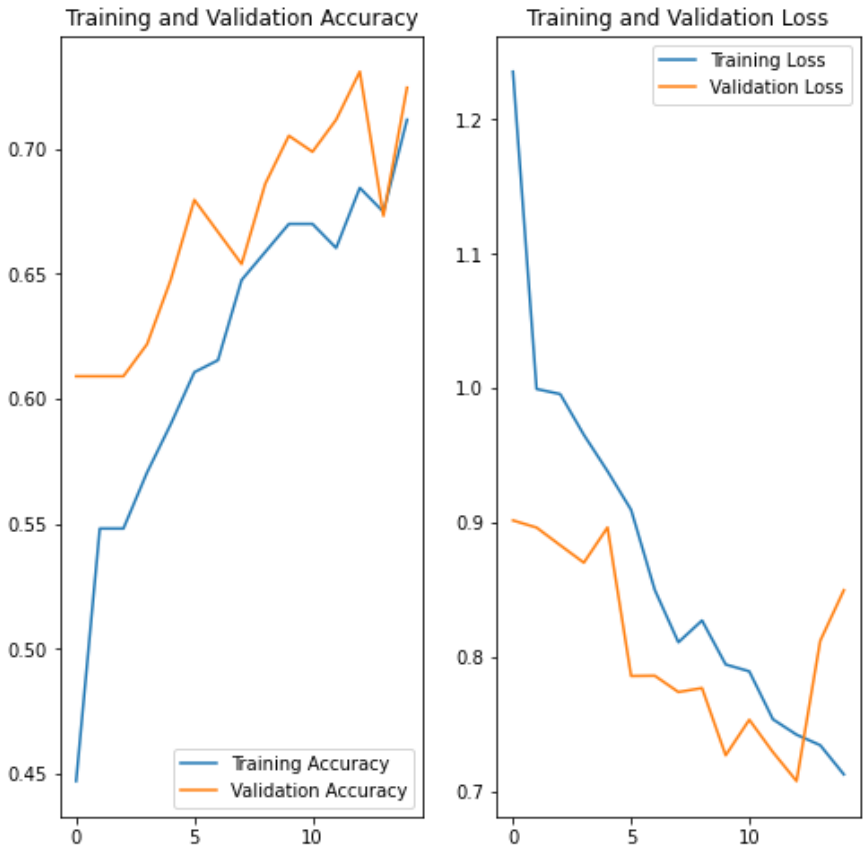


Figura 16: Resultados del entrenamiento de modelo de clasificación. Fuente: Imagen propia.

En la figura 17 se puede observar la matriz de confusión que se obtuvo a partir de los datos de prueba. Para entender mejor los resultados, se puede hacer uso del reporte de clasificación que proporciona la librería scikit learn (figura 18) el cual muestra varias métricas para evaluar el desempeño de un modelo de clasificación en la predicción de las tres clases: benigno (0), maligno (1) y normal (2).

Según el reporte, el modelo tiene una precisión del 46% para la clase benigno, 12% para la clase maligno, 68% para la clase normal. En cuanto al recall, el modelo tiene un valor del 85% para la clase benigno, 7% para la clase maligno y 11% para la clase normal. El puntaje F1, que combina precisión y recall, es del 60% para la clase benigno, 9% para la clase maligno y 19% para la clase normal.

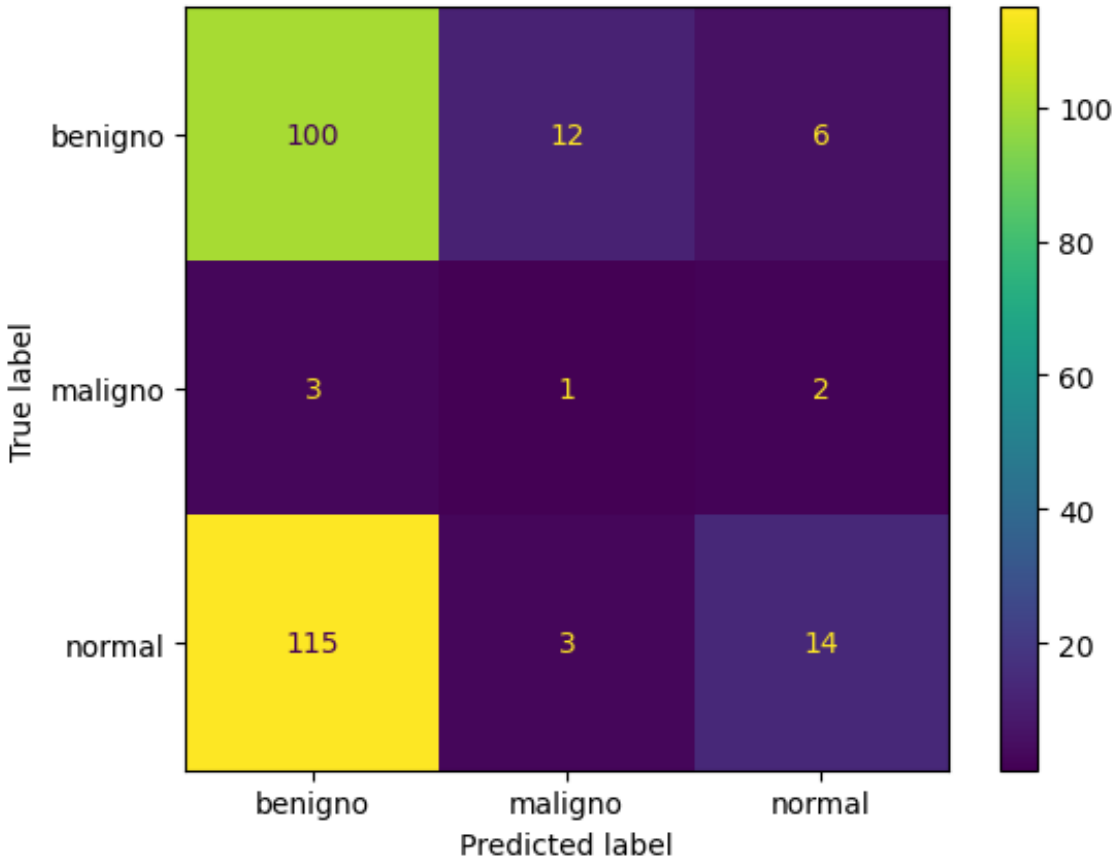


Figura 17: Matriz de confusión modelo de clasificación. Fuente: Imagen propia.

```
[38]  1  print(classification_report(y_true=y_true, y_pred=y_pred))
```

```
...                precision    recall  f1-score   support

             0       0.46      0.85      0.60       118
             1       0.12      0.07      0.09        28
             2       0.68      0.11      0.19       136

     micro avg       0.46      0.41      0.43       282
     macro avg       0.42      0.34      0.29       282
  weighted avg       0.53      0.41      0.35       282
   samples avg       0.46      0.41      0.43       282
```

Figura 18: Reporte de modelo de clasificación. Fuente: Imagen propia.

En general, el primer modelo implementado, mostró un desempeño aceptable en la predicción de clase benigno, con una predicción y recall relativamente altos, a como se muestra un resumen de los resultados en la tabla 1. Sin embargo, el desempeño del modelo en la predicción de clases maligno y normal fue poco satisfactorio, con valores bajos tanto en precisión como en recall. Esto sugiere que el modelo podría mejorarse en la predicción de estas clases.

Por tal motivo, a partir de estos resultados, se buscaron alternativas de mejora del modelo, optando por su reentrenamiento utilizando las arquitecturas AlexNet y ResNet-50. Una vez reentrenado, se volvieron a realizar predicciones sobre nuevos datos de prueba. Los resultados obtenidos, mejoraron considerablemente, al punto de alcanzar un resultado dentro del rango de excelencia, similares a los resultados de estudios referidos [6].

Clase	**Precision**	**Recall**	**Puntaje F1**
Benigno	46%	85%	60%
Maligno	12%	7%	9%
Normal	68%	11%	19%

Tabla 3: Resumen resultados modelo de clasificación

2. Comparar el desempeño del modelo con el de otros modelos o herramientas de diagnóstico del cáncer de mama.

Es recomendable comparar el desempeño del modelo con el de modelos desarrollados por otros investigadores, o con el de herramientas comerciales disponibles en el mercado.

3. Identificar posibles áreas de mejora del modelo.

Podría ser de interés para el proyecto, indagar sobre posibles fuentes de error o sesgo en el modelo, y proponer formas de mejorar el desempeño del modelo (por ejemplo, utilizando técnicas de regularización, añadiendo más datos para entrenar el modelo, o utilizando técnicas de enriquecimiento de datos).

En general, el modelo parece tener un desempeño aceptable en la predicción de la clase benigno, con una precisión y un recall relativamente altos. Sin embargo, el desempeño del modelo en la predicción de las clases maligno y normal es bajo, con valores bajos tanto en precisión como en recall. Esto sugiere que el modelo podría mejorarse en la predicción de estas clases.

Algunas posibles áreas de mejora incluyen:

- Ajustar los parámetros del modelo para mejorar su capacidad de distinguir entre las clases maligno y normal.
- Proporcionar más datos de entrenamiento para estas clases, especialmente para casos difíciles o ambiguos.
- Explorar diferentes arquitecturas de redes neuronales que puedan ser más adecuados para este problema en particular.

El modelo de segmentación tiene una precisión del 85%, puede considerarse que tiene un buen desempeño. Sin embargo, para este caso podría mejorarse aún más al incluir los nuevos datos en su reentrenamiento. Esto es especialmente importante ya que los nuevos datos representan casos que el modelo no ha visto antes o reflejan cambios en las condiciones en las que se aplicara el modelo.

Para el modelo de clasificación, se realizó un proceso de fine-tuning, detallado en la etapa de modelado. Se evaluó el desempeño de los modelos entrenados sobre datos de prueba recolectados luego del periodo de prueba inicial.

En la figura 19 se muestra la matriz de confusión para el modelo AlexNet y en la figura 20 para el modelo ResNet-50.

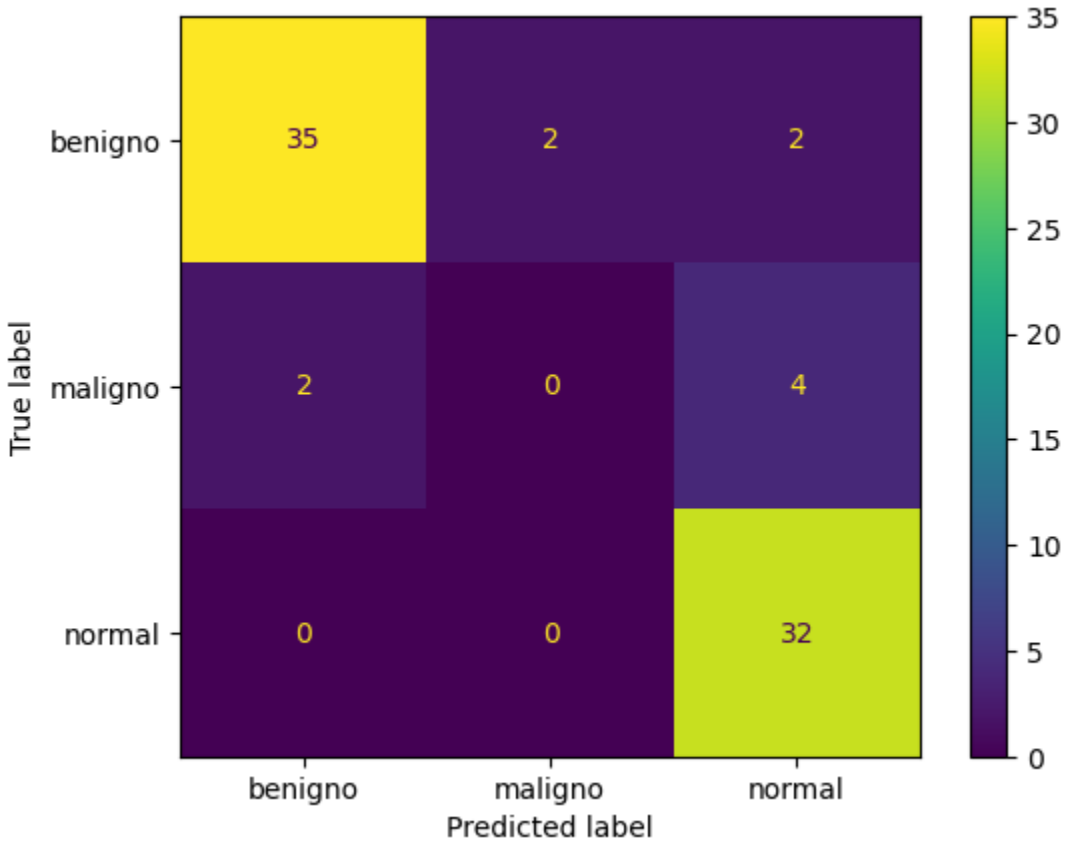


Figura 19: Matriz de confusión modelo AlexNet. Fuente: Imagen propia.

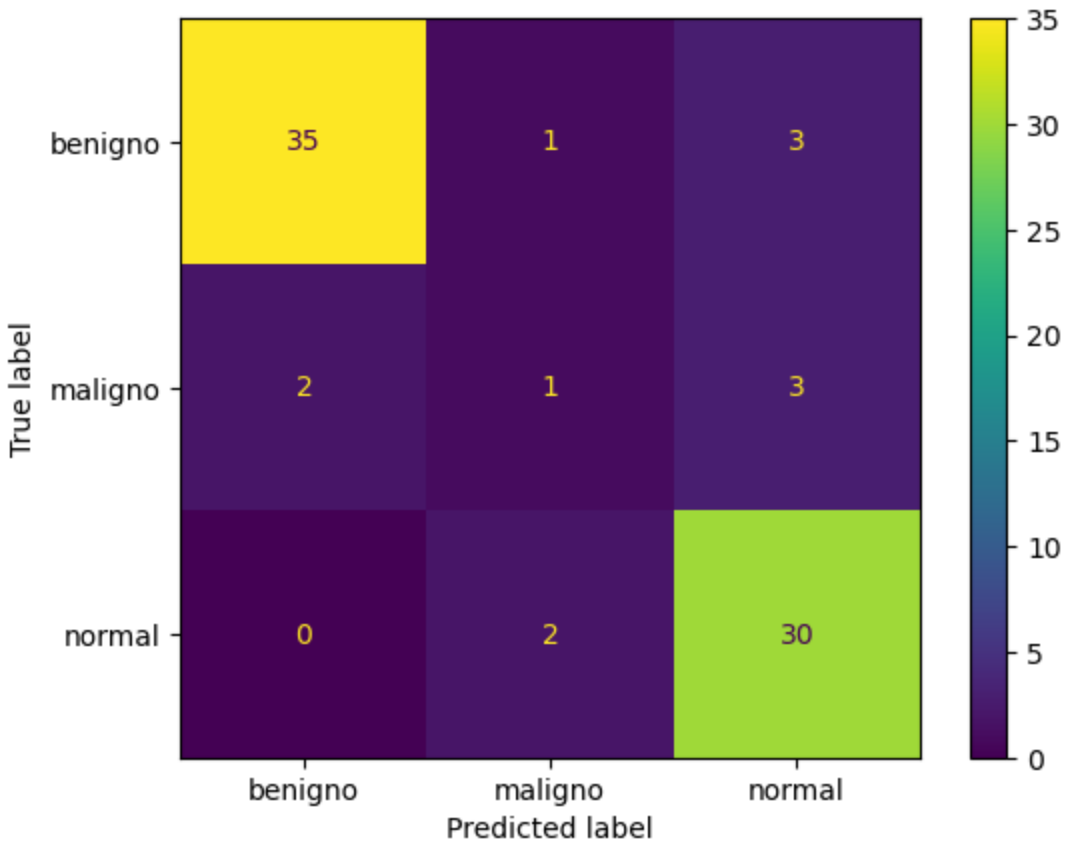


Figura 20: Matriz de confusión modelo ResNet-50. Fuente: Imagen propia.

De igual manera, para explicar los resultados de ambas matrices de confusión hacemos uso del reporte de clasificación para la arquitectura AlexNet (figura 21) y ResNet-50 (figura 22).

Para el modelo AlexNet, teniendo en cuenta que las clases son benigno (0), maligno (1) y normal (2):

- Para la clase **benigno**, el modelo tiene una precisión del 95%, lo que significa que el 95% de las instancias que el modelo predijo como benignas eran realmente benignas. El recall es del 90%, lo que significa que el modelo identificó correctamente el 90% de las instancias benignas. El puntaje F1, que es una medida que combina precisión y recall, es del 92%.
- Para la clase **maligno**, tanto la precisión como el recall y el puntaje F1 son 0. Esto sugiere que el modelo no pudo identificar correctamente ninguna instancia maligna.

- Para la clase **normal**, la precisión es del 84% y el recall es del 100%, lo que significa que el modelo identificó correctamente todas las instancias normales, pero también predijo incorrectamente algunas instancias como normales. El puntaje F1 es del 91%.

Para el modelo ResNet-50:

- Para la clase **benigno**, el modelo tiene una precisión del 95%, lo que significa que el 95% de las instancias que el modelo predijo como benignas eran realmente benignas. El recall es del 90%, lo que significa que el modelo identificó correctamente el 90% de las instancias benignas. El puntaje F1, que es una medida que combina precisión y recall, es del 92%.
- Para la clase **maligno**, la precisión es del 25% y el recall es del 17%, lo que significa que el modelo identificó correctamente el 17% de las instancias malignas y el 25% de las instancias que el modelo predijo como malignas eran realmente malignas. El puntaje F1 es del 20%.
- Para la clase **normal**, la precisión es del 83% y el recall es del 94%, lo que significa que el modelo identificó correctamente el 94% de las instancias normales y el 83% de las instancias que el modelo predijo como normales eran realmente normales. El puntaje F1 es del 88%.

Los resultados los modelos AlexNet y ResNet-50 pueden ser resumidos en la tabla 2 y 3 respectivamente.

Clase	**Precisión**	**Recall**	**Puntaje F1**
Benigno	95%	90%	92%
Maligno	0%	0%	0%
Normal	84%	100%	91%

Tabla 4: Resultado de clasificación de modelo AlexNet.

Clase	Precisión	Recall	Puntaje F1
Benigno	95%	90%	92%
Maligno	25%	17%	20%
Normal	83%	94%	88%

Tabla 5: Resultado de clasificación de modelo ResNet-50.

En resumen, ambos modelos tuvieron un buen desempeño en la predicción de clases benigno y normal, pero tuvieron dificultades para identificar correctamente las instancias malignas, siendo superior el modelo ResNet-50 al poder reconocer cierto porcentaje de esta clase. Esto sugiere que se podría mejorar el entrenamiento de ambos modelos para esta clase en particular. Una forma de hacerlo, es obteniendo más registros por un tiempo mayor al llevado a cabo en este estudio.

```
1  print(classification_report(y_true=y_true, y_pred=y_pred_alexnet))
```

```
              precision    recall  f1-score   support

           0       0.95      0.90      0.92        39
           1       0.00      0.00      0.00         6
           2       0.84      1.00      0.91        32

   micro avg       0.87      0.87      0.87        77
   macro avg       0.60      0.63      0.61        77
weighted avg       0.83      0.87      0.85        77
 samples avg       0.87      0.87      0.87        77
```

Figura 21: Reporte de clasificación de modelo AlexNet. Fuente: Imagen propia.

```
1  print(classification_report(y_true=y_true, y_pred=y_pred_resnet))
```

	precision	recall	f1-score	support
0	0.95	0.90	0.92	39
1	0.25	0.17	0.20	6
2	0.83	0.94	0.88	32
micro avg	0.86	0.86	0.86	77
macro avg	0.68	0.67	0.67	77
weighted avg	0.84	0.86	0.85	77
samples avg	0.86	0.86	0.86	77

Figura 22: Reporte de clasificación de modelo ResNet-50. Fuente: Imagen propia.

La etapa de evaluación en el modelo CRISP-DM se refiere a las tareas que se llevan a cabo para examinar y evaluar el desempeño del modelo de aprendizaje automático en relación con los objetivos y las metas establecidos en la etapa de entendimiento del negocio, con el objetivo de determinar si el modelo tiene un desempeño adecuado y si hay áreas en las que se puede mejorar.

6.6 Despliegue

La etapa de despliegue en el modelo CRISP-DM, se refiere a las tareas que se llevan a cabo para poner el modelo de aprendizaje automático en producción y hacerlo disponible para su uso por parte de los usuarios [43]. En el caso de un modelo de diagnóstico del cáncer de mama, esta etapa incluye las siguientes tareas:

1. Crear una interfaz de usuario para el modelo.

Es conveniente diseñar una interfaz gráfica de usuario (GUI) que permita a los usuarios ingresar los datos requeridos para realizar un diagnóstico (por ejemplo,

imágenes de mamografía, resultados de análisis de sangre), y que muestre el resultado del diagnóstico de manera clara y concisa.

En este proyecto desarrollamos una interfaz web utilizando como principal herramienta la librería de Python Streamlit. Esta nos permite desarrollar interfaces web para aplicaciones de machine learning y ciencias de datos de una manera rápida y utilizando solo código Python.

En su fase de pruebas, la interfaz web consistía en 2 módulos: predicciones y archivos subidos. El primero, predicciones, es el módulo principal el cual le permite al médico radiólogo subir un archivo DICOM y realizar predicciones. El segundo, archivos subidos, nos ayuda a hacer predicciones sobre archivos que han sido recolectados del periodo de tiempo de marzo del 2022 a enero de 2023. Estos archivos fueron utilizados para reentrenar y ajustar el modelo a los datos generados por IXCHEN.

2. Integrar el modelo en un sistema de información médica existente.

Esto incluye, conectar el modelo con un sistema de información médica (por ejemplo, un sistema de gestión de usuarios), para que los resultados del diagnóstico del cáncer de mama se puedan almacenar y acceder de manera sencilla.

Desafortunadamente, ellos no tienen un sistema de información médica existente y todos los reportes y resultados de exámenes de los usuarios se manejan por medio de documentos físicos. Por lo tanto, esta fue el primer intento por modernizar la forma de trabajo actual de tal forma que los datos no puedan ser utilizados en futuros proyectos investigativos, tal como el presente.

Por lo tanto, para este proyecto, con la ayuda de la librería Streamlit de Python, se desarrolló un sistema de información el cual del flujo de trabajo que se detalla a continuación, así como cada uno de sus elementos.

El proceso principal de este sistema consiste en la realización de predicciones, para esto, el medico de turno debe seleccionar el archivo a subir, para el cual ya se ha cambiado la extensión correcta. Cuando se realiza la predicción, se genera

un reporte con los resultados de las predicciones para el modelo de segmentación y el modelo de clasificación.

El medico tiene la posibilidad de ajustar estos valores de tal manera que reflejen las predicciones correctas. Luego, el medico envía un reporte con los resultados y el ajuste correcto de los mismos. Es aquí donde se realizan varios procesos internos, empezando con guardar el archivo del que se hizo la predicción en un contenedor de un servicio de storage account en Azure. En esta cuenta de almacenamiento (storage account), consiste en 2 contenedores, uno utilizado para almacenar los archivos que se han recolectado en el periodo de tiempo de marzo del 2022 a enero del 2023 y otro contenedor para almacenar los archivos a los que se le ha hecho predicciones y nos sirve para, en la fase siguiente de pruebas, ajustar el modelo de aprendizaje con los datos generados por IXCHEN.

Adicional, se ha creado un archivo de Excel para darle seguimiento a las métricas que utilizaremos para mejorar ambos modelos de aprendizaje. Esto también puede hacerse con una librería de Python llamada MLflow, la cual sirve para este propósito. Sin embargo, para este proyecto se decidió utilizar Excel en lugar de MLflow, debido a que es un proyecto relativamente pequeño y Excel funciona.

3. Realizar pruebas de despliegue del modelo.

Se debe utilizar el modelo en un entorno de pruebas controlado (por ejemplo, con datos reales de usuarios), para asegurarse de que el modelo funciona de manera adecuada y se integra correctamente en el sistema de información médica.

Para poder desplegar el modelo y hacerlo disponible a través de internet, se utilizaron los servicios de Azure, el cual dispone de un conjunto de recursos que permiten agrupar servicios pertenecientes a un mismo proyecto. Incluye, un servicio de aplicación (app service) que facilita ejecutar la solución de Python con la librería Streamlit y el cual está conectado al repositorio privado de Github que contiene la solución, el servicio de aplicación también se encarga de construir esta solución para luego hacerla disponible y finalmente una cuenta de

almacenamiento (storage account) en la cual se almacenan los archivos DCM que se han utilizado en el sistema.

A continuación, una descripción general de los pasos anterior mencionados para desplegar un modelo en Azure y hacerlo disponible a través de internet:

- Utilizar los servicios de Azure para agrupar los servicios pertenecientes al mismo proyecto.
- Crear un servicio de aplicación (app service) para ejecutar la solución de Python con la librería Streamlit.
- Conectar el servicio de aplicación al repositorio privado de Github que contiene la solución.
- Utilizar el servicio de aplicación para construir la solución y hacerla disponible.
- Crear una cuenta de almacenamiento (storage account) en Azure para almacenar los archivos DCM utilizados en el sistema.

Es importante destacar que, a pesar de que el modelo esta disponible para ser utilizado en la web, existen ciertas limitaciones que deben ser tomadas en cuenta, como: necesidad de acceso a la red global, velocidad y precio del enlace a Internet, costo de procesamiento en la nube, entre otros. Alternativamente, se puede instalar en red local e incluso stand alone (equipo independiente). Para trabajar en ambiente de producción en la nube, con varios médicos radiólogos accediendo al modelo, se recomienda el aumento de las características del servicio de aplicación que ejecuta la solución de Python en Azure u otra plataforma con capacidad para desplegar el modelo de aprendizaje automático implementado.

4. Proporcionar soporte y mantenimiento del modelo.

En esta tarea, es conveniente hacerse preguntas sobre los objetivos del modelo, responderlas y resolver problemas que puedan surgir al utilizarlo (por ejemplo, problemas de integración con el sistema de información médica), y realizar actualizaciones periódicas del modelo para mejorar su desempeño o añadir nuevas funcionalidades.

En las etapas de pruebas se ha proporcionado soporte a los médicos radiólogos en el uso de la herramienta y de cada módulo. Esto se llevó a cabo a través de una demostración en persona con los médicos encargados de utilizar el modelo. Igualmente, ellos tuvieron la oportunidad de seguir el proceso con datos de usuarios reales y tuvimos la oportunidad de resolver dudas existentes sobre el modelo y los pasos necesarios utilizarlo, los cuales son detallados a continuación:

1- Se deben extraer los archivos del dispositivo ultrasonido, para hacer esto, el medico debe primero filtrar los estudios por aquellos relacionados a cáncer de mama. Estos son exportados a una memoria USB.
2- Estos archivos son exportados en carpetas para cada usuario y dentro de la carpeta las imágenes sin formato. Por lo tanto, un paso previo a poder realizar las predicciones es cambiar la extensión a DCM, que es el formato para archivos de imágenes médicas, DICOM. Debido a las limitaciones de la librería Streamlit, no podemos escoger un archivo sin formato y cambiarle la extensión internamente. En consecuencia, este proceso manual debe hacerse antes de realizar predicciones.
3- En su vista principal, la interfaz web nos muestra un punto de entrada de datos en el cual debemos seleccionar el archivo para el cual vamos a realizar las predicciones.
4- Una vez seleccionado el archivo, se muestra un recuadro el cual debemos ajustar a la parte central de la imagen de ultrasonido. Este paso es necesario para evitar introducir ruido al modelo con datos que no son necesarios para detectar la zona afectada y el tipo de tumor.
5- Se realiza la predicción.

6- Obtenemos las predicciones y, según el criterio del médico radiólogo, se ajustan las predicciones de manera que sirva para retroalimentación para ajustar el modelo con los nuevos datos generados por IXCHEN.

Para el uso de este modelo fuera de la plataforma desarrollada, guardamos el modelo utilizando el estándar de la librería Keras en formato h5. Este formato almacena el modelo entrenado listo para ser utilizado para hacer predicciones sobre datos nuevos.

Esto nos habilita el poder utilizar el modelo de diferentes formas. Una estrategia común es la de crear una API (Application Programming Interface, por sus siglas en ingles), donde se desarrolla la funcionalidad y parámetros de cada “endpoint” que utilizara el modelo. Otra alternativa es utilizando los “spaces” de Hugging Face, la cual es una plataforma que se enfoca en avanzar y democratizar la inteligencia artificial a través del código abierto y la ciencia abierta. Los spaces son una forma que utilizan la mayoría de las empresas que construyen modelos de aprendizaje de acceso libre para mostrar el funcionamiento del mismo.

Podría decirse que, durante la etapa de despliegue, en el modelo CRISP-DM, se realizan aquellas tareas necesarias para poner el modelo de aprendizaje automático en producción y hacerlo disponible para su uso por parte de los usuarios. El objetivo debe ser, garantizar el funcionamiento adecuado y correcto del modelo implementado, y su integración adecuada en un sistema de información médica existente, de manera eficiente.

7. Conclusiones

La implementación de un modelo de machine learning para el diagnóstico del cáncer de mama ha demostrado ser un paso significativo en la adaptación tecnológica en el campo de la medicina. A través de este proyecto, hemos logrado identificar fuentes de datos pertinentes y modelos factibles para su implementación, procesar los datos recopilados para su uso en el modelo, y crear un modelo de aprendizaje automático.

El modelo implementado y utilizado por el Centro de Mujeres IXCHEN ha demostrado ser una herramienta valiosa para apoyar el diagnóstico médico del cáncer de mama. Sin embargo, es importante recordar que este modelo es una herramienta de apoyo y no reemplaza el juicio clínico experto de los profesionales médicos.

En resumen, este proyecto ha demostrado el potencial de los modelos de machine learning en la mejora del diagnóstico del cáncer de mama. Esperamos que este trabajo sirva como un punto de partida para futuras investigaciones y desarrollos en este campo.

8. Recomendaciones

Al finalizar este trabajo monográfico hemos detectado algunas mejoras que podrían aplicarse para darle continuidad a este proyecto, las cuales son enumeradas a continuación:

1- Implementar el uso de prácticas de MLOps[3] para la automatización de procesos que, debido al tamaño del proyecto, son realizados manualmente. Estos procesos incluyen: reentrenamiento del modelo, procesamiento de datos y despliegue de modelo actualizado.

2- Debido al rápido crecimiento del campo de la inteligencia artificial, actualmente existen métodos más avanzados para resolver el problema principal de esta monografía, haciendo uso arquitecturas modernas potenciadas por los “transformers” los cuales han tenido éxito al ser aplicado a modelos de lenguaje y que también pueden ser implementados en tareas que involucren multimodalidad (combinando visión y lenguaje) [55] [56].

3- Automatizar el proceso de selección de zona central del ultrasonido correspondiente a la imagen ultrasonográfica. Actualmente este proceso es realizado de forma manual y puede crear confusiones al momento de decidir la zona que debería de ser recortada, teniendo así un efecto directo en el desempeño de los modelos que la utilicen.

[3] MLOps, por sus siglas en inglés, Machine Learning Operations, se refiere a un conjunto de prácticas para operacionalizar una solución de aprendizaje automático.

9. Bibliografía

[1] Ministerio de la Salud, «Mapa Nacional de la Salud en Nicaragua,» 2020. [En línea]. Available: http://mapasalud.minsa.gob.ni/mapa-de-padecimientos-de-salud-de-nicaragua/. [Último acceso: 20 4 2021].

[2] W. Al-Dhabyani, M. Gomaa, H. Khaled y F. Aly, «Dataset of Breast Ultrasound Images,» *Data in Brief,* 28 February 2020.

[3] R. Fajardo, J. Brandon, M. Gomez y J. Armando, «Modelo en Machine Learning para el Diagnóstico del Cáncer de Mama,» *Proyecto de Investigacion presentado para el grado de Especialista en Ingeniería de Software,* p. 168, 2020.

[4] A. E. Leivi, «Análisis de la implementación de Machine Learning en el diagnóstico por imágenes,» Universidad de San Andrés, Provincia de Buenos Aires, 2019.

[5] UCI Machine Learning Repository, «Breast Cancer Wisconsin Dataset (Diagnostic),» [En línea]. Available: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic). [Último acceso: Junio 2021].

[6] Y. Shen, F. E. Shamout, J. R. Oliver, J. Witowski, K. Kannan, J. Park, N. Wu, C. Huddleston, S. Wolfson, A. Millet, R. Ehrenpreis, D. Awal, C. Tyma, N. Samreen, Y. Gao, C. Chhor, S. Gandhi y Cindy, «Artificial Intelligence System Reduces False-Positive Findings in the Interpretation of Breast Ultrasound Exams,» Cold Spring Harbor Laboratory Press, 2021.

[7] S. Farah E., S. Yiqiu, W. J., O. Jamie R, K. Kawshik, W. Nan, P. Jungkyu, R. Beatriu, M. L., H. L. y G. Krzysztof J., «The NYU Breast Ultrasound Dataset v1.0,» 2021.

[8] O. Ronneberger, P. Fischer y T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» University of Freiburg, Germany, 2015.

[9] C. Camacho Piedra y V. Espíndola Zarazúa, «Actualización de la nomenclatura BI-RADS® por mastografía y ultrasonido,» Permanyer México SA de CV, 2018.

[10] Python Software Foundation, «Welcome to Python.org,» Python Software Foundation, [En línea]. Available: https://www.python.org/. [Último acceso: 2023 Noviembre 11].

[11] Anaconda Inc., «Packages | Cloud: Anaconda Data Science and Python Tools,» Anaconda Inc., [En línea]. Available: https://anaconda.cloud/package-categories. [Último acceso: 2023 Noviembre 11].

[12] Nature, «Ascent of machine learning in medicine,» *Nature Materials,* p. 18, 2019.

[13] M. Gail, L. Brinton, B. D, D. Corle, G. SB, S. C y M. JJ, «Projecting individualized probabilities of developing breast cancer for white females who are being examined annually,» *J Natl Cancer Inst,* vol. 81, nº 24, pp. 1876-1886, 20 December 1989.

[14] O. L. Mangasarian, W. N. Street y W. H. Wolberg, «Breast Cancer Diagnosis and Prognosis Via Linear Programming,» *Operations Research,* vol. 43, nº 4, pp. 548-725, 1995.

[15] W. Wolberg, S. W. Nick y O. Mangasarian, «Image analysis and machine learning applied to breast cancer diagnosis and prognosis,» *Anal Quant Cytol Histol,* pp. 77-87, 1995.

[16] W. H. Wolberg, W. N. Street y O. Mangasarian, «Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates,» *Cancer Lett,* pp. 163-171, 1994.

[17] S. Mohammed, S. Darrab, S. Noaman y G. Saake, «Analysis of Breast Cancer Detection Using Different Machine Learning Techniques,» *Data Mining and Big Data,* vol. 1234, pp. 108-117, 2020.

[18] A. P. Pawlovsky y M. Nagahashi, «A Method to Select a Good Setting for the kNN Algorithm when Using it for Breast Cancer Prognosis,» *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI),* pp. 189-192, 2014.

[19] K. Chévez, F. Reñazco y S. García, «Fundamentos y Aplicaciónes de la Robotica y su Importancia en Nicaragua,» Universidad Nacional Autonoma de Nicaragua, Managua, 2006.

[20] G. Espinoza, N. Gaitán y T. Silva, «Metodos y lenguajes de programación utilizados en la Robótica, con énfasis en la programación Gestual,» Universidad Nacional Autonoma de Nicaragua, Managua.

[21] M. A. Larios Zambrana, «Comportamiento clinico Patologico del Cancer de mama en pacientes atendidas en el Hospital Escuela Carlos Roberto Huembes en el periodo de Enero 2011-Diciembre 2015,» Universidad Nacional Autónoma de Nicaragua, Managua, 2017.

[22] Organización Panamericana de la Salud - Organización Mundial de la Salud, «Día Mundial contra el Cáncer 2018: Nosotros podemos, yo puedo,» 2018. [En línea]. Available: https://www.paho.org/nic/index.php?option=com_content&view=article&id=912:dia-mundial-contra-cancer-2018-nosotros-podemos-yo-puedo&Itemid=244. [Último acceso: 20 04 2021].

[23] A. L. Fradkov, «Early History of Machine Learning,» *21th IFAC World Congress,* vol. 53, nº 2, pp. 1385-1390, 2020.

[24] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2019, p. 819.

[25] A. Muñoz, Cancer, Genes y Nuevas Terapias, Madrid: Hélice, 1997.

[26] J. M. Junco, A. M. Loza y Á. A. Gónzalez, «Bases moleculares del cáncer,» *Revista de Investigación Clínica,* vol. 58, nº 1, pp. 56-70, 2006.

[27] T. Mitchell, Machine Learning, 1 ed., McGraw-Hill Education, 1997.

[28] P. N. Stuart J. Russell, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall, 2009.

[29] A. Halevy, P. Norvig y F. Pereira, «The Unreasonable Effectiveness of Data,» IEEE Computer Society, 2009.

[30] MathWorks, «Support Vector Machine (SVM),» MathWorks, [En línea]. Available: https://la.mathworks.com/discovery/support-vector-machine.html. [Último acceso: 20 01 2022].

[31] UCI Machine Learning Repository, «UCI Machine Learning Repository: Iris Data Set,» UCI Machine Learning Repository, [En línea]. Available: https://archive.ics.uci.edu/ml/datasets/iris. [Último acceso: January 2022].

[32] S. Ray, «A Quick Review of Machine Learning Algorithms,» de *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019.

[33] V. Sharma, R. Sachin y D. Anurag, «A comprehensive study of artificial neural networks.,» *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 2, nº 10, pp. 278-284, 2012.

[34] A. Dey, «Machine Learning Algorithms: A Review,» *International Journal of Computer Science and Information Technologies,* vol. 7, nº 3, pp. 1174-1179, 2016.

[35] A. Jain, J. Mao y K. M. Mohiuddin, «Artificial neural networks: a tutorial,» *Computer,* vol. 29, nº 3, pp. 31-44, 1996.

[36] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» Nature, 2015.

[37] A. Ajit, K. Acharya y A. Samanta, «A Review of Convolutional Neural Networks,» IEEE, Vellore, 2020.

[38] OpenGenus Tech Review Team, «Multilayer Perceptrons vs CNN,» [En línea]. Available: https://iq.opengenus.org/multilayer-perceptrons-vs-cnn/. [Último acceso: 10 Agosto 2023].

[39] Uniqtech, «Multilayer Perceptron (MLP) vs Convolutional Neural Network in Deep Learning,» 22 Diciembre 2018. [En línea]. Available: https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1.

[40] M. Elgendy, Deep Learning for Vision Systems, Manning Publications, 2020.

[41] J. Krohn, G. Beyleveld y A. Bassens, Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence, Addison-Wesley Professional, 2019.

[42] P. Costa, «Redes neuronales convolucionales explicadas,» [En línea]. Available: https://pochocosta.com/podcast/redes-neuronales-convolucionales-explicadas/.

[43] CRISP-DM Consortium, «CRISP-DM 1.0: Step-by-step data mining guide,» SPSS, 2000.

[44] A. Grigorev, Machine Learning Bookcamp, Manning Publications Co., 2021.

[45] A.-I. Andrés, M.-J. Leonel y Z.-D. Martha, «An overview of deep learning in medical imaging,» Elsevier Ltd, Bogotá, 2021.

[46] Indigo Technologies, «IndiGo,» Indigo Technologies, [En línea]. Available: https://indigo.tech/.

[47] Clinical Graphics, «Clinical Graphics - ¿En qué consisten los datos DICOM?,» Clinical Graphics, [En línea]. Available: https://www.clinicalgraphics.com/es/asistencia/actualizacion-de-los-datos-dicom/en-que-consisten-los-datos-dicom.

[48] A. Y. LeCun, L. Bottou, G. B. Orr y K.-R. Müller, «Efficient BackProp,» de *Neural Networks: Tricks of the Trade*, vol. 7700, Berlin, Springer, 1998, pp. 9-48.

[49] S. Raschka, «Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning,» 2018.

[50] K. He, X. Zhang, S. Ren y J. Sun, «Deep Residual Learning for Image Recognition,» Microsoft Research, 2015.

[51] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» *Advances in Neural Information Processing Systems 25 (NIPS 2012),* pp. 1097-1105, 2012.

[52] V. Kurama, «A Review of Popular Deep Learning Architectures: AlexNet, VGG16, and GoogleNet,» Paperspace by DigitalOcean, 2020. [En línea]. Available: https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/.

[53] Datagen, «ResNet-50: The Basics and a Quick Tutorial,» Datagen, [En línea]. Available: https://datagen.tech/guides/computer-vision/resnet-50/.

[54] T. M. Ayyar, «A practical experiment for comparing LeNet, AlexNet, VGG and ResNet models with their advantages and disadvantages,» 6 Noviembre 2020. [En línea]. Available: https://tejasmohanayyar.medium.com/a-practical-experiment-for-comparing-lenet-alexnet-vgg-and-resnet-models-with-their-advantages-d932fb7c7d17.

[55] Z. Kai, Y. Jun, Y. Zhiling, L. Yixin, A. Eashan, F. Sunyang, C. Xun, C. Chen, Z. Yuyin, L. Xiang, H. Lifang, D. Brian D., L. Quanzheng, C. Yong, H. Liu y S. Lichao, «BiomedGPT: A Unified and Generalist Biomedical Generative Pre-trained Transformer for Vision, Language, and Multimodal Tasks,» arxiv, 2023.

[56] W. Guangyu, Y. Guoxing, D. Zongxin, F. Longjun y L. Xiaohu, «ClinicalGPT: Large Language Models Finetuned with Diverse Medical Data and Comprehensive Evaluation,» arxiv, 2023.

[57] C. Schröer, F. Kruse y J. M. Gómez, «A Systematic Literature Review on Applying CRISP-DM Process Model,» *CENTERIS 2020 - International Conference on ENTERprise Information Systems / ProjMAN 2020 - International Conference on Project MANagement / HCist 2020 - International Conference on Health and Social Care Information Systems and Technologies 2020, CENTERI,* vol. 181, pp. 526-534, 2021.

[58] Microsoft Docs, «What is the Team Data Science Process? | Microsoft Docs,» Microsoft, 17 11 2020. [En línea]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview. [Último acceso: 20 5 2021].

[59] Google Cloud, «Machine Learning workflow,» Google, [En línea]. Available: https://cloud.google.com/ai-platform/docs/ml-solutions-overview. [Último acceso: 20 5 2021].

[60] D. P. Chávez Guadarrama y K. Y. González Flores, «Detección de Cáncer de Mama mediante Redes Neuronales,» Instituto Politécnico Nacional, Ciudad de México, 2016.

Anexos

Anexo 1: Ejemplo de implementación de transfer learning

```
# Cargar un modelo pre-entrenado en ImageNet
base_model = cargar_modelo_pre_entrenado("imagenet")

# Cargar las capas del modelo base para no sacudirlo durante el entrenamiento
for capa in base_model.layers:
    capa.trainable = False

# Agregar nuevas capas al final del modelo base para adaptarlo a la tarea nueva
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
predicciones = Dense(10, activation='softmax')(x)

# Crear el nuevo modelo
modelo_final = Model(inputs=base_model.input, outputs=predicciones)

# Compilar el modelo
modelo_final.compile(optimizer="rmsprop", loss="categorical_crossentropy")

# Entrenar el modelo en el nuevo conjunto de datos
modelo_final.fit(datos_entrenamiento, etiquetas_entrenamiento)
```

Figura 23: Pseudocódigo de Transfer Learning. Fuente: Imagen propia.

La Figura 23 muestra un pseudocódigo de los pasos necesarios para implementar la técnica de transfer learning, los cuales se resumen a continuación:

1- Cargar un modelo pre-entrenado.
2- Congelar las capas del modelo base para que no se actualicen durante el entrenamiento.
3- Agregar nuevas capas al final del modelo base para adaptarlo a la nueva tarea.
4- Crear el nuevo modelo.
5- Compilar el modelo.
6- Entrenar el modelo en el nuevo conjunto de datos.

Anexo 2: Conjunto de datos Universidad El Cairo

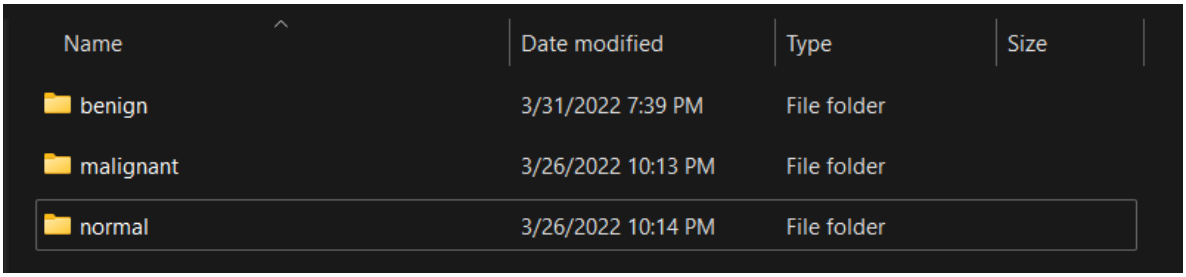


Figura 24: Distribución de las imágenes de conjunto de datos Universidad El Cairo. Fuente: Imagen propia.

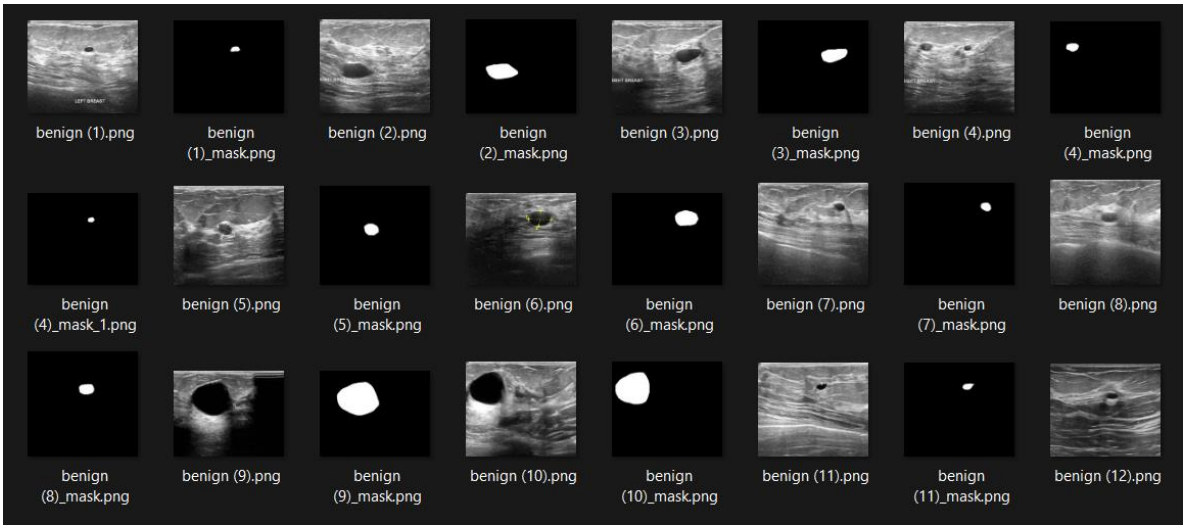


Figura 25: Ejemplo de datos para clase benigno. Fuente: Imagen propia.

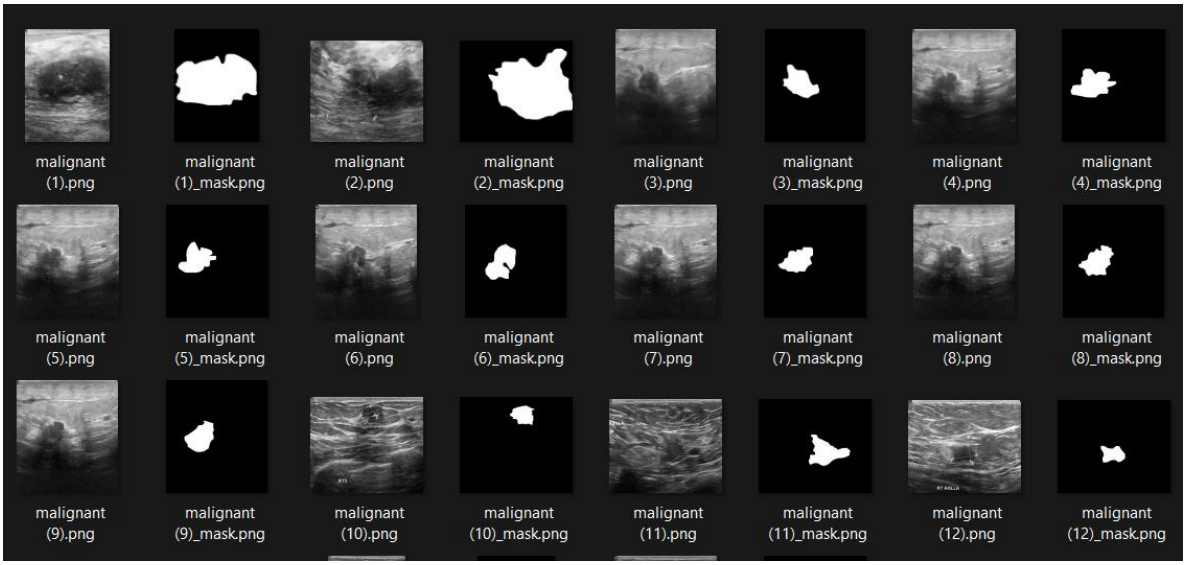


Figura 26: Ejemplo de datos para clase maligno. Fuente: Imagen propia.

b

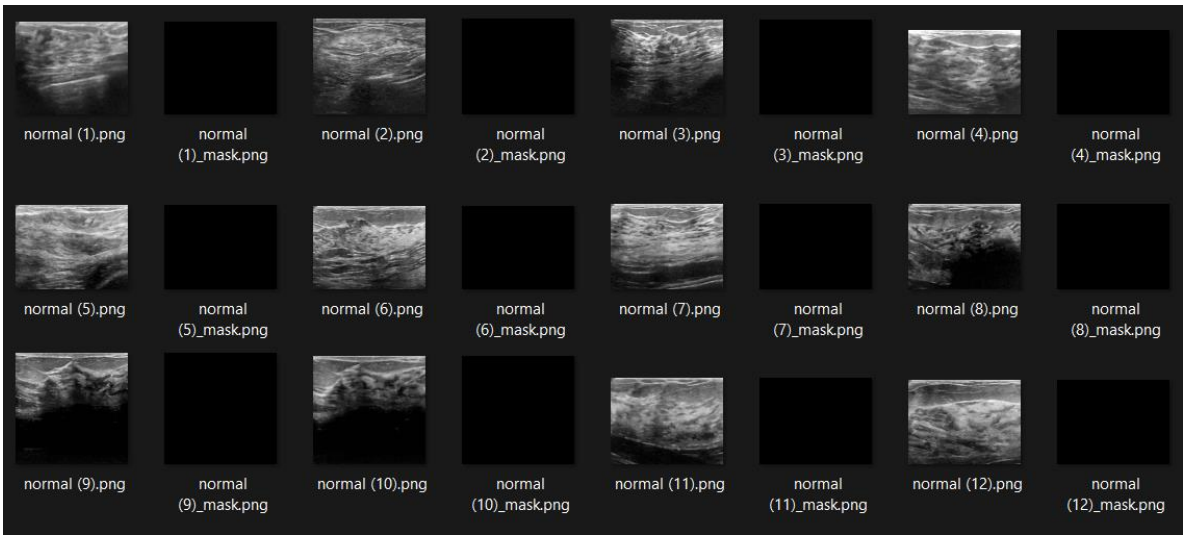
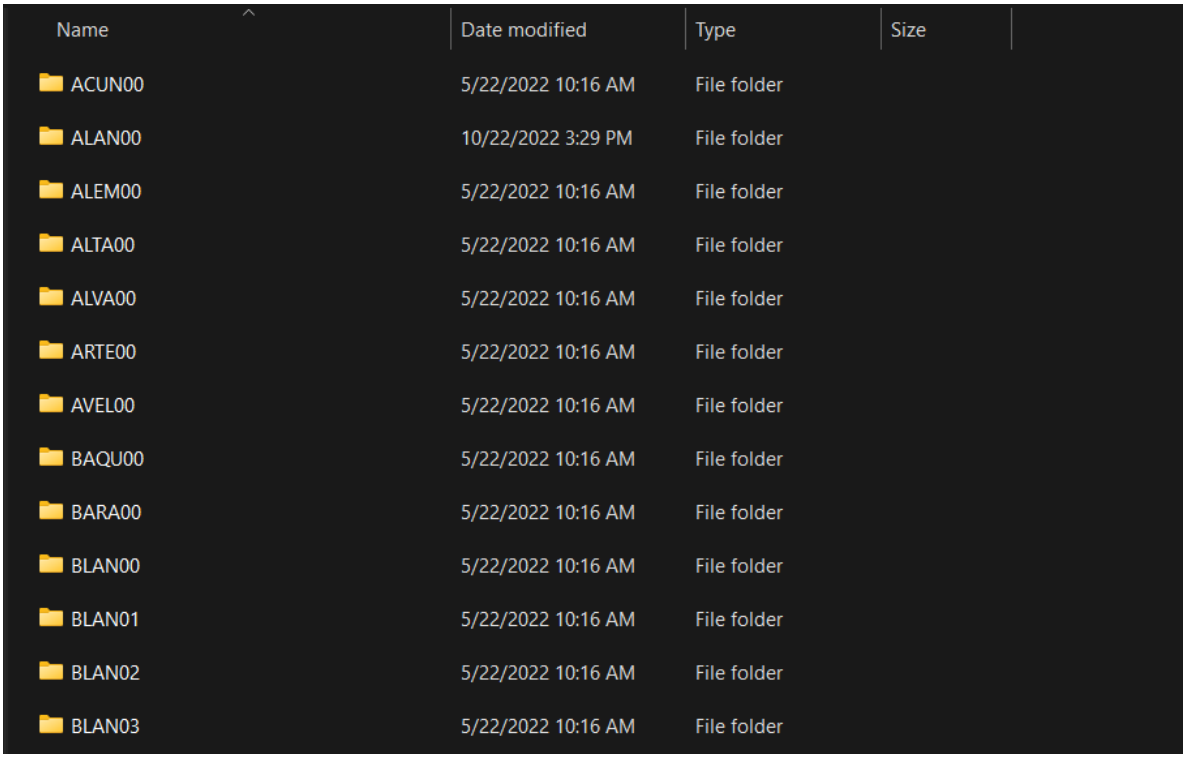


Figura 27: Ejemplo de datos para clase normal. Fuente: Imagen propia.

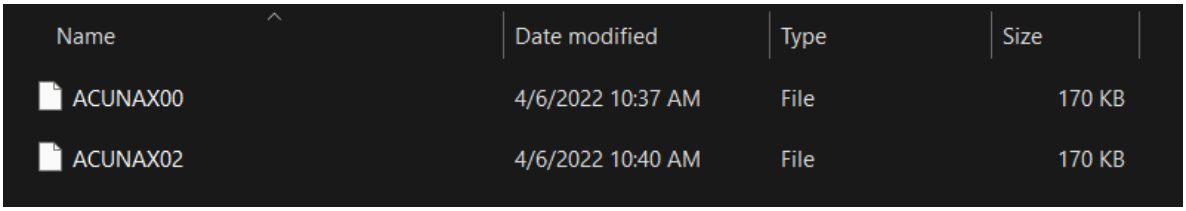
En la Figura 24 se muestra como luce el agrupamiento del conjunto de datos [2], una vez descargado a una estación de trabajo local. En las figuras 25, 26 y 27 se muestran ejemplos de cómo lucen los datos en cada clase: benigno, maligno y normal respectivamente. Se hace notar que, cada imagen de ultrasonido dispone de su “máscara”, que representa la segmentación de la zona afectada.

Anexo 3: Forma de extracción de datos de IXCHEN



Name	Date modified	Type	Size
ACUN00	5/22/2022 10:16 AM	File folder	
ALAN00	10/22/2022 3:29 PM	File folder	
ALEM00	5/22/2022 10:16 AM	File folder	
ALTA00	5/22/2022 10:16 AM	File folder	
ALVA00	5/22/2022 10:16 AM	File folder	
ARTE00	5/22/2022 10:16 AM	File folder	
AVEL00	5/22/2022 10:16 AM	File folder	
BAQU00	5/22/2022 10:16 AM	File folder	
BARA00	5/22/2022 10:16 AM	File folder	
BLAN00	5/22/2022 10:16 AM	File folder	
BLAN01	5/22/2022 10:16 AM	File folder	
BLAN02	5/22/2022 10:16 AM	File folder	
BLAN03	5/22/2022 10:16 AM	File folder	

Figura 28: Ejemplo de archivos exportados de un Ecógrafo. Fuente: Imagen propia.



Name	Date modified	Type	Size
ACUNAX00	4/6/2022 10:37 AM	File	170 KB
ACUNAX02	4/6/2022 10:40 AM	File	170 KB

Figura 29: Archivo individuales de cada paciente. Fuente: Imagen propia.

En la Figura 28 se muestra un ejemplo de cómo los exámenes son exportados a partir de un ecógrafo. Cada carpeta representa a una usuaria de servicios médicos. La Figura 29 muestra cómo se exportan los exámenes realizados a cada una de ellas. Se puede observar que los archivos exportados no tienen extensión.

Anexo 4: División de conjuntos de datos de entrenamiento y validación

```
training_ds = tf.keras.utils.image_dataset_from_directory(
    directory=data_dir, \
    labels='inferred',
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
    )
training_ds
```

```
Found 780 files belonging to 3 classes.
Using 624 files for training.
```

Figura 30: Separación conjunto de datos de entrenamiento. Fuente: Imagen propia.

```
validation_ds = tf.keras.utils.image_dataset_from_directory(
    directory=data_dir,
    labels='inferred',
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
    )
validation_ds
```

```
Found 780 files belonging to 3 classes.
Using 156 files for validation.
```

Figura 31: Separación conjunto de datos de validación. Fuente: Imagen propia.

En la Figura 30 se muestra el código necesario para hacer la separación del conjunto de datos total de la primera fase en datos de entrenamiento. La Figura 31 muestra el mismo proceso para obtener los datos de validación.

Anexo 5: Selección estratificada de conjunto de datos de entrenamiento y validación

```python
import os
import pandas as pd

root_dir = './Dataset_BUSI_with_Gt_classification_new/'

folders = ['benign', 'malignant', 'normal']

data = []

for folder in folders:
    files = os.listdir(os.path.join(root_dir, folder))

    for file in files:
        data.append([os.path.join(root_dir, folder, file), folder])

df = pd.DataFrame(data, columns=['file_path', 'folder'])
```

	file_path	folder
0	./Dataset_BUSI_with_Gt_classification_new/beni...	benign
1	./Dataset_BUSI_with_Gt_classification_new/beni...	benign
2	./Dataset_BUSI_with_Gt_classification_new/beni...	benign
3	./Dataset_BUSI_with_Gt_classification_new/beni...	benign
4	./Dataset_BUSI_with_Gt_classification_new/beni...	benign
..	...	...
933	./Dataset_BUSI_with_Gt_classification_new/norm...	normal
934	./Dataset_BUSI_with_Gt_classification_new/norm...	normal
935	./Dataset_BUSI_with_Gt_classification_new/norm...	normal
936	./Dataset_BUSI_with_Gt_classification_new/norm...	normal
937	./Dataset_BUSI_with_Gt_classification_new/norm...	normal

[938 rows x 2 columns]

Figura 32: Lista de imágenes y su clasificación. Fuente: Imagen propia.

```python
from sklearn.model_selection import train_test_split

X = df['file_path']
y = df['folder']

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2, random_state=42)
```

Figura 33: División estratificada de conjuntos de datos. Fuente: Imagen propia.

f

```python
# moving train destination
import shutil

destination_dir = './Dataset_BUSI_with_Gt_classification_train_val/train'

for file_path, folder in zip(X_train, y_train):
    file_name = os.path.basename(file_path)

    os.makedirs(os.path.join(destination_dir, folder), exist_ok=True)

    destination_path = os.path.join(destination_dir, folder, file_name)

    shutil.copy(file_path, destination_path)
```

Figura 34: Copia de conjunto de datos de entrenamiento a un nuevo directorio. Fuente: Imagen propia.

```python
# moving val destination
destination_dir = './Dataset_BUSI_with_Gt_classification_train_val/valid'

for file_path, folder in zip(X_test, y_test):
    file_name = os.path.basename(file_path)

    os.makedirs(os.path.join(destination_dir, folder), exist_ok=True)

    destination_path = os.path.join(destination_dir, folder, file_name)

    shutil.copy(file_path, destination_path)
```

Figura 35: Copia de conjunto de datos de validación a un nuevo directorio. Fuente: Imagen propia.

Inicialmente, en la Figura 32, se puede observar cómo se preparan las imágenes, de tal forma que, se dispone de la imagen y su clasificación. Luego, en la Figura 33, se realiza una división estratificada de conjuntos de datos de entrenamiento y validación. En las figuras 34 y 35 se muestra cómo se copia a un nuevo directorio los datos de entrenamiento y validación, respectivamente.

Anexo 6: Categorías de evaluación BI-RADS

TABLA 3. Categorías de evaluación BI-RADS®

Categoría 0	Mastografía: incompleta. Evaluación de imagen adicional necesaria y/o mastografías anteriores para su comparación Ultrasonido y resonancia magnética: incompleta. Evaluación de imagen adicional necesaria		
Categoría 1	Negativa		
Categoría 2	Benigna		
Categoría 3	Probablemente benigna		
Categoría 4	Sospechosa	Mastografía y ultrasonido	4A: Baja sospecha de malignidad 4B: Moderada sospecha de malignidad 4C: Alta sospecha de malignidad
Categoría 5	Altamente sugestiva de malignidad		
Categoría 6	Diagnóstico maligno comprobado por biopsia		

Figura 36: Tabla de categorías de evaluación BI-RADS [9].

Esta tabla muestra las categorías de evaluación de BI-RADS y su “traducción” a las categorías de normal o negativa, benigna y maligna. Esta tabla se usó de referencia para poder obtener la clasificación correcta basada en el reporte del médico radiólogo.

h

Anexo 7: Arquitectura U-Net (modelo de segmentación) implementada en TensorFlow

```python
import tensorflow as tf
from tensorflow.keras import layers

def unet(input_shape, num_classes):
    inputs = tf.keras.Input(shape=input_shape)

    # Encoder
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(pool1)
    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(conv2)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(pool2)
    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(conv3)
    pool3 = layers.MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = layers.Conv2D(512, 3, activation='relu', padding='same')(pool3)
    conv4 = layers.Conv2D(512, 3, activation='relu', padding='same')(conv4)
    pool4 = layers.MaxPooling2D(pool_size=(2, 2))(conv4)

    # Bottleneck
    conv5 = layers.Conv2D(1024, 3, activation='relu', padding='same')(pool4)
    conv5 = layers.Conv2D(1024, 3, activation='relu', padding='same')(conv5)

    # Decoder
    up6 = layers.concatenate([layers.UpSampling2D(size=(2, 2))(conv5), conv4], axis=3)
    conv6 = layers.Conv2D(512, 3, activation='relu', padding='same')(up6)
    conv6 = layers.Conv2D(512, 3, activation='relu', padding='same')(conv6)

    up7 = layers.concatenate([layers.UpSampling2D(size=(2, 2))(conv6), conv3], axis=3)
    conv7 = layers.Conv2D(256, 3, activation='relu', padding='same')(up7)
    conv7 = layers.Conv2D(256, 3, activation='relu', padding='same')(conv7)

    up8 = layers.concatenate([layers.UpSampling2D(size=(2, 2))(conv7), conv2], axis=3)
    conv8 = layers.Conv2D(128, 3, activation='relu', padding='same')(up8)
    conv8 = layers.Conv2D(128, 3, activation 'relu', padding 'same')(conv8)

    up9 = layers.concatenate([layers.UpSampling2D(size=(2, 2))(conv8), conv1], axis=3)
    conv9 = layers.Conv2D(64, 3, activation 'relu', padding 'same')(up9)
    conv9 = layers.Conv2D(64, 3, activation 'relu', padding 'same')(conv9)

    # Output layer
    outputs = layers.Conv2D(num_classes, (1 ,1), activation 'softmax')(conv9)

    # Create model
    model = tf.keras.Model(inputs=inputs , outputs=outputs)

return model
```

Figura 37: Implementación arquitectura U-Net. Fuente: Imagen propia.

En la Figura 37 se puede ver el código necesario para implementar la arquitectura U-Net en el framework de aprendizaje profundo TensorFlow con Keras. En esta, se distinguen: la etapa inicial del encoder que se encarga de codificar y la imagen de entrada; decoder, que trata de reconstruir la imagen previamente codificada y output que se encarga de dar como resultado la imagen final segmentada.

Anexo 8: Proceso de fine-tuning utilizando las arquitecturas AlexNet y ResNet-50

```
from pathlib import Path
from fastai.vision.all import *
```

```
import torch
torch.cuda.empty_cache()
```

```
path = Path('./')
```

```
dls = ImageDataLoaders.from_folder(
    path,
    train='train',
    valid='valid',
    item_tfms=Resize(256),
    bs=8
)
```

Figura 38: Preparación de conjunto de datos para proceso de fine-tuning. Fuente: Imagen propia.

```
learn = vision_learner(dls, alexnet, metrics=[error_rate, accuracy])
```

Figura 39: Preparación de modelo base y conjunto de datos para hacer fine-tuning. Fuente: Imagen propia.

```
learn.fine_tune(10)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.307300	0.810020	0.287234	0.712766	00:23

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.962675	0.730335	0.260638	0.739362	00:15
1	0.731420	0.671351	0.228723	0.771277	00:15
2	0.906934	0.640281	0.223404	0.776596	00:15
3	0.713911	0.619577	0.234043	0.765957	00:15
4	0.633347	0.570178	0.207447	0.792553	00:16
5	0.543695	0.540378	0.212766	0.787234	00:16
6	0.420230	0.535379	0.191489	0.808511	00:16
7	0.409691	0.433293	0.186170	0.813830	00:16
8	0.374294	0.456574	0.170213	0.829787	00:16
9	0.340338	0.438709	0.159574	0.840426	00:15

Figura 40: Resultado proceso de fine-tuning. Fuente: Imagen propia.

Figura 41: Resultados de proceso de fine-tuning. Fuente: Imagen propia.

Estas imágenes dan una idea de los pasos dados para realizar el proceso de fine-tuning, sobre una arquitectura pre-entrenada (AlexNet). Inicialmente, en la Figura 38, se prepara el conjunto de datos en el formato esperado por el "ImageDataLoader" el cual espera tener una carpeta para los datos de entrenamiento y otra para los datos de validación. En la Figura 39, se muestra cómo se utilizan estos datos y la arquitectura que será ajustada con ellos. Luego se realiza el fine-tuning durante 10 iteraciones. La figura 40 muestra el resultado de cada iteración y cómo la métrica de precisión (accuracy) va mejorando en cada una de ellas. Finalmente, la Figura 41, muestra algunos resultados obtenidos sobre el conjunto de datos de validación.

Anexo 9: Proceso para implementar la solución desarrollada en esta monografía.

Se describen a continuación, los pasos a seguir para implementar el modelo en un equipo independiente, al ser la manera más básica posible. El código fuente a utilizar, se encuentra en CD adjunto a esta monografía.

1- Instalar el lenguaje de programación Python [10].
2- Se deberán instalar las librerías contenidas en el archivo requirements.txt. Para esto, en la consola, se debe ir al directorio del proyecto y ejecutar el siguiente comando “pip install -r requirements.txt”.
3- Para ejecutar la aplicación, se escribe el comando: “streamlit run app.py”.
4- Finalmente, una vez ejecutado el comando anterior se deberá de abrir automáticamente un nuevo navegador con la aplicación funcionando.