UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE COMPUTACIÓN

DESARROLLO DE UNA LIBRERÍA PARA LA CLASIFICACIÓN DE IMÁGENES DE MAMOGRAFÍAS EN FORMATO DICOM.

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Ciencias de la Computación

AUTORES:

WILSON BRYAN AGUILAR SIQUIGUA

PABLO GABRIEL CACUANGO PUJOTA

TUTOR:

WASHINGTON ARSENIO RAMÍREZ MONTALVAN

Quito - Ecuador
2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Wilson Bryan Aguilar Siquigua con documento de identificación N° 1721181210 y Pablo Gabriel Cacuango Pujota con documento de identificación N° 1753543261, manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 14 de marzo del año 2022.

Wilson Bryan Aguilar Siquigua

1721181210

Pablo Gabriel Cacuango Pujota

1753543261

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Wilson Bryan Aguilar Siquigua con documento de identificación N° 1721181210 y Pablo Gabriel Cacuango Pujota con documento de identificación N° 1753543261, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos los autores del Proyecto Técnico: "Desarrollo de una librería para la clasificación de imágenes de mamografías en formato DICOM.", el cual ha sido desarrollado para optar por el titulo de: Ingeniero en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la biblioteca de la Universidad Politécnica Salesiana.

Quito, 14 de marzo del año 2022.

Wilson Bryan Aguilar Siquigua

1721181210

Pablo Gabriel Cacuango Pujota

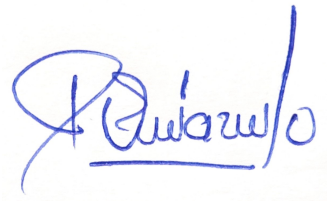
1753543261

CERTIFICADO DE DIRECIÓN DEL TRABAJO DE TITULACIÓN

Yo, Washington Arsenio Ramírez Montalvan con documentos de identificación N° 171080468 1, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UNA LIBRERÍA PARA LA CLASIFICACIÓN DE IMÁGENES DE MAMOGRAFÍAS EN FORMATO DICOM., realizado por Wilson Bryan Aguilar Siquigua con documento de identificación N° 1721181210 y por Pablo Gabriel Cacuango Pujota con documento de identificación N° 1753543261, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 14 de marzo del año 2022.

Atentamente,



Ing. Washington Arsenio Ramírez Montalvan, Ph.D.

1710804681

DEDICATORIA

Dedico este trabajo a mi madre Marcela Siquigua y mi padre Wilson German Aguilar por todo el esfuerzo y sacrificio que han hecho para que pueda llegar hasta aquí, además de haberme enseñado los valores y principios que hoy forman parte de mi persona.

A mis hermanos y en especial a mi tía Bertha Siquigua quienes me han brindado su apoyo incondicional durante todo el proceso de mi formación. También a mis sobrinos quienes me han dado momentos de alegría y me inspiran a ser un buen ejemplo de persona para ellos.

Finalmente a mis compañeros de la carrera y en especial a mi amigo Gabriel Cacuango con quienes hemos atravesado un largo camino y varias dificultados para culminar esta etapa de nuestra vida.

Wilson Bryan Aguilar Siquigua

A Dios con mucho amor y gratitud por darme a unos padres que me han brindado su apoyo y cariño durante toda esta etapa siendo para mi un ejemplo de honestidad y esfuerzo por lo cual estoy eternamente agradecido. A mis hermanos, por ser un apoyo incondicional durante toda mi vida y quienes nunca bajaron los brazos para que yo tampoco lo haga.

A mi amigo Wilson Aguilar y compañeros por brindarme su amistad durante todo el camino y compartir este momento con aquellos que lamentablemente ya no están. A mi novia y amiga Yadira, por estar conmigo en aquellos momentos en que el estudio y el trabajo ocuparon mi tiempo y esfuerzo. Gracias por tu ayuda.

Pablo Gabriel Cacuango Pujota

AGRADECIMIENTOS

A la Universidad Politécnica Salesiana, por aportar con todo el conocimiento y herramientas para lograr la culminación de nuestra carrera.

A nuestro tutor Ing. Washington Ramírez, por su acertada guía durante el desarrollo de este proyecto, brindar su conocimiento y enseñarnos que siempre hay algo más para aprender. Por su dedicación, tiempo y compromiso. Por ser nuestra guía y caminar siempre a nuestro lado, mostrándonos las mejores alternativas con un apoyo incondicional para alcanzar nuestro objetivo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ALGORITMOS

RESUMEN

El cáncer de mama es una de las principales causas de mortalidad en mujeres, debido a la falta de información, concientización y en especial a la omisión de controles. Actualmente los diagnósticos son realizados por personal médico capacitado; sin embargo, puede existir la posibilidad de cometer errores humanos en los análisis mamográficos. El presente trabajo se fundamenta en la creación de una librería para para analizar imágenes médicas mamográficas DICOM con técnicas de Machine Learning. La metodología utilizada sigue los pasos como la lectura de imágenes mamográficas DICOM evaluadas con Fuzzy C-Means, K-Means, Random Forest, k-Nearest Neighbors, Support Vector Machines y Convolutional Neural Networks. Replanteamiento de los algoritmos de Machine Learning escritos en C/C++ de tal manera que se integren a la librería. Se utilizó un dataset de 150 imágenes mamográficas DICOM, y los datos resultantes fueron comparados utilizando MatLab para verificar la tasa de error para cada algoritmo. Para la evaluación de la librería se obtienen los resultados, el error promedio entre la librería y MatLab fue de 0.7 %, el benchmarking para cada técnica en CPU fue de 11.47 %, para memoria RAM su uso promedio es de 300 Mb y el tiempo de ejecución fue 101 segundos en promedio. Se concluye que a pesar de que el rendimiento de la librería es aceptable, puede ser optimizado aplicando técnicas de pre-procesamiento, segmentación, extracción y selección de características reduciendo la carga computacional.

Palabras clave: Clasificación, imágenes médicas, DICOM, librería en C/C++, Machine Learning, Cáncer de mama.

ABSTRACT

The breast cancer is one of the main causes of mortality in women, due to lack of information, awareness and above all the omission of controls. Currently, diagnoses are made by trained medical staff; however, there may be the possibility of human errors in the mamographic analysis. The mothodology used follows the steps such as reading DICOM mammographic images evaluated with Fuzzy C-Means, K-Means, Random Forest, k-Nearest Neighbors, Support Vector Machines and Convolutional Neural Networks. Rethinking Machine Learning algorithms written in C/C++ in such a way that they could be integrated inside the library. A dataset of 150 DICOM mammographic images was used, and the resulting data were compared using Matlab to verify the error rate for each algorithm. For the library evaluation the results obtained are, the average error between the library and MatLab was 0.7 %, the benchmarking for each technique in CPU was 11.47 %, for RAM memory its average use is 300 Mb and the time of use was 101 seconds on average. It is concluded that although the performance of the library is acceptable, it can be optimized by applying pre-processing, segmentation, extraction and feature selection techniques, reducing the computational load.

Keywords: Classification, medical images, DICOM, C/C++ library, Machine Learning, breast cancer. This work is based on the creation of a library to classify DICOM medical images with Machine Learning techniques.

INTRODUCCIÓN

El cáncer de mama es una de las principales causas de mortalidad en las mujeres y puede ser detectado haciendo uso de las mamografías (Routray & Rath, 2018). La clasificación en el área Machine Learning es una de las técnicas más comunes cuando se trata de dividir un conjunto de datos en ciertas categorías (Kamalakannan et al., 2015). En la medicina se utiliza clasificación trabajar en imágenes mamográficas que ayuden a detectar posibles casos de cáncer. Existen diferentes técnicas de clasificación como: Arboles de decisión, K-nearest Neighbour, Nave bayes, entre otros (Kumari & Saxena, 2018).

En el desarrollo de software es común empaquetar el código en diferentes componentes que sean lo suficientemente flexibles para que puedan ser utilizados en distintos proyectos (Pirklbauer et al., 1993). Haciendo uso de este mecanismo se puede crear una librería para clasificar imágenes de mamografías en formato DICOM.

A nivel mundial el cáncer de mama o neoplasia maligna es la más común en la población femenina abarcando aproximadamente un 25 % de todos los tipos de canceres de mujeres alrededor del mundo. En el 2020 se detectaron 2.2 millones de casos estableciéndolo como el más común también se establece que una de cada doce mujeres se enfermará de cáncer de mama a lo largo de su vida. Lamentablemente en el mismo año se han registrado 685,000 defunciones como consecuencia de cáncer de mama siendo menos la probabilidad de superar esta enfermedad (Organizacón Mundial de la Salud, 2021).

Los países con pocos recursos económicos como en India y Sudáfrica la posibilidad de supervivencia al cáncer es del 66 % y el 40 %, respectivamente (Organizacón Mundial de la Salud, 2021). También se toma en cuenta que en América ocupa un total de 15.4 % de total de

la mortalidad en mujeres a nivel mundial de un total de 4.4 millones de decesos, estableciendo en América como el cáncer más comúnmente diagnosticado en mujeres (Sung et al., 2021). A pesar de que en los últimos años se ha presentado una mejora gracias a varias campañas que se han realizado como la autoexploración y chequeos médicos como las mamografías. La falta de conciencia y de equipos para la detección de estas enfermedades genera un riesgo evidente en la población.

En Ecuador, según el ministerio de salud pública y datos del INEC, de las 3,430 mortalidades reportadas por este motivo dentro del rango de los años 2012 y 2017, el 99.3 % de las personas que fallecieron por esta causa fueron mujeres. Solo en el año 2017 se registraron 670 defunciones por cáncer de mama en mujeres y 3 en hombres, que corresponde la primera a una tasa del 3.99 % de defunciones por cada 100,000 habitantes. La mortalidad en orden decente en mujeres es el siguiente: mama, estómago, cuello uterino, bronquios y pulmón, y de hígado y vías biliares. En los últimos reportes Quito ocupa el primer lugar en diagnósticos de cáncer de mama en mujeres (Ministerio de Salud Pública, 2021).

PROBLEMA

El presente proyecto de investigación responderá a la siguiente problemática: ¿En qué medida ayudaría el desarrollo de una librería para la clasificación de imágenes de mamografías en formato DICOM?. En la industria existen muchas librerías con módulos de clasificación de imágenes que son muy genéricas y no están adaptadas para ser utilizadas con formatos de imagen DICOM que son esenciales en el área médica. El procesamiento de este tipo de imágenes puede ser una operación con alto costo de recursos computacionales. El uso de lenguajes como C y C++ ayuda a la optimización de recursos computacionales como el procesador y la memoria. De esta manera con la ayuda de clasificadores que se encuentren optimizados se puede mejora

la predicción y brindar diferentes puntos de vista para poder diagnosticar y facilitar el trabajo de un médico especialista y radiólogos. El área encargada del desarrollo de las diferentes técnicas de clasificación es el Machine Learning y los desarrolladores en este campo son muy solicitados. Es evidente que existe un gran incremento en el número de desarrolladores en los últimos años. Sin embargo, aún existe escases de profesionales en el área de Machine Learning.

Debido a esto se han creado diversas librerías que no se encuentran lo suficientemente optimizadas para trabajar con imágenes en formato DICOM. El portal Business Broadway realizó un estudio tomando como base datos oficiales la encuesta de Kaggle, una comunidad de profesionales en Machine Learning y Data Science. Si bien estas son herramientas que facilitan la implementación de algoritmos de clasificación y predicción, lo hacen de una manera que no se puede aprovechar al máximo los recursos computacionales (Pereira et al., 2021).

Un mal diagnostivo por parte de los radiólogos es una de los principales problemas. Al estar ellos sometidos a una mayor carga de trabajo pueden cometer errores en el diagnostico final. Esto puede concluir a generar un aumento de falsos positivos y no se trate a los pacientes de forma oportuna.

El presente proyecto tiene como finalidad la implementación de una librería que realice la tarea de clasificación de imágenes mamográficas DICOM de manera óptima. Para lo cual se utilizará los lenguajes de programación C/C++ y así garantizar la eficiencia en el consumo de recursos computacionales. La librería podrá servir de apoyo a profesionales de TI que desarrollen aplicaciones en el campo médico, así como también a radiólogos y mamógrafos.

ANTECEDENTES

Chethan, Vishwanath, Patil, & Vijetha (2020), su trabajo tuvo por objetivo la implementación de datos pulmonares para poder realizar una clasificación de nodos pulmonares. La meto-

dología que se utilizó fue la cuantitativa y un enfoque experimental, ya que se realizaron varias pruebas para mejorar el porcentaje de precisión del modelo. En los resultados se puede observar que el modelo de clasificación final tuvo una precisión del 70 %. Esto ayuda a determinar si la imagen de los pulmones de un paciente tiene cáncer o no. El autor concluye que el modelo logrado aun no es capaz de clasificar las diferentes etapas del cáncer. Sin embargo, se puede mejorar utilizando datos en tiempo real y eliminando objetos falsos.

El estudio de Chethan et al. (2020) tuvo como objetivo general realizar un proceso de clasificación multiclase de mamografías con la implementación de redes neuronales, luego adaptarlos a un modelo de incetion-v3 Google donde esta emplea resultados mediante una base de datos para detección de mamografías. La metodología utilizada es cuantitativa ya que al generar estas redes neuronales obtienen la precisión de los datos. Luego se cuantifican en los resultados obtenidos, se logra una presión de resultados un accuracy de 97 % y una pérdida o margen de error de únicamente el 0.02 % respectivamente. Concluyendo que la intervención de varios parámetros puede producir cambios radicales en el optimizador y provocando de esta forma un congelamiento en el modelo –v3.

El autor propuso en su trabajo diferentes técnicas de clasificación que pueden ser utilizadas en imágenes mamográficas. Su metodología fue experimental y cuantitativa ya que realizo varias pruebas en los diferentes algoritmos de clasificación, obteniendo varias medidas de precisión para cada uno de ellos. Como resultado se obtuvo una comparación entre la precisión, especificidad y sensibilidad para cada uno de los clasificadores. Se concluye que no todas las técnicas de clasificación se utilizan en todas las investigaciones. Así que, si se usa la técnica adecuada, esta dará mejores resultados Kamalakannan et al. (2015).

En el trabajo de Álvaro & Stefany (2020) se utilizan redes neuronales convolucionales ya que han ganado su espacio en el campo de la medicina debido a su característica de generar

clasificaciones de imágenes al discernir ciertas singularidades permitiendo la oportunidad de detectar lesiones o tumores en las imágenes la metodología utilizada será cuantitativa debido a que al aplicar algoritmos de Machine Learning con redes neuronales convolucionales de puede separar en categorías los resultados se puede establecer que la redes neuronales convolucionales son capaces de ir aumentando sus pronósticos a medida que los datos que poseen sean verídicos y con el entrenamiento necesario para implementarlos en la sección de pruebas. Obteniendo las siguientes conclusiones se establece que la red posea con un 60 % de exactitud.

El trabajo de Okamoto & Kohana (2016) tiene por objetivo la construcción de una librería en C++ que permita construir aplicaciones web. La metodología que usa es experimental y consta de 2 fases de desarrollo: la capa de HTTP y la capa de aplicación. Como resultado se tiene un pequeño framework que permite la creación de aplicaciones web. Se concluye que esta librería hace uso de un modelo basado en un solo hilo, por lo que tiene problemas de rendimiento cuando hay demasiados clientes.

Falcou (2015) describe en su trabajo el desarrollo de librerías modernas con C++. Muestra la importancia de trabajar con paralelismo en el mundo actual debido a los fuertes procesos que se ejecutan simultáneamente en nuestro entorno. Los resultados obtenidos de este trabajo revelan nuevas técnicas para garantizar el calidad, mantenibilidad y rendimiento de las aplicaciones. Se concluye que los nuevos modismos y técnicas pueden generar alto rendimiento en los códigos y más aún en el trabajo sistemas paralelos o híbridos.

Lamentablemente no existen librerías adecuadas para la clasificación de imágenes mamográficas en formato DICOM. Por lo que el presente proyecto proporcionará la creación de una librería (DicomClassifier) que permita hacer uso de diferentes técnicas para la clasificación de imágenes médicas DICOM. La librería DicomClassifier utilizará imágenes de mamografías, lo que permitirá detectar si existen anomalías. Esto brindará un apoyo significativo a desarrollado-

res inmersos en el área de imagenología médica y especialmente en modalidades de imágenes mamográficas. Así como también al personal técnico TI en el área de salud, investigadores y estudiantes de las carreras relacionadas al área de informática y de salud.

ALCANCE

Como primer paso se llevará a cabo una investigación para estructurar el estado del arte acerca de los diferentes algoritmos y técnicas de clasificación de imágenes en el área médica. Luego de tener un repositorio de artículos y libros con la información necesaria, se va a filtrar aquellos que aporten mayor valor al proyecto.

De igual manera se llevará a cabo una recopilación de artículos acerca de implementación de diferentes librerías en C/C++. Esto permitirá conocer el mecanismo más óptimo a seguir para la implementación de la librería. En este punto también se va a descartar aquellos artículos que se considere que no aportan lo suficiente al desarrollo del proyecto.

Posteriormente una vez establecidos los algoritmos se procede a implementarlos en el lenguaje de programación C/C++. Con los algoritmos listos que creará un mecanismo de comunicación que con la finalidad de integrarlos en un solo paquete y puedan ser llamados desde archivos externos. La entrada de datos que tomará la librería estará establecida solo para imágenes médicas en formato DICOM.

Como siguiente punto se evaluará la flexibilidad que esta tiene al ser usado en diferentes entornos de desarrollo. Por lo que se realizarán pequeñas pruebas para observar el comportamiento de la librería. Se utilizará los IDE Code::Blocks y QT bajo el sistema operativo Linux. De esta manera se espera que la librería pueda trabajar en distintos entornos de desarrollo sin presentar inconvenientes en su ejecución.

Finalmente, se procederá a verificar su rendimiento estableciendo como parámetros tiempos

de ejecución, es decir el tiempo que se demora la librería en procesar y devolver un resultado. Además de medir los recursos que esta usa como son el consumo de memoria y procesador.

OBJETIVOS

Objetivo General

Desarrollar una librería para la clasificación de imágenes de mamografías en formato DICOM utilizando tecnología Open Source.

Objetivos Específicos

- Recopilar estudios relacionados con técnicas de clasificación para imágenes mamográficas DICOM.
- Analizar el proceso que conlleva crear una librería para diferentes entornos.
- Analizar algoritmos de clasificación para imágenes mamográficas DICOM en C/C++.
- Construir una librería en C/C++ con algoritmos de clasificación para imágenes mamográficas DICOM.
- Comprobar la integración de la librería en diferentes entornos de desarrollo.
- Evaluar el rendimiento de la librería a través de métricas de tiempo y recursos computacionales.

FUNDAMENTOS TEÓRICOS

1.1 CÁNCER DE MAMA

El cáncer de mama se produce debido a una expansión y crecimiento de células anormales en el organismo. Son muy pequeñas, no causan síntomas y tienen muy poca probabilidad de diseminación. Si no son tratadas y diagnosticadas a tiempo pueden expandirse y ser perjudicial para el portador. Existen dos tipos de neoplasias benignas y malignas. En las benignas las masas crecen mucho pero no se dispersan he invaden otros tejidos ni partes del cuerpo. Las malignas masas se diseminan he invaden otras partes del cuerpo y viajan a través de los conductos sanguíneos para expandirse definidas como neoplasma y tumor.

Esta enfermedad no es transmisible, es decir que no puede infectar otras personas, pero si se debe a factores como el consumo de alcohol, la obesidad, el envejecimiento, consumo de tabaco. Para evitar el riesgo de contraer cáncer de mama existen métodos que se deben hacer habituales como mantener una dieta sana, evitar el consumo de alcohol y tabaco, no consumir sustancias hormonales y realizar chequeos constantes a partir de los 40 años (Sung et al., 2021).

1.2 MAMOGRAFÍA

El portal Cancer.net (2021) quien está avalado por ASCO (American Sociaty of Clinical Oncology) lo define como una radiografía que ayuda a la detección de cáncer de mama en las mujeres. Las imágenes que produce se denominan mamografías. Estas imágenes pueden mostrar pequeños tumores que no se pueden sentir. También pueden encontrar inconsistencias en la glándula mamaria. Se tiene dos tipos de mamografías las cuales son:

- ***Mamografía de detección:*** Es común esta prueba en mujeres que no presentan síntomas

o dolor en sus glándulas mamarias. La finalidad de esta mamografía es buscar de forma temprana, cuando puede ser más tratable.

- ***Mamografía de diagnóstico:*** Normalmente se da esta prueba cuando existen sospecha o aparición de algún síntoma para que sea motivo suficiente de análisis. Se puede recomendar por los siguientes motivos: i) Muestra un área sospechosa en la mamografía. ii) Cuenta con pequeñas masas en la glándula mamaria iii) Presencia de otros síntomas inusuales.

1.2.1 Masas

Son consideradas un área dentro del tejido con una estructura más densa que presenta una forma y un borde que lo destacan del resto del tejido de la mama estas masas pueden o no contener calcificaciones. Las masas se pueden producir con diferentes factores ya sea por quistes que son pequeñas bolsas llenas de líquido que no es canceroso o perjudicial o tumores sólidos que no son cancerosos es decir que no pueden invadir otras partes del tejido o expandirse a otros, pero también existe la posibilidad de que si sean un indicador de la presencia de cáncer en el tejido de la glándula mamaria (Sociedad Americana Contra el Cáncer, 2022).

1.2.2 Microcalcificaciones

Las microcalcificaciones están formadas por diferentes compuestos como sales de calcio junto con metales como magnesio, zinc y hierro, estas pueden estar presente en cualquier tejido del cuerpo humano y su aparición es más comúnmente en la mama. La mayoría de calcificaciones que se presentan generalmente son benignas a pesar de esto es difícil clasificarlos y se necesitan un estudio más detallado sobre los componentes que se encuentran a su alrededor (Cruz-Morales et al., 2012).

1.2.3 Normal

Al momento de realizarse una mamografía pueden no presentarse presencia de quistes o microcalcificaciones, pero al realizarse una mamografía es importante ya que con las mamografías que se presenten a futuro poder compararlas, esto servirá generar un diagnóstico y evidenciar posibles cambios en el tejido mamario.

1.3 IMÁGENES MÉDICAS

La finalidad de imágenes médicas es usar diferentes técnicas y procesos para simular imágenes de un cuerpo humano. Con la intervención de profesionales realizar análisis clínicos y de esta forma diagnosticar, buscar y tratar enfermedades o lesiones en un paciente. El proceso para obtener imágenes del cuerpo humano con ayuda de la tecnología es hacer que una amplia variedad de energía sea capaz de penetrar diferentes tipos de tejido con la ayuda de sensores encapsulan esa información y lo transforman en un resultado generalmente a escala de grises lo cual es posible ser interpretada de manera visual (Srivastava, 2013).

1.3.1 Estándar Health Level 7

HL7 es un estándar que a tomado popularidad y es uno de los principales empleados en los sistemas hospitalarios el cual esta formado por segmentos definidos en una norma en donde el primer segmento contiene el mensaje del encabezado seguido del remitente, destinatario, hora, fecha, nivel de seguridad la versión para elaborar el mensaje y reconocimiento del remitente. En resumen, HL7 permite una trasmisión no estructurada, enfocada a mostrar información y generar informes en base a dispositivos enlazados (Aceves, 2015).

1.3.2 Estándar DICOM

Las imágenes médicas en formato DICOM es uno de los archivos más comunes en la medicina. El formato consta de metadatos y pixeles, se almacenan ambos con un método de compresión de archivos, estos archivos forman un modelo de metadatos. Con esta información se pueden formar un ***Information Object Definition*** (IOD) que es la recopilación de varias imágenes de un paciente con sus respectivos dato. Con esta estructura no se corre el riesgo de perder información del usuario y poder hacer un seguimiento en caso de ser necesario (Caffery et al., 2021).

1.4 MACHINE LEARNING

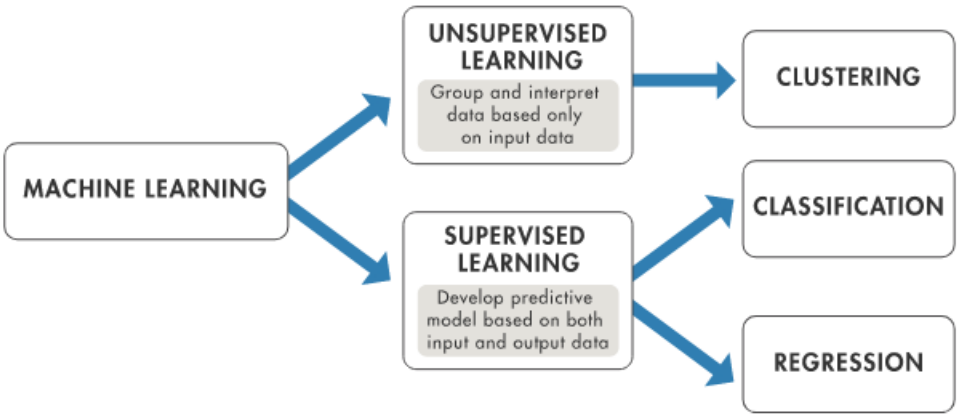
El Machine Learning es usado para la extracción de información en los últimos años ha tomado más fuerza,en la Figura 1 se puede visualizar sus respectivos campos. Esta herramienta está presente desde hace más de veinte años. El enfoque en este caso es realizar un aprendizaje desde el área médica a partir de datos observacionales o simulacros. Hoy en día el aprendizaje de maquina es utilizado en un amplio campo que va desde la detección y clasificación de imágenes de rayos X hasta la detección y clasificación de cáncer de mama o neoplastia maligna (Cruz & Wishart, 2006).

1.4.1 Aprendizaje no supervisado

El aprendizaje no supervisado o automático es una rama de Machine Leaning. Su objetivo es el estudio del comportamiento y el razonamiento humano, el cual a base de las experiencias va tomando mejores decisiones y tiene un gran campo de estudio y aplicación como problemas de clasificación, asociación, agrupamiento y rasgos de similitud (Cristina Pérez Verona & García, 2016). Estos métodos dependen mucho de los cálculos matemáticos para determinar la

Figura 1

Diagrama del aprendizaje de máquina.



Nota: *Diagrama con los campos que abarca el Machine Learning. Fuente: (MathWorks, 2018).*

similitud.

1.4.1.1 Clustering. El clustering es parte fundamental del Machine Learning. Permite a un algoritmo no supervisado o automático la capacidad de entrenar y reconocer de los datos sobre los cuales se desarrolla. Todo esto se repite en el procesamiento de un computador de forma que encuentra patrones o agrupaciones ocultos en los datos, es posible tratar grandes cantidades de información y datos en cortos periodos de tiempo con el menor número de errores (MathWorks, 2018).

1.4.2 Aprendizaje supervisado

El aprendizaje supervisado es uno de los más usados actualmente, este consiste un grupo de datos denominado datos de entrenamiento. El entrenamiento es llevado a cabo con otro grupo de datos denominados características. Estos datos de entrada etiquetados se emparejan con los de etiquetado, y a partir de este el algoritmo supervisado crea un modelo entre las características

y los datos que están en la salida para realizar predicciones para un nuevo conjunto de datos resultado de este proceso (mathworks, 2022).

1.4.2.1 Clasificación.

Dentro de los conceptos de Machine Learning supervisado, el termino de clasificación se refiere a ejecutar la clasificación va a predecir una categoría, es decir una clasificación de personas o cosas según su criterio o jerarquía (Malacara, s.f.). La utilidad del procesamiento de imágenes es muy amplia actualmente casi a cualquier campo el más común la medicina en la cual ayuda con el procesamiento de imágenes con fines de obtener diagnósticos médicos. En otras áreas se ven involucrados en el análisis de recursos naturales o en la actualidad, reconociendo terrenos mediante drones. Actualmente la clasificación de imágenes es usada en varias áreas de la inteligencia artificial ya que con la capacidad de cómputo actualmente es más accesible el uso de estas tecnologías para el desarrollo de nuestra sociedad. Por lo cual lo que a futuro se espera de la visión por computador posee altas expectativas al mejorar el Hardware que se posee actualmente.

1.4.2.2 Regresión.

Por otra parte, en conceptos de Machine Learning la regresión logística es utilizado en la estadística para fines predictivos o explicativos, se da cuando se tiene una variable que cambia su valor y puede ser categorizada entre un valor de 0 y 1 produciéndose un evento y predecir de esta forma la probabilidad de que ocurra un cierto evento (Chitarroni, s.f.).

1.4.3 Deep learning

El aprendizaje profundo ha tenido un gran avance en los últimos años en el procesamiento de imágenes, reconocimiento de voz, procesamiento natural del lenguaje, visión por computador. Su base de aprendizaje son las redes neuronales artificiales con aprendizaje de representación,

el objetivo de este es simular el comportamiento del cerebro humano. La diferencia entre los algoritmos tradicionales de aprendizaje automático y el Deep Learning es que puede aprender características y patrones de datos sin necesidad de utilizar etiquetas. Por lo tanto, Deep Leaning es una muy buena opción para conjuntos de datos complejos y a gran escala (Wen et al., 2020).

1.4.3.1 Redes neuronales.

Las redes neuronales también denominadas como artificial neural network es un algoritmo dentro de Machine Learning basado en redes neuronales biológicas. Herramienta que posee la capacidad de generalizar, aprender y procesar de manera autónoma datos, usado generalmente en trabajos de clasificación y de regresión. En las tareas de clasificación busca organizar datos de entrada en diferentes clases, mientras que en regresión busca predecir un parámetro de salida que se desconoce, su principal potencial esta en el reconocimiento de patrones y la predicción de comportamientos (Sarmiento-Ramos, 2020).

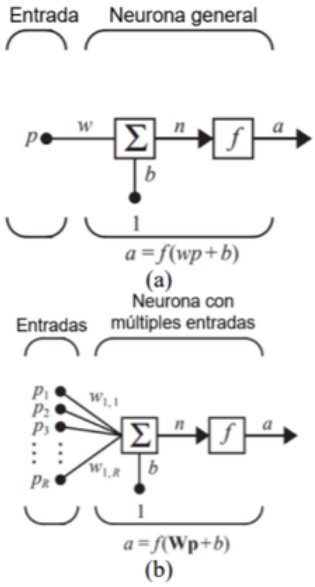
- ***Unidades.*** También llamadas como neuronas, estas son funciones que contienen valores de los pesos (weight) y el sesgo (bias). Por esta unidad pasan cada uno de los datos de entrada, los procesa haciendo uso del peso y sesgo para retornar una salida (Aggarwal, 2018). La estructura básica de una neurona se muestra en la Figura 2.
- ***Pesos.*** Los valores de los pesos son necesarios para que la red pueda aprender a generalizar un problema. Es la representación de la sinapsis de una neurona en el proceso de aprendizaje.
- ***Sesgo.*** También se la llama bias. Es un valor que se agrega después del procesamiento que realiza la neurona.

1.4.3.1.1 Función de Activación.

Las funciones de activación ayudan a transformar una señal en entrada hacia una señal de salida que a su vez servirá de entrada para la siguiente

Figura 2

Neuronas Artificiales.



Nota: *Representación de neuronas artificiales: (a) de una entrada y (b) de múltiples entradas. Fuente: (Sarmiento-Ramos, 2020).*

capa de la red neuronal (Sharma et al., 2020). Se utilizan generalmente para mapear los datos a valores más pequeños, y tratar de que la salida siga un patrón lineal. Las funciones de activación más utilizadas son:

- ***Sigmoid.*** Es una de las funciones de activación más utilizadas. Transforma los valores en el rango que va de 0 hasta 1. Se lo puede observar en la Ecuación 1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

- ***Tanh.*** Función tangente hiperbólica, se parece a la funcione sigmoidea, pero es simétrica cerca del origen. La salida de esta función resulta en distintos signos. Para esta función se utiliza la Ecuación 2.

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2}$$

- ***Relu.*** La función Relu o también llamada Unidad Lineal Rectificada ayuda para que no todas las neuronas se activen al mismo tiempo. La neurona solo se va a activar cuando el resultado sea mayor a cero. Su definición se encuentra en la Ecuación 3.

$$f(x) = max(0,x) \tag{3}$$

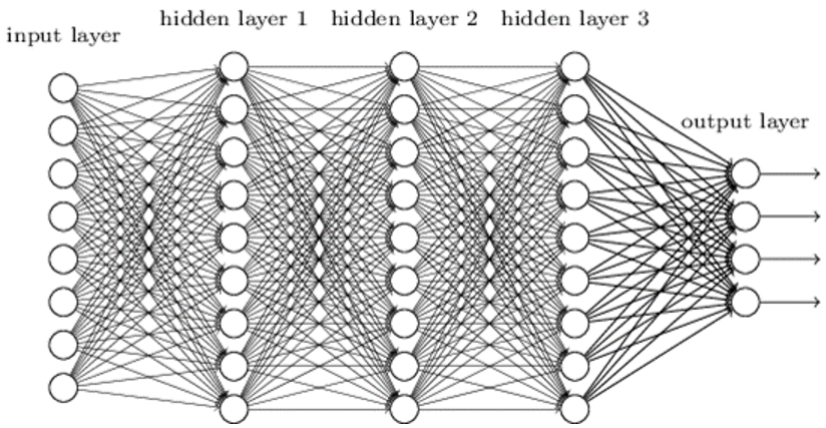
- ***Softmax.*** Esta función es una combinación entre varias funciones sigmoide. Se usa para clasificación binaria, por lo que se puede utilizar en problemas de clasificación multiclase. Devuelve la probabilidad para cada clase en un conjunto de datos. Se lo puede visualizar el la Ecuación 4.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4}$$

1.4.3.1.2 Capas. Las capas es un aumento de complejidad para las redes neuronales, lo que provoca que su salida no siga una linealidad como se puede observar en la Figura 3. Cada capa también tiene un cierto número de unidades o neuronas, la cantidad depende de la configuración. Una red neuronal tiene principalmente 2 capas, la de entrada y la de salida. Las capas extra que se agregan a la red neuronal se las llama capa oculta. Una red neuronal con más de 2 capas ocultas y a la vez con varias neuronas en su estructura, se las denomina red neuronal profunda (Suryansh S., 2018). En la Figura 3 se puede apreciar un ejemplo de una red neuronal profunda con 3 capas ocultas.

Figura 3

Red Neuronal Profunda.



Nota: *Arquitectura básica de una red neuronal profunda. Fuente: (Suryansh S., 2018).*

1.4.3.1.3 Algoritmos de optimización. El proceso de entrenamiento de una red neuronal puede tomar bastante tiempo. Existen diversas técnicas que se pueden utilizar para acelerar este proceso, entre ellas tenemos a los algoritmos de optimización. Los parámetros que suelen utilizar estos algoritmos son: i) t, que representa la iteración en la que se encuentra. ii) w, que es el peso que se va a actualizar en la iteración t. iii) α, que representa la constante de aprendizaje. iv) $\partial L/\partial W$, es el valor del gradiente L o la función de perdida w. (Raimi, s.f.)

- ***Gradient Descent.*** Es el algoritmo de optimización más básico, pero a la vez el que más se utiliza. Evalúa la forma en la que se calculan los pesos para que la función logre llegar a un valor mínimo. La Ecuación 5 utiliza el valor del peso actual junto al gradiente $\partial L/\partial W$ y esto se lo multiplica.

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t} \tag{5}$$

- ***Momentum.*** El optimizador de momentum es un algoritmo que se preocupa por los

gradientes que se utilizaron en las capas anteriores. Tiene la finalidad de reducir la alta variación del gradiente y poder suavizar la convergencia. Como se puede observar en la Ecuación 6, para ello simula un mecanismo de fricción y evita que el impulso crezca mucho. Según Géron (2019), se agrega un parámetro β que se le llama impulso y esta en el rango de 0 (alta fricción) y 1 (sin fricción). El valor típico para este parámetro β es de 0.9.

$$\begin{aligned} w_{t+1} &= w_t - \alpha m_t \\ m_t &= \beta m_{t-1} + (1-\beta)\frac{\partial L}{\partial w_t} \end{aligned} \tag{6}$$

- ***Agrad.*** Este algoritmo trata de que la tasa de aprendizaje del modelo vaya variando en cada iteración. Ecuación 7 se puede observar que se divide la constante de aprendizaje para la raíz de υ_t junto a un valor de tolerancia ε. Esto representa la acumulación de los gradientes hasta esa iteración (Géron, 2019).

$$\begin{aligned} w_{t+1} &= w_t - \frac{\alpha}{\sqrt{\upsilon_t + \varepsilon}} \cdot \frac{\partial L}{\partial w_t} \\ \upsilon_t &= \upsilon_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2 \end{aligned} \tag{7}$$

- ***RMSprop.*** En comparación con el optimizador AGRAD, este algoritmo va acumulando los gradientes que se utilizaron en las últimas iteraciones como se puede ver en la Ecuación 8. Para ello usa el promedio móvil cuadrado de los gradientes junto con el valor de fricción β.

$$\begin{aligned} w_{t+1} &= w_t - \frac{\alpha}{\sqrt{\upsilon_t + \varepsilon}} \cdot \frac{\partial L}{\partial w_t} \\ \upsilon_t &= \beta \upsilon_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial w_t}\right]^2 \end{aligned} \tag{8}$$

- ***ADAM*** Llamado también estimación adaptativa del momento, es una combinación de Momentum y RMSprop. Usa la acumulación de los gradientes del algoritmo de *Momen-*

tum y la division de la constante de aprendizaje para la raíz de υ que es el promedio móvil de los gradientes como se observa en la Ecuación 9 . Generalmente los valores que se usan son: i) $\alpha = 0.0001$ ii) $\beta_1 = 0.9$ iii) $\beta_2 = 0.999$ iv) $\varepsilon = 10^{-8}$

$$
\begin{aligned}
w_{t+1} &= w_t - \frac{\alpha}{\sqrt{\hat{\upsilon}_t + \varepsilon}} \cdot \hat{m}_t \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{\upsilon}_t &= \frac{\upsilon_t}{1 - \beta_2^t}
\end{aligned}
\tag{9}
$$

1.4.4 Clasificación de imágenes

Las clasificaciones de imágenes con aplicaciones de inteligencia artificial son capaces de resolver varios problemas en diferentes áreas como en el sector científico, industrial, seguridad o médico, permitiendo eliminar actividades costosas para evitar posibles errores humano. Para llevar a cabo la clasificación en necesario hacer una extracción de características que identifiquen a cada tipo de imagen para reconocer patrones como color, textura, densidad.

1.4.4.1 Fuzzy C-Means. La técnica de Fuzzy C-Means es una técnica no supervisada de machine learning que consiste en agrupar un conjunto de puntos en varios grupos. Ha sido muy usado para tareas de reconocimiento de patrones, clasificación, minería de datos y segmentación de imágenes (Hung & Yang, 2001).

Según Nascimento, Mirkin, & Moura-Pires (2000) para realizar el algoritmo de Fuzzy C-Means se realizan de manera iterativa una serie de pasos hasta llegar a un nivel de tolerancia válido. El primero es seleccionar un valor de c que son los grupos en los que se va a dividir el conjunto de datos, un valor de m que se encuentra entre el rango de 1.25 a 2 para poder inicializar la matriz de partición U y valor de ε para la tolerancia. Después se procede a calcular los centros

de los clusters o también llamados centroides para cada grupo que va desde $i = 1, ..., c$. Esto se puede ver en ecuación 10.

$$v_i^{(t)} = \frac{\sum_{k=1}^{n} \left(u_{ik}^{(t)}\right)^m X_k}{\sum_{k=1}^{n} \left(u_{ik}^{(t)}\right)^m} \tag{10}$$

Donde V_i es el valor del nuevo centroide. El siguiente paso consiste en actualizar la matriz de partición U lo cual se obtiene siguiendo la 11.

$$u_{ik}^{t+1} = \left[\sum_{j=1}^{c} \left(\frac{\|x_k - v_i^{(t)}\|^2}{\|x_k - v_j^{(t)}\|^2} \right)^{\frac{2}{m-1}} \right]^{-1} \tag{11}$$

Se repite este proceso hasta que el $\left|U^{(t+1)} - U^{(t)}\right| \leq \varepsilon$,o a su vez se alcance cierto número de iteraciones.

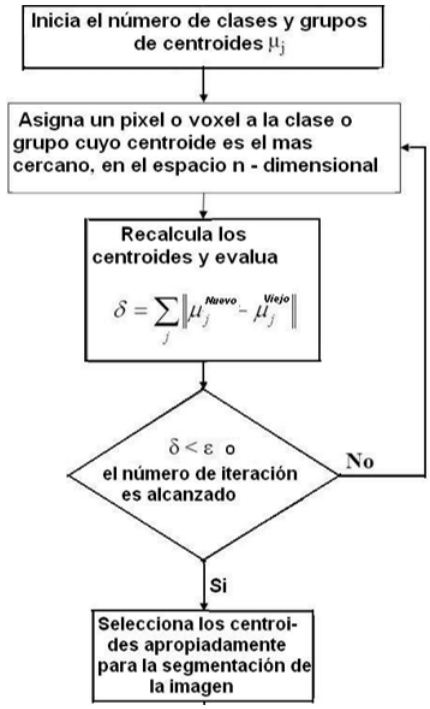
1.4.4.2 K-Means. El algoritmo de K-Means es un algoritmo de clasificación el cual es un proceso de agrupamiento que permite identificar distintos grupos dentro de una muestra separándolos por características. La estructura del algoritmo se puede apreciar en la Figura 4 y su finalidad es identificar ciertos grupos dentro de una misma muestra con sus respectivos centros, calcular el centro y las aglomeraciones que se producen a su alrededor.

El algoritmo utiliza un parámetro denominado k que es el numero clusters a identificar. Su principal objetivo es intentar minimizar la distancia que se encuentra entre los centroides y un cluster es decir la suma de las distancias de los puntos y el centroide (Francisco Jose Torres Hoyos, 2012).

K-Means utiliza el error cuadrático medio para minimizar la varianza total con la Ecuación 12:

Figura 4

Pasos de ejecución de K-Means.



Nota: *procedimiento del algoritmo de K-Means con el cálculo del error euclidiano. Fuente: (Francisco Jose Torres Hoyos, 2012).*

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i^j - c_j\|^2 \tag{12}$$

1.4.4.3 Random Forest (RF). El algoritmo RF pertenece a un grupo del que proviene denominado métodos conjuntos, se basa en un compendio de árboles de decisiones, los árboles de decisiones son métodos supervisados que se pueden usar tanto para clasificación como para regresión. Es utilizada para determinar el curso de una acción o mostrar una probabilidad siguiendo una ramificación con los árboles de decisiones bajo en concepto de divide y vencerás (Paolettia et al., 2019). Su funcionamiento es el siguiente, por cada árbol de decisión implantado con los datos de entrada se evalúa sucesivamente en cada uno de sus nodos y se divide en

conjuntos más pequeños y homogéneos, para luego ser enviados a diferentes ramas hasta llegar al final del árbol donde se lo conoce como hoja, la que nos dice a qué clase pertenece el dato y se realiza una votación entre las demás hojas para darnos la clase final al que pertenece el dato, la desventaja de RF es que debido a que se incrementa el número de árboles,llega a ser difícil de procesar para todos los parámetros (Paolettia et al., 2019).

- ***Etapa de entrenamiento.*** En esta etapa el algoritmo busca optimizar parámetros de las funciones en cada pequeño segmento que crea un nuevo árbol, luego utiliza la siguiente Ecuación 13 de ganancia de información,

$$I_l = H(j) - \sum_{i\varepsilon 1,2} \frac{\left|S_j^i\right|}{\left|S_j\right|} H\left(S_i^j\right) \tag{13}$$

  Donde S es el conjunto de muestras por dividir en el nodo y S^i son los dos subconjuntos que se crean en el nodo, esta función tiene el objetivo de medir la entropía del grupo y varia del tipo de problema a analizar (Medina-Merino & Ñique-Chacón, 2017).

- ***Exactitud de disminución de medida.*** En la elaboración de RF se utiliza Bootstrap, con el objetivo de entrenar al árbol que se agrega en el bosque a generar, a los casos que no son considerados para el entrenamiento se los denomina ***out-of-bag (OOB)***. Esto ayuda a estimar un error en la clasificación y sirve para realizar una estimación del valor de la variable, el comportamiento de la variable funciona de la siguiente forma se escoge un error de clasificación OOB, luego se toma una al azar y se permutan sus valores en el entrenamiento, ocasionando que esa variable tenga diferencia con el valor tomado al azar. Luego vuelve a calcular el OOB. Y lo compara con el error calculado inicialmente por lo tanto si su valor cambia se afirma que esa variable es importante ya que representa un desequilibrio en la medición. Este proceso se repite para todas las variables y luego

se la ordene acuerdo a los cambios suscitados en los errores OOB (Medina-Merino & Ñique-Chacón, 2017).

- ***Gini de distribución media.*** Otra forma de estimar la importancia de las variables del modelo RF es a través de Gini que es un criterio que consiste en seleccionar una variable en cada árbol creado y realiza una disminución de la medida, la importancia del árbol se establece como la suma de los decrementos atribuidos a esta variable (Medina-Merino & Ñique-Chacón, 2017).

1.4.4.4 K-nearest neighbors (KNN).

El algoritmo de KNN o k-vecinos más cercanos, este difiere a que se expande al vecino más cercano a k como valor en la toma de la decisión y la clasificación permite al algoritmo utilizar mucha información y de cierto modo omite el proceso de aprendizaje a diferencia con los otros algoritmos de clasificación. Este algoritmo es uno de los métodos más antiguos y su funcionamiento es simple se toma una categoría de la muestra y se la prueba si se mantiene el conjunto de entrenamiento y la métrica de la distancia sin cambios esta se clasifica, por ejemplo, para un conjunto de datos E y la instancia del vecino más cercano es X entonces la categoría de Y es el resultado de la decisión, el proceso de decisión específico se muestra en la Ecuación 14.

$$g_j(X) = ming_i(X)\, i = 1,2,...,C \tag{14}$$

La regla del vecino más cercano parte desde los dos aspectos que son: La convergencia y el error de generalización. Para el cálculo del error del vecino más cercano se puede comprender como la probabilidad que el punto a medirse sea diferente de la categoría del punto del vecino más cercano y su tasa de error se representa en la siguiente Ecuación 15.

$$P(error) = 1 - \sum_{c \varepsilon Y} P(c \mid x) P\left(c \mid x'\right) \tag{15}$$

Se asume que el calculo de la muestra es independiente e igualitaria. Una muestra x se puede encontrar dentro del rango de distancia d alrededor de x de modo que en la Ecuación 16 y Ecuación 17 se muestra el clasificador bayesiano.

$$c^* = arg\, max_{c \varepsilon Y} P(c \mid x) \tag{16}$$

Por lo tanto

$$P(error) \leq 1 - \sum_{c \varepsilon Y} P^2(c^* \mid x) \tag{17}$$

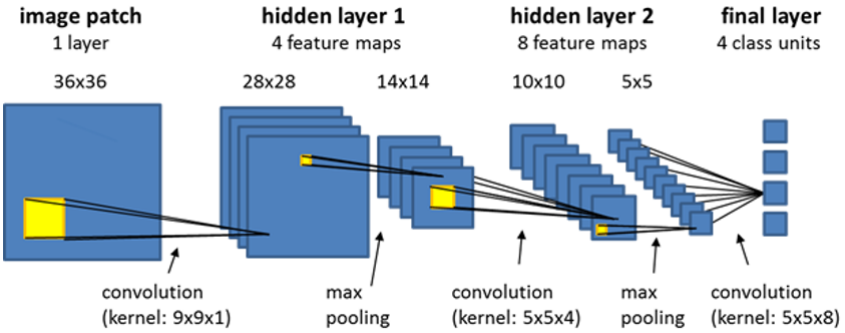
En conclusión, el vecino más cercano no es únicamente la construcción también se debe considerar la tasa de error generalizada no debe ser más del doble que la tasa de error bayesiana.

1.4.4.5 Convolutional Neural Networks (CNN). CNN son un tipo de redes neuronales multicapa. Está enfocado a campos de la visión por computador y procesamiento de lenguaje natural. Una red neuronal convolucional está hecha un conjunto de una o varias capas de convolución y agrupamiento como se representa en la Figura 5, seguida de una o varias capas totalmente conectadas, además de una capa de salida. La base de la red convolucional es el conjunto de capas de convolución (Ghosh et al., 2020).

- ***Capa convolucional.*** Esta capa contiene una serie de kernels o también llamados filtros que van a ser aplicados sobre los datos de entrada. La salida de esta capa genera un mapa con las características y que sirve de entrada para la siguiente capa de la red neuronal.
- ***Kernel.*** El kernel es una matriz de valores discretos, cada valor de la matriz se lo puede describir como peso del kernel. Durante el proceso de entrenamiento el kernel aprende a

Figura 5

Redes Neuronales Convolucionales.



Nota: *Estructura básica de las redes neuronales convolucionales. Fuente: (Suryansh S., 2018).*

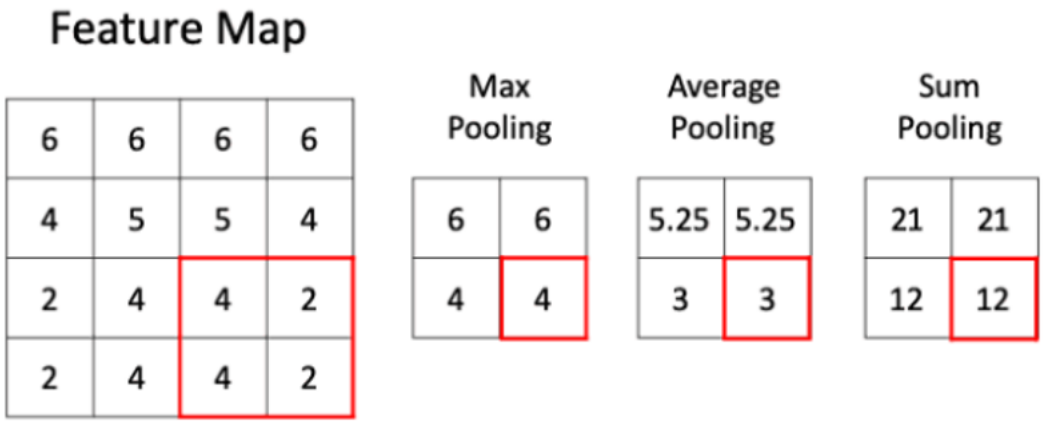
extraer información útil de los datos de entrada (Ghosh et al., 2020).

- ***Capa de agrupamiento.*** La capa de agrupamiento o en ingles pooling layer como se muestra en le Figura 6, sirve para reducir el tamaño del mapa de características. De manera similar a la capa de convolución se aplica una especie de filtro que mapea la matriz de entrada y aplica una operación sobre cada segmento de la matriz. Las operaciones que se pueden aplicar generalmente son el máximo, la media o la suma (Cowley, s.f.).
- ***Capa totalmente conectada*** Esta capa equivale a la capa oculta de una red neuronal normal. Los resultados de la capa anterior se enlazan a todas las neuronas o unidades de la siguiente capa (Cowley, s.f.).

1.4.4.6 SVM. Las Máquinas de soporte vectorial (Support Vector Machine o SVM), son una técnica de clasificación que está basado en el aprendizaje supervisado. Esta técnica intenta resolver el problema de clasificar un conjunto de puntos en 2 grupos. Visto desde una perspectiva conceptual, se puede decir que, dado un conjunto de datos de entrada, estos se ubican en un plano muy característico. Dentro de este plano se construye una recta lo que permite dividir el conjunto de puntos en 2 clases o grupos (Cortes & Vapnik, 1995).

Figura 6

Capa de agrupamiento



Nota: *Funcionamiento de la capa de agrupamiento y sus respectivas operaciones. Fuente: (Hussain, s.f.).*

- ***Kernel.*** Dentro de SVM se puede resolver problemas de clasificación en los cuales sus datos sigan una tendencia lineal, es decir que se puedan dividir estos datos a través de una línea recta. El Kernel o "Truco del Kernel" permite separar los datos que no se pueden dividir a través de una recta. Para ello existen diferentes funciones que mapean los datos hacia un espacio dimensional superior durante el proceso de clasificación. Esto permite poder dividir las clases de una manera más correcta (Kaestner, 2013).

  - ***linear*** Representa el aprendizaje para 2 clases como se muestra en la Ecuación 18.

$$K(x_1, x_2) = (x_1, x_2) \tag{18}$$

  - ***Polinómica*** Es útil cuando se trata de procesamiento de imágenes. El parámetro d es el grado del polinomio como se puede ver en la Ecuación 19.

$$K(x_1, x_2) = (x_1 x_2 + 1)^d \tag{19}$$

  - ***Sigmoide*** Se puede utilizar como puente cuando se usa redes neuronales. También

representa un kernel solo para ciertos valores β_0 y β_1 en la Ecuación 20.

$$K(x_1, x_2) = tanh\left(\beta_0 x_1^T x_2 + \beta_0\right) \tag{20}$$

– ***RBF*** También llamada función base radial o gaussiana. Es un kernel de propósito general se usa cuando no se conoce la tendencia de los datos. Sigma representa el ancho del kernel se encuentra en la Ecuación 21.

$$K(x_1, x_2) = exp\left(-\gamma \|x_1 - x_2\|^2\right) \tag{21}$$

Para valores de $\gamma > 0$, se suele utilizar la Ecuación 22:

$$\gamma = \frac{1}{2\sigma^2} \tag{22}$$

- ***Parámetros de ajuste:***

  Regularización. También conocido como parámetro C, sirve para que el optimizador de SVM evite clasificar erróneamente cada dato en el conjunto de entrenamiento. Cuando el valor de C es alto, el margen de la recta será más pequeño. En caso de que el valor de C sea pequeño, el margen será grande (Zoltan, s.f.). Gamma, el parámetro γ nos indica la influencia que tienen los datos al momento de definir la recta. Un γ alto considera datos que están cerca de la recta, mientras que un valor bajo considera a los datos que se encuentran a una mayor distancia (Zoltan, s.f.).

- ***C-SVC:*** En un conjunto de datos de entrenamiento $X_i \in R_p, i = 1, ..., n$, que pertenecen a 2 clases. Y un vector $y \in \{1, -1\}_n$. C-SVC intenta resolver el siguiente problema de optimización en la Ecuación 23:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{sujeto a} \quad & y_i (w^T \phi(x_i) + b) \geq 1 - \xi, \\ & \xi_i \geq 0, 1, ..., l. \end{aligned} \tag{23}$$

Donde $\phi\,(xi)$ transforma los datos xi hacia un espacio dimensional mayor y $C > 0$ es el parámetro de regularización,como se muestra en la Ecuación 24. El objetivo es lograr encontrar los valores de $w \in R_p$ y $b \in R$, para la función de decisión y así poder predecir los nuevos datos de entrada (Chang & Lin, 2011).

$$sgn\left(w^T \phi\,(x) + b\right) \tag{24}$$

ν-SVC: El ν-support vector classification. Introduce el parámetro $\nu \in (0, 1]$. Se demuestra que ν una cota superior de la fracción de errores de entrenamiento y una cota inferior de la fracción del vector de soporte (Chang & Lin, 2011). Este modelo de SVM intenta resolver el siguiente problema de optimización en la Ecuación 25:

$$\begin{aligned} \min_{w,b,\xi,p} \quad & \frac{1}{2} w^T w - vp + \frac{1}{l} \sum_{i=1}^{l} \xi_i \\ \text{sujeto a} \quad & y_i (w^T \phi(x_i) + b) \geq p - \xi, \\ & \xi_i \geq 0, i = 1, ..., l, \qquad p \geq 0. \end{aligned} \tag{25}$$

La función de decisión para este modelo se muestra en la Ecuación 26.

$$sgn\left(\sum_{i=1}^{l} y_i \alpha_i K\,(x_i, x) + b\right) \tag{26}$$

- ***Distribution Estimation (One-class SVM).*** Propuesto por Schölkopf, Platt, Shawe-Taylor, Smola, & Williamson (2001). Para poder estimar el soporte para una distribución de alta dimensionalidad. Dado un conjunto de datos xi que no tienen ninguna información

sobre su clase. One-class SVM intenta resolver el siguiente problema en la Ecuación 27.

$$\begin{aligned} \min_{w,\xi,p} \quad & \frac{1}{2}w^T w - p + \frac{1}{vl}\sum_{i=1}^{l}\xi_i \\ \text{sujeto a} \quad & w^T \phi(x_i) \geq p - \xi, \\ & \xi_i \geq 0, i = 1, ..., l. \end{aligned} \tag{27}$$

La función de decisión se representa en la Ecuación 28.

$$sgn\left(\sum_{i=1}^{l}\alpha_i K(x_i, x) - p\right) \tag{28}$$

1.5 LENGUAJES DE PROGRAMACIÓN Y HERRAMIENTAS

1.5.1 C++

Luis Joyanes e Ignacio Zahonero en su libro Programación en C Metodología, algoritmos y estructura de datos, definen al lenguaje C como:

"C es un lenguaje de programación de propósito general asociado, de modo universal, al sistema operativo UNIX. Sin embargo, la popularidad, eficacia y potencia de C, se ha producido porque este lenguaje no está prácticamente asociado a ningún sistema operativo, ni a ninguna máquina, en especial. Ésta es la razón fundamental, por la cual C, es conocido como el lenguaje de programación de sistemas, por excelencia" (Aguilar et al., 2005).

1.5.2 Cmake

Es una herramienta de código abierto con la finalidad de crear, probar y empaquetar software. Ayuda a la ejecución y configuración de archivo directamente con el compilador facilitando y agilizando el proceso de adaptar un entorno para su ejecución (*CMake*, s.f.).

1.5.3 Dcmtk

Es un compendio de librerías que contiene gran parte de los formatos DICOM, ayuda a examinar, construir y convertir archivos también lo acompaña el código fuente completo y se encuentra disponible como código abierto (*DCMTK - DICOM Toolkit*, 2022).

1.5.4 QT

Según la documentación oficial, QT no es un lenguaje de programación por sí solo. Es un framework para el desarrollo de aplicaciones multiplataforma. Utiliza un preprocesador MOC (Meta-Object Compiler) que extiende las funcionalidades del lenguaje C++. Esto permite que el código escrito en QT pueda sea compatible con los compiladores C++ como GCC, ICC, MinGw y MSVC (QT.io, s.f.).

1.5.5 Code::Blocks

Es un IDE de código abierto basado en C/C++ y Fortran, que cubre las necesidades de los usuarios, con facilidad de expandirse y lograr realizar las configuraciones que se requieran. Se puede añadir cualquier funcionalidad instalando o codificando un complemento (*Code::Blocks - Code::Blocks*, s.f.).

1.5.6 Librería:

Son un conjunto de métodos y clases de utilería que pueden ser utilizadas por el desarrollador para obtener una determinada funcionalidad (Ninla Elmawati Falabiba, 2019). Al poder contar con las librerías, se puede hacer uso de una gran variedad de funciones que ayudan a la reutilización de código y mejora el modularidad del software o programa que se desarrolla.

1.5.7 Licencia Massachusetts Institute of Technology (MIT)

La librería DicomClassifier se define con una licencia MIT software de tipo gratuito. Permite al usuario final la autorización de modificas,fusionar,copiar,distribuir. La licencia no contiene cláusulas como publicidad. Por la presente se concede permiso, sin cargo, a cualquier persona que obtenga una copia de este software y los archivos de documentación asociados a la librería DicomClassifier, para tratar en el Software sin restricciones, incluidos, entre otros, los derechos usar, copiar, modificar, fusionar, publicar, distribuir, sublicenciar y/o vender copias del Software, y para permitir a las personas a quienes se les provisto para hacerlo.

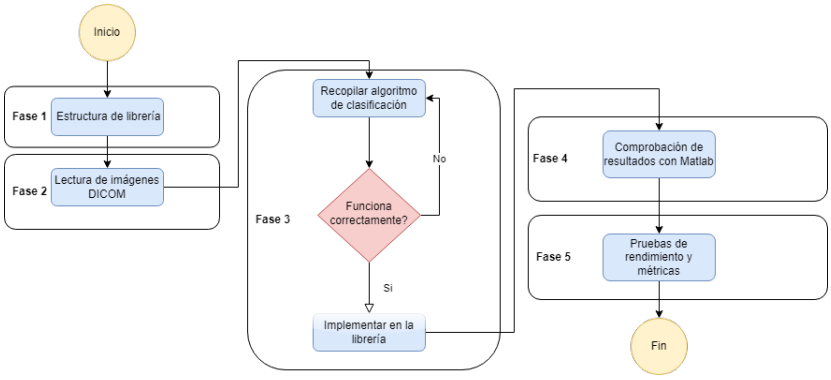
MARCO METODOLÓGICO

2.1 DISEÑO Y CONSTRUCCIÓN DE LA LIBRERÍA

Para el desarrollo de la librería que permita manejar imágenes médicas DICOM y pueda manejar diferentes técnicas de clasificación, se siguió la estructura definida en la Figura 7 que consta de cinco fases. i) Estructura de la librería, aquí definimos la estructura de carpetas y archivos para la librería DicomClassifier. ii) Lectura de imágenes DICOM, donde nos centramos en crear un módulo para leer y extraer la matriz de imagen del archivo DICOM. iii) Implementación de técnicas de clasificación, fase donde se hizo una recopilación de algoritmos de clasificación en el lenguaje C++ y se hizo las respectivas modificaciones para que se pueda adaptar a la librería DicomClassifier. iv) Comprobación con Matlab, donde el objetivo fue comparar los resultados de la librería DicomClassifier con Matlab y asegurarnos de que la implementación de las diferentes técnicas fue correcta. v) Pruebas de rendimiento y métricas, en esta fase consiste en evaluar el rendimiento de las técnicas implementadas y analizar los distintos resultados.

Figura 7

Fases de desarrollo de la librería



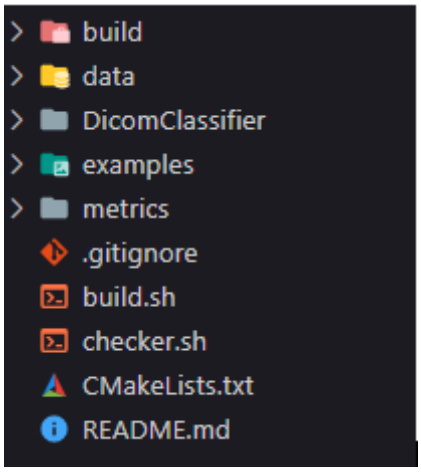
Nota: *Fases del desarrollo de la librería. Elaborado por: Los autores.*

2.1.1 Fase 1: Estructura de la librería

La estructura de carpetas general de la librería que se aprecia en la Figura 8, consta de las siguientes carpetas. La carpeta build se encuentran todos los archivos que pasaron bajo el proceso de compilación. Esta carpeta se enlaza con los nuevos proyectos de C++ que necesiten usar esta librería, por defecto esta carpeta no existe, pero se puede generar de manera automática ejecutando el script build.sh. La carpeta data contiene una imagen en formato DICOM que sirve de muestra. En la carpeta examples están diferentes ejemplos acerca de cómo utilizar la librería para leer imágenes médicas DICOM y utilizarlas como entrada para poder realizar un modelo de clasificación utilizando las diferentes técnicas implementadas. El directorio metrics tiene el código fuente para poder obtener métricas de rendimiento como son el consumo de memoria, CPU y tiempos de ejecución.

Figura 8

Estructura de archivos



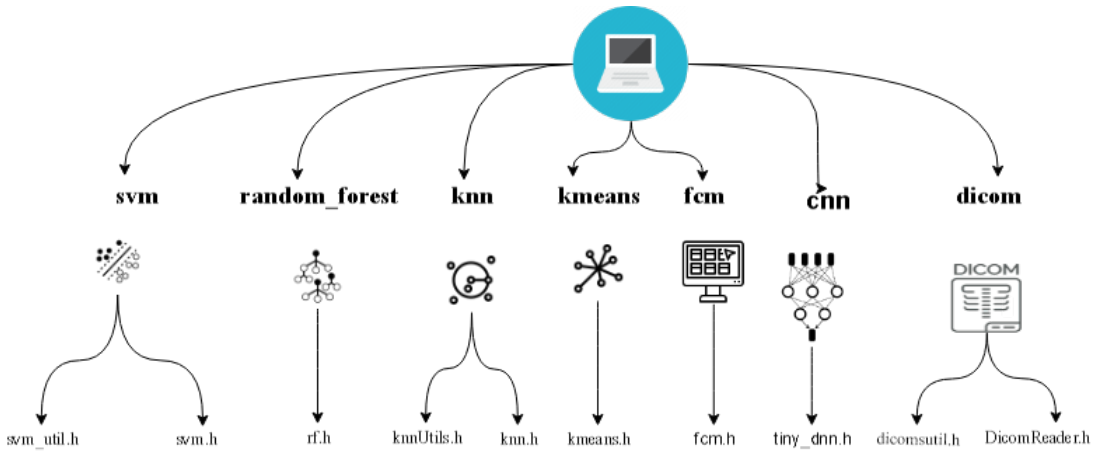
Nota: *Estructura de carpetas para el proyecto. Elaborado por: Los autores.*

En el directorio DicomClassifier está todo el código fuente de la librería, esta se encuentra dividida por diferentes módulos que pueden ser importados desde un nuevo proyecto de C++.

Estos módulos a su vez se dividen en base a la tarea a realizar (lectura de imagen o técnica de clasificación), el detalle de organización se puede observar en la Figura 9. También contiene las cabeceras que se necesitan importar para poder hacer referencia a los distintos métodos que contiene la librería DicomClassifier.

Figura 9

Estructura de las cabeceras.



Nota: *Estructura de cabeceras para acceder a la funcionalidades de la librería. Elaborado por: Los autores.*

La librería proporciona dos scripts para facilitar algunas tareas. i) *checker.sh* ayuda a verificar que las dependencias necesarias para construir la librería DicomClassifier estén instaladas en el sistema operativo. ii) *build.sh* hace uso de CMake para automatizar el proceso de compilación y construir la carpeta build de la librería.

2.1.2 Fase 2: Lectura de imagen DICOM

Para la creación del módulo que nos ayuda con el manejo de imágenes DICOM el primer paso fue implementar la librería dcmtk. Esta librería contiene un conjunto de herramientas que facilitan la lectura de archivos DICOM para posteriormente obtener la matriz de imagen. Una vez integrado dcmtk a la librería DicomClassifier seguimos el proceso descrito en el Algoritmo

1 para poder extraer la matriz de imagen y guardarla en un puntero que regresa al usuario para su posterior uso.

Algoritmo 1 Lectura de imagen DICOM

Input: Profundidad de Color;
Output: Matriz de imagen.

```
w ← width
h ← height
Inicializar un puntero con las dimensiones de la imagen w y h
if imagen is NULL then
    return NULL
end if
if status is NORMAL then
    if imagen is GRISES then
        Extraer Pixel desde imagen
        for i = 0 hasta w do
            for j = 0 hasta h do
                puntero = Pixel_ij
            end for
        end for
    else
        Error en la imagen
    end if
end if
return Puntero imagen
```

Para facilitar el acceso a esta funcionalidad y abstraer de mejor manera el manejo de la imagen se creó una clase llamada DicomReader. Esta clase recibe como parámetro la ubicación donde se encuentra el archivo DICOM del que se va a extraer la imagen. Contiene los siguientes métodos:

getWidth(): Devuelve el ancho de la imagen.

getHeight() : Devuelve el alto de la imagen.

getImageArray(depth) : Devuelve un putero con los valores de la matriz de imagen.

- ***depth*** Profundidad de color (12 bits para imágenes DICOM).

get<type>ImageMatrix(depth): Funciones auxiliares para devolver la matriz de imagen utilizando vectores.

- ***depth*** Profundidad de color (12 bits para imágenes DICOM).

De manera interna cuando se crea un objeto ***DicomReader*** esta clase guarda una referencia hacia una instancia de ***DicomImage*** que es propio de la librería dcmtk, además de guardar las propiedades de alto y ancho de la imagen. También se registra los respectivos decodificadores para que se pueda leer imágenes que tengan algún tipo de compresión y no tener ningún tipo de problemas durante la lectura de estos archivos. Para poder hacer uso de esta clase y sus métodos en otro proyecto necesita importar la cabecera ***"dicom/DicomReader.h"*** Además de modulo ***dicom/DicomReader.h*** se creó un conjunto de utilidades que nos facilitan la lectura de grupos de imágenes. Estas utilidades puedes ser llamadas importando el archivo ***"dicom/dicomutils.h".*** Las funciones definidas son las siguientes:

getDicomFilesPath(path): Devuelve un vector con la ruta de todas las imágenes DICOM que se encuentran en un directorio.

- ***path*** Directorio donde se encuentran el conjunto de imágenes.

makeDicomFilesPath(dicomDirectory, summaryFile): Devuelve un vector con la ruta de todas las imgenes DICOM que se encuentran en el directorio.

- ***dicomDirectory*** Directorio donde se va a escanear las imágenes.
- ***summaryFile*** Archivo de texto CSV con el formato. NOMBRE IMAGEN,ETIQUETA.

makeDicomLabels(summaryFile): Devuelve un vector con todas las etiquetas que pertenecen a cada imagen. El parámetro ***summaryFile*** hace referencia al archivo de texto CSV con el formato: NOMBRE IMAGEN,ETIQUETA.

saveFile(dataList, delimiter, filename): Guarda un vector de etiquetas en un archivo de texto.

- ***dataList*** Vector con los valores de las etiquetas.

- ***delimiter*** Delimitador que se utiliza para separar los valores de las etiquetas.
- ***filename*** Nombre del archivo que se va a guardar.

saveData(data, delimiter, filename, header): Crea un archivo CSV en base a los parámetros dados.

- ***data:*** matriz en formato vector con los valores a guardar. Si se utiliza cabecera el ultimo valor representa la etiqueta a la que pertenece cada punto.
- ***delimiter*** Delimitador que se va a utilizar para separar los datos.
- ***filename*** Nombre del archivo que se va a guardar.
- ***header*** Valor true o false que representa si se guarda el archivo con cabecera o no. El nombre de la última columna representa a las etiquetas de cada punto y se llama LABEL.

Este archivo cuenta con otras utilidades para modificar la estructura de datos en algunos algoritmos. Su funcionamiento se detallará más adelante.

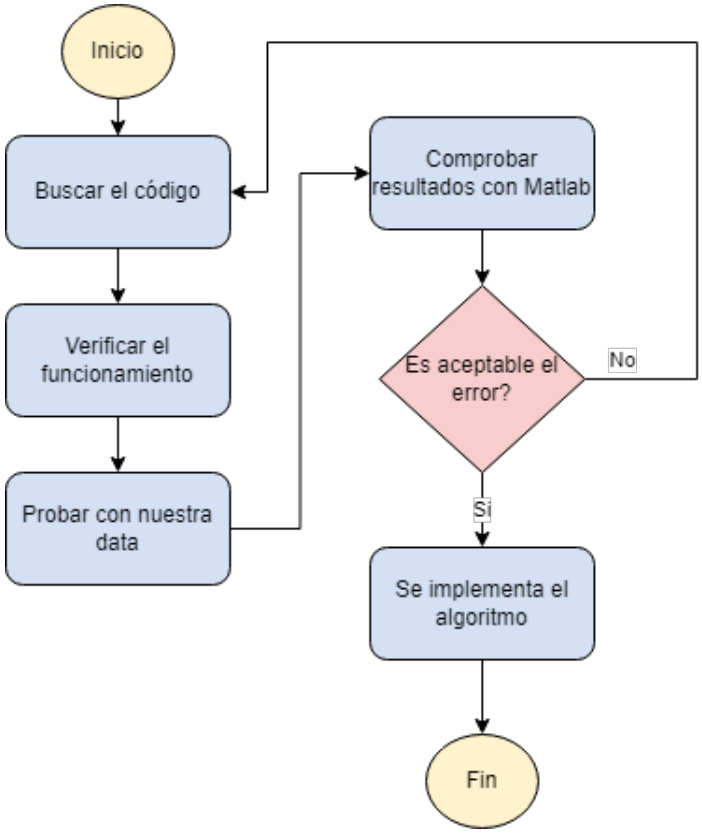
2.1.3 Fase 3: Implementación de técnicas de clasificación

Las técnicas de clasificación son el núcleo de la librería DicomClassifier, estas procesan el dato de entrada que será una imagen DICOM y dependiendo de las necesidades de los usuarios escoger la técnica que más se ajuste a sus necesidades. El proceso que se realizó para la implementación de los algoritmos se puede observar en la Figura 10 .

Utilizando como base el análisis del estado del arte acerca de las distintas técnicas que se usan en la clasificación de imágenes mamográficas. Se procedió a realizar una recopilación del código fuente de las distintas técnicas de clasificación a través del servicio de GitHub. Después de obtener el código, se configura el entorno para que se ejecute y poder analizar el funcionamiento. En algunos casos pueden ocurrir inconsistencias debido a una mala implementación

Figura 10

Implementación de técnicas



Nota: *Proceso para implementar una técnica de clasificación a la librería DicomClassifier. Elaborado por: Los autores.*

o problemas con distintas dependencias. Cuando la técnica recopilada funciona de manera correcta, se utiliza la matriz de imagen DICOM en escala de grises. Al realizar esto obtenemos un resultado, el mismo se compara con la herramienta Matlab, ya que este software tiene ya implementado en su gran mayoría estos algoritmos. En caso de que el algoritmo resulte en valores diferentes a los calculados por Matlab, se realiza las correcciones necesarias. Cuando el margen de error con respecto a los resultados de Matlab son muy similares, entonces se procede a implementarlo en la librería DicomClassifier. Hay que tener en cuenta que en este punto el código fuente de las distintas técnicas pertenecen a aplicaciones generalmente de consola o están hechos para que sean utilizados por otros lenguajes, por lo que el proceso de implemen-

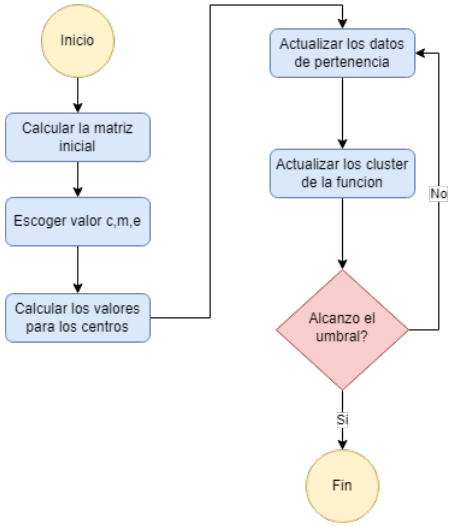
tación consiste en adaptar dicho código para que pueda ser utilizado en conjunto con la librería DicomClassifier de una manera sencilla.

2.1.3.1 Fuzzy C-Means..

Esta basado en C-Means, es un algoritmo para clustering de partición difusa. Se utiliza en la minería de datos, reconocimiento de patrones, segmentación y agrupamiento de imágenes. Esta técnica sigue el procedimiento descrito en la Figura 11. Asigna a cada dato un grado de pertenencia dentro de cada cluster y por consecuente un dato puede pertenecer parcialmente a más de un cluster de tal modo que los datos pertenecen en algún grado a todos los cluster variando su valor entre 0 y 1.

Figura 11

Diagrama Fuzzy C Means



Nota: *Diagrama de Fuzzy C Means. Elaborado por: Los autores.*

La fuente original del algoritmo de Fuzzy C-Means consistía en una aplicación de consola que realizaba el proceso de clustering. Como entrada utilizaba un archivo de texto con la cantidad de datos, número de clusters, cantidad de dimensiones, coeficiente fuzzyness y el criterio de parada. En el Algoritmo 2 se puede observar los pasos que se realiza para clasificar una serie de puntos.

Algoritmo 2 Algortimo Fuzzy C-Means

Input: Matriz de imagen;
Output: centroides c , U .

1: $c \leftarrow clusters$
2: $E \leftarrow E = 0{,}001$
3: $maxIteraciones \leftarrow (e.j.maxIteraciones = 100)$
4: **for** $t = 1$ hasta $maxIterations$ **do**
5: Actualizar la matriz U
6: Calcular los nuevos centros de los clusters V^t
7: Calcular la nueva función J_m^t
8: **if** $\left(abs\left(J_m^t - J_m^{t-1}\right) < E\right)$ **then**
9: break;
10: **else**
11: ${J_m}^{t-1} = {J_m}^t$
12: **end if**
13: **end for**
14: **return** U

Para que este algoritmo funcione en conjunto con la librería DicomClassifier, se creó una clase llamada *FCM*, que toma como parámetros el valor del coeficiente ***fuzzyness*** y el criterio de parada ***épsilon.*** La clase *FCM* cuenta con los siguientes métodos ayuda a separar la lógica en distintas funcionalidades.

init(data, clusters, num_points, num_dimensions): Inicializa los valores que necesita el algoritmo y se encarga de la validación. Cuenta con los parámetros:

- ***data*** Puntero con los datos que se van a clasificar, el conjunto de datos de imágenes DICOM.
- ***clusters*** Cantidad de grupos o clusters que se van a clasificar el conjunto de datos.
- ***num_points*** Número total de datos que se van a evaluar, en este caso el total de imágenes.
- ***num_dimensions*** Número de dimensiones o características que tiene cada dato, en este caso la dimensión total de las imágenes.

eval(): El paso de evaluación, lo que realiza es el cálculo de los vectores centroides y calcular la diferencia para de esta forma quedarse con el menor valor y actualizar la condición de salida.

getCenters(): Devuelve puntero con los valores de los centroides.

getMembershipMatrix(): Obtiene un puntero con los valores de la matriz de pertenencia.

saveClusters(prefix): Guarda en archivos de texto los índices de los puntos que pertenecen a cada cluster. El parámetro ***prefix*** es el prefijo del nombre con el que se guarda el archivo.

saveMembershipMatrixU(name): Guarda la matriz de pertenencia en un archivo de texto. El parámetro ***name*** es el nombre del archivo con el que va a guardar.

saveCenters(name) Guarda los valores de los centroides en un archivo de texto. El parámetro ***name*** es el nombre del archivo con el que va a guardar.

Además, para poder convertir la matriz de imagen DICOM en un tipo de dato válido para esta clase se puede hacer uso de ***"dicom/dicomutils.h"*** y utilizar el método:

asFCMPointsData(data, points, dims): Toma la data y lo transforma en un formato válido para enviarlo como entrada para FCM. Tiene los parámetros:

- ***data*** Puntero que contiene la matriz con los valores del conjunto de datos.
- ***points*** Número total de datos que contiene la matriz del conjunto de datos.
- ***dims*** Número total de características o dimensiones que tiene cada dato a evaluar.

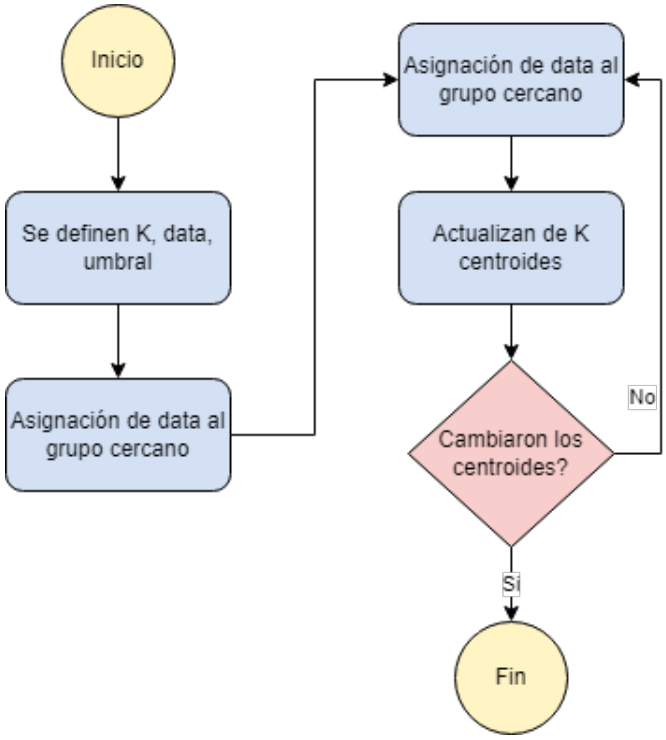
Para hacer uso del algoritmo de Fuzzy C-Means se utiliza la cabecera ***"fcm/fcm.h".***

2.1.3.2 K-Means. Este es un algoritmo de clasificación no supervisado, básicamente agrupa objetos en k grupos basándose en sus características se similitud luego este también realiza una suma de distancias entre objetos y usa la distancia cuadrática. Este algoritmo termina su proceso cuando ya no existe un cambio en los centroides que se calculan en cada iteración, podemos ver el comportamiento del algoritmo en la Figura 12

La implementación original de K-Means es una aplicación de consola que toma un archivo de texto con los puntos y calcula los respectivos valores de los centroides. Para implementar

Figura 12

Diagrama K-Means.



Nota: *Diagrama de K-Means. Elaborado por: Los autores.*

esta técnica en la librería DicomClassifier se desacoplaron las clases principales que ejecutan el algoritmo para agregarlas como un módulo a la librería DicomClassifier. Utiliza la clase ***Point*** para modelar un punto del conjunto de datos y los valores que corresponden a cada una de sus dimensiones. La clase ***Cluster*** se encarga de la agrupación de los distintos puntos del conjunto de datos, también de evaluar los valores de locadas respectivos centroides. Por último, la clase ***KMeans*** realiza la ejecución general del algoritmo y expone los mecanismos para acceder a los valores de los centroides. En el Algoritmo 3 se analiza el funcionamiento general y la estructura del algoritmo.

Para utilizar el algoritmo de K-Means, necesita la cabecera ***"kmeans/kmeans.h"*** , esto nos data acceso a la clase ***KMeans*** que se encarga de la ejecución de esta técnica.

La clase ***KMeans*** toma como parámetros de entrada el valor de k, la cantidad de iteraciones

Algoritmo 3 Algoritmo KMeans

Input: X= Matriz de imagen;
Output: C= K cluster centroides

Selecciona un subconjunto aleatorio C de X como el conjunto inicial de centros de clústeres
while true **do**
    for $(i \leq N; i \leq N; i = i+1)$ **do**
        Asignar x_i al cluster mas cercano
        $m[i] = argmin \|x_i - c_k\|^2$
    end for
    Recalcular los clusters
    for $(k = 1; k \leq K; k = k+1)$ **do**
        el grupo S_k contiene el conjunto de puntos X_i que están más cerca del centroide C_k
        $S_k = \{X_i \mid m[i] = k\}$;
        Calcular el nuevo centro C_k como la media de los puntos a los que pertenecen S_k
        $C_k = \frac{1}{|S_k|} \sum X_i$
    end for
end while
return C

y el directorio donde guardan los resultados. Posee los siguientes métodos:

run(points): Ejecuta el algoritmo tomando como entrada el conjunto de datos points. El tipo de dato es un vector de la clase ***Point.*** El parámetro ***points*** representa un dato con sus características o dimensiones, son todos los datos a evaluar.

getClustersValues(): Devuelve un vector con los valores de los clusters.

saveClusters(prefix): Guarda los valores de los centroides de los distintos clusters en un archivo. El parámetro ***prefix*** es el prefijo del nombre del archivo.

savePoints(points): Guarda el número de cluster al que pertenece cada punto del conjunto de datos en un archivo de texto. Toma como base el índice del punto para guardar el valor del cluster al que pertenece. El parámetro ***points*** es un vector de ***Point*** que contiene la información del cluster al que pertenece cada punto.

Para poder convertir los datos que provienen de imágenes DICOM en un formato valido que utiliza KMeans se utiliza la cabecera ***"dicom/dicomutils.h"*** junto al siguiente método:

getKMeansPoints(data): Utiliza una matriz y lo transforma a un vector basado en la clase

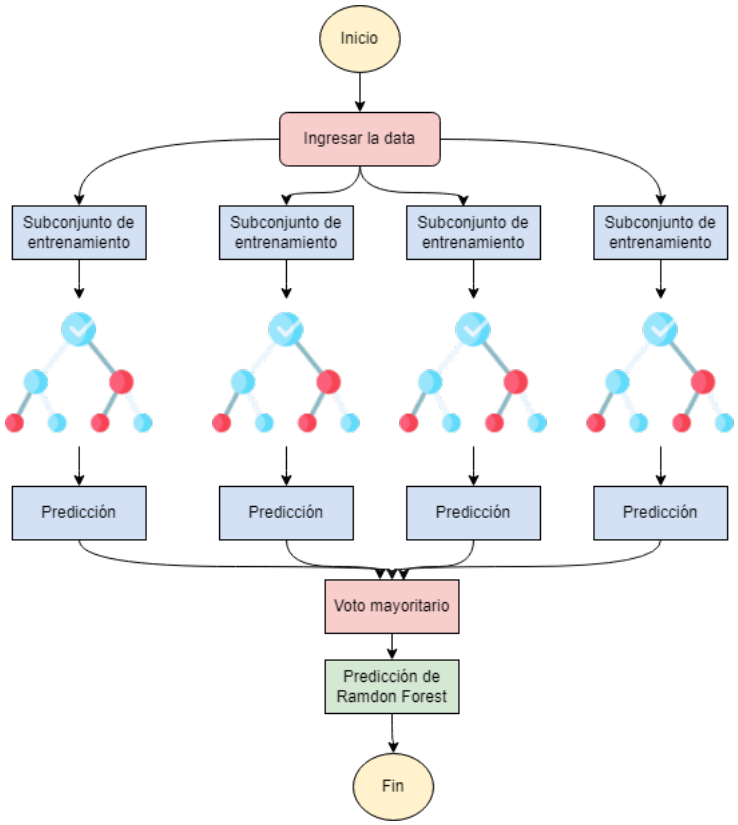
Point. El resultado que devuelve sirve como entrada para ejecutar el algoritmo de ***KMeans.*** Toma como entrada el parámetro ***data*** que es la matriz de datos que se desea transformar.

2.1.3.3 Random Forest (RF).

RF es una técnica de lenguaje supervisado que básicamente lo que realiza es crear varios árboles de decisiones sobre un conjunto de datos y de estos datos los divide en pequeñas partes. Con cada parte genera un nuevo árbol y con esto se va entrenando para al final combinar sus resultados, de tal forma se logra obtener una predicción general de cada subconjunto,todo esta estructura se puede apreciar en la Figura 13. Esta implementación originalmente se encuentra como aplicación de consola y también tiene su versión para el lenguaje de programación R tomando como base la aplicación de consola. También la

Figura 13

Diagrama Ramdon Forest.



Nota: *Diagrama Ramdon Forest. Elaborado por: Los autores.*

ejecución de esta versión de RF hace uso de los threads nativos de C++, lo que permite utilizar varios hilos durante la ejecución y procesamiento de los datos. Hay que tener en cuenta que para poder utilizar esta característica se necesita utilizar el estándar 11 de C++, ya que desde esa versión se implementan los threads de forma nativa. Para adaptar la librería DicomClassifier se implementa las clases principales y crear un adaptador de dichas clases para ejecutar la técnica por medio de funciones sencillas. Esta versión toma como entrada un archivo csv donde se definen todos los puntos a evaluar junto a sus respectivas etiquetas.

Se creó la clase RF que nos sirve como puente que conecta la ejecución del algoritmo de Random Forest y también nos ayuda para inicializar los valores de los parámetros que va a utiliza. Esta clase requiere como entrada un valor entero (***trees***) que representa la cantidad de árboles para el proceso de entrenamiento. En la Algoritmo 4 se muestra el funcionamiento del RF.

Algoritmo 4 Random Forest

Input: Matriz de imagen;
1: **for** $i = 1$ hasta c **do**
2: datos de entrenamiento de muestra aleatoria D con reemplazo para producir D_i
3: crear un nodo raíz N_i que contenga D_i
4: Llamar BuildTree(N_i)
5: **end for**
6: BuiltTree(N)
7: **if** N contiene instancias de una unica clase **then**
8: Return;
9: **else**
10: Seleccionar al azar el porcentaje de las posibles funciones de división en N Seleccione la característica F con la mayor ganancia de información para dividir en crear f nodos secundarios de N,N_i, ... , N_f donde F tiene f valores posibles(F_1, ... , F_f)
11: **for** $i = 1$ hasta f **do**
12: Establecer el contenido de N_i, a D_i, son todas las instancias en N que coinciden F_i
13: **end for**
14: **end if**

Para utilizar esta clase se necesita importar ***"rf/rf.h"*** en un nuevo proyecto. También tiene los siguientes métodos que ayudar a interactuar con la ejecución del algoritmo.

setFile(filename): Establece el archivo csv que contiene los datos de entrenamiento o evaluación para el algoritmo. Cada columna debe tener una cabecera y debe estar delimitado por espacios. El parámetro ***filename*** sirve para la ubicación del archivo de entrenamiento.

setPredictFile(filename): Establece la ubicación del archivo binario que representa el modelo entrenado. El parámetro ***filename*** sirve para la ubicación del archivo de entrenamiento.

setDepVarName(name): Establece el nombre de la columna que contiene los valores que representan a las etiquetas de cada punto en el conjunto de datos. El parámetro ***name*** establece el nombre de la columna de las etiquetas.

init(showOutput): Inicializa el algoritmo con los valores establecidos. El parámetro ***showOutput*** Determina el valor true o false, que representa si se muestran los resultados por consola.

Dentro de la clase RF se tiene una propiedad llamada forest que representa a la clase Forest y contiene los métodos para ejecutar el algoritmo. Esta propiedad cuenta con las siguientes funciones.

run(verbose, compute_oob_error): Ejecuta el algoritmo, utiliza el mismo método tanto para entrenar o para predecir utilizando como base si antes se cargó el archivo binario del modelo entrenado o no.

- ***verbose*** Valor true o false que representa el mostrar información adicional en la consola.
- ***compute_oob_error*** Valor true o false que representa el cálculo del error oob.

writeOutput(): Guarda el resumen del entrenamiento y ejecución del algoritmo en un archivo.

WriteConfusionFile(): Guarda en un archivo la matriz de confusión resultado del entrenamiento.

WritePredictionFile(): Guarda en un archivo las etiquetas que son resultado de la predicción.

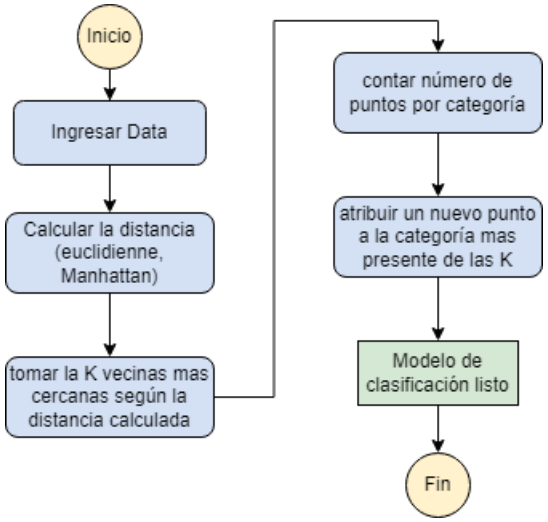
SaveToFile(): Guarda el modelo entrenado en un archivo binario.

Para generar los archivos CSV que contienen los conjuntos de datos junto a sus respectivas etiquetas se puede ejecutar el método ***saveData()*** que se encuentra dentro de ***"dicom/dicomutils.h"*** . Hay que usar el espacio como delimitador y guardarlo con cabecera, caso contrario no se leerá correctamente los datos.

2.1.3.4 k-Nearest Neighbor (KNN). El algoritmo KNN es uno de tipo supervisado el cual consta de encontrar los vecinos más cercanos en un punto y tratar de calcular su valor, es importante destacar que este algoritmo es muy sensible al valor que utilizamos como k ya que los resultados obtenidos varían mucho en base a este valor. Esta técnica sigue los pasos descritos en la Figura 14.

Figura 14

Diagrama KNN.



Nota: *Diagrama de k-Nearest Neighbor. Elaborado por: Los autores.*

La implementación original del algoritmo de KNN consta de una aplicación de consola, la utiliza un dataset para el entrenamiento y otro dataset para la fase de pruebas, también usa un valor de k (cantidad de vecinos cercanos). La aplicación de consola nos devuelve el porcentaje de éxito que tiene el algoritmo, así como el porcentaje de fracaso. De manera general esta

implementación realiza el procedimiento que se puede observar en el Algoritmo 5.

Algoritmo 5 Algoritmo K-Nearest Neighbor

Input: X= Matriz de imagen, K numero de vecinos, t tupla de entrada para clasificar;
Output: C= clase a la que es asignado t

```
1: N = 0
2: Encontrar conjunto de vecinos N para t
3: for d E D do
4:     if |N| ≤ K then
5:         N = N ∪ d
6:     else
7:         if ∋ u ∈ N tal que sim(t,u) ≥ sim(t,d) then
8:             N = N − u
9:             N = N ∪ d
10:        end if
11:    end if
12: end for
13: c= clase a la que mas u ∈ N es clasificada
```

Para separar lógica de la ejecución y resultados del contenedor que permite ejecutar la aplicación de consola, se crearon varias clases que nos permiten ordenar los datos. La clase ***DatasetPointer*** hace referencia al dataset que se utiliza durante el proceso de entrenamiento y pruebas del algoritmo. La clase ***SingleExecutionResults*** abstrae los resultados que tiene un conjunto de datos dentro del dataset. ***KNNResults*** nos da una interfaz para poder acceder a los resultados que nos da la ejecución del algoritmo. La clase ***KNN*** nos ayuda para poder inicializar y ejecutar el algoritmo y nos devuelve los resultados utilizando la clase ***KNNResults*** . Para usar las clases mencionadas se necesita de la cabecera ***"knn/knn.h"*** que contiene las clases para la ejecución de la técnica y también la cabecera ***"knn/dataset.h"*** que contiene la estructura de datos con que utiliza ***KNN*** . Para poder crear los distintos dataset puede utilizar la utilidad ***"knn/knnUtils.h"*** la cual tiene la siguiente función:

makeDataset(rows, cols, nLabels, sampleData, labels): Nos ayuda a crear un objeto DatasetPointer que sirve de entrada para el algoritmo de KNN. Los parámetros que emplea son los siguientes:

- ***rows*** Valor entero que representa la cantidad de puntos en el dataset.
- ***cols*** Valor entero que representa el total de características o dimensiones que tiene cada punto en el dataset.
- ***nLabels*** Valor entero que representa el total de etiquetas que se distribuyen en el dataset.
- ***sampleData*** Puntero con los valores de los puntos con sus respectivas dimensiones que se van a evaluar.
- ***labels*** Puntero con los valores de las etiquetas que pertenecen a cada punto.

Para inicializar el algoritmo con los valores del dataset debe utilizar la clase KNN que se encuentra en el fichero ***"knn/knn.h".*** Esta clase toma como entrada un objeto del tipo ***DatasetPointer*** que representa el dataset entrenamiento para la técnica. Esta clase solo tiene un método que nos ayuda a ejecutar el algoritmo.

run(k, target) Empieza a entrenar el algoritmo con los parámetros dados. Devuelve un objeto de la clase ***KNNResults*** con los resultados.

- ***k*** Valor entero que representa la cantidad de vecinos cercanos.
- ***target*** Objeto de tipo DatasetPointer que representa el conjunto de datos de prueba. Valores con los que se evalúa el algoritmo.

Al terminar de ejecutar, el algoritmo retorna un objeto del tipo ***KNNResults*** que contiene los resultados. Este objeto tiene los siguientes métodos:

top1Result() Devuelve un objeto de la clase SingleExecutionResults con los resultados del porcentaje de éxito y fracaso. Valores que describen la precisión para el dataset.

getPredictions() Devuelve el resultado de las predicciones para cada punto en el dataset de prueba.

getRawResults() Devuelve los resultados en formato ***DatasetPointer*** .

Por último, para poder acceder a los porcentajes de éxito y fracaso tenemos la clase ***SingleExecutionResults*** con los siguientes métodos:

successRate() Devuelve el porcentaje de éxito para el dataset.

rejectionRate() Nos devuelve el porcentaje de fracaso para el dataset.

En caso de necesitar convertir datos en formato vector a un puntero se utilizan los métodos ***parseKNNData*** y ***parseKNNLabels*** que se encuentran en el fichero ***"dicom/dicomutils.h".***

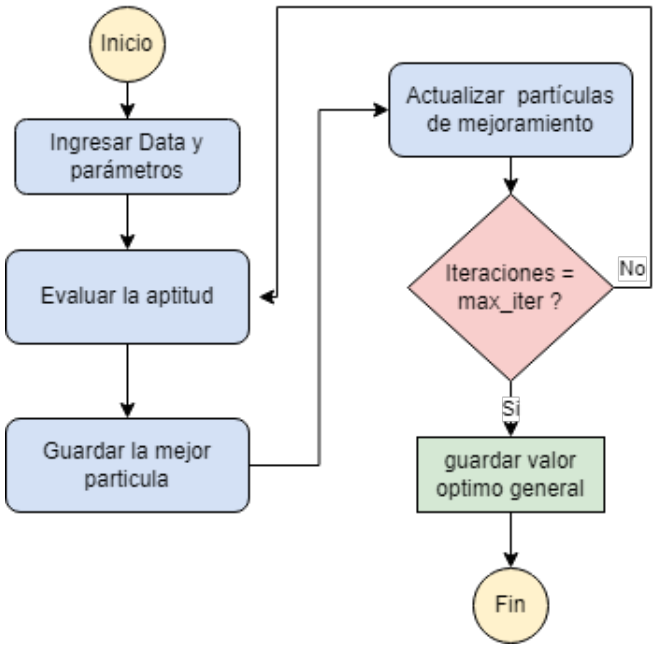
2.1.3.5 Convolutional Neural Network (CNN). El algoritmo de CNN está enfocado en el tratamiento de imágenes actualmente también se lo utiliza para el procesamiento de texto, pero su fuerte son las imágenes, en la librería DicomClassifier recibe como data inicial la matriz de la imagen DICOM, luego este pasara a calcular la mejor aptitud de la imagen fraccionando la información de entrada en fragmentos más pequeños para de esta manera obtener un dato de predicción como se puede visualizar en la Figura 15.

Para la implementación de CNN o redes neuronales convolucionales se utilizó en el framework de tiny_dnn. Este es un framework para redes neuronales diseñado en C++. El framework tiny_dnn tiene ya establecido varias funciones que nos ayudan a crear distintos tipos de redes neuronales. En este caso la implementación para la librería DicomClassifier fue más sencilla ya que al ser un framework, la interfaz que permite comunicar al directamente con la creación y ejecución de una red neuronal ya está lista. En este punto se implementan los ficheros y se realizan las configuraciones necesarias para que funcione en conjunto con la librería DicomClassifier. Para ejecutar esta técnica hay que asegurarse de que el compilador soporte la versión 14 del estándar de C++ y utilizarla en el proyecto. De manera general en la Figura 16 se puede observar cómo funciona CNN que se implementó en la versión de la librería DicomClassifier.

Para acceder a las definiciones de tipos, clases y funciones que nos ayudan a construir una red neuronal hay que importar la cabecera ***"net/nn.h"*** . También hay que tener en cuenta que

Figura 15

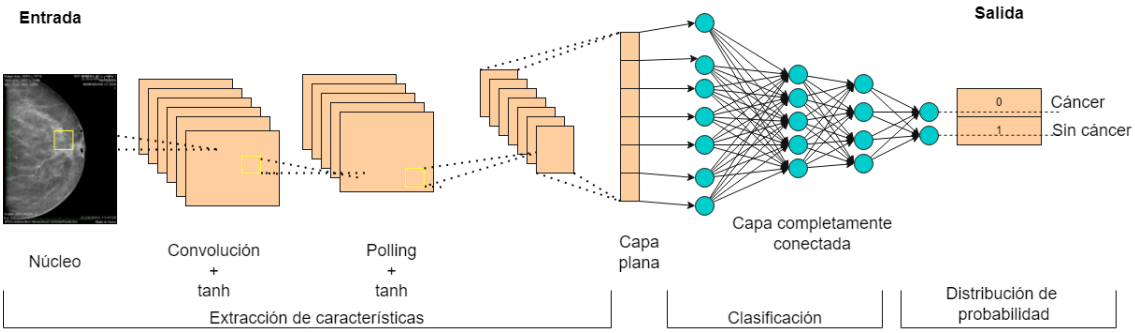
Diagrama CNN.



Nota: *Diagrama de Convolutional Neural Network. Elaborado por: Los autores.*

Figura 16

Estructura CNN.



Nota: *Funcionamiento de la red neuronal convolucional implementada en la librería DicomClassifier. Elaborado por: Los autores.*

esta implementación cuenta con el uso de ***namespaces*** que agrupa un conjunto de identificadores según su uso.

Con el archivo importado hacer uso del ***namespaces nn*** , esto trae los tipos básicos para

construir una red neuronal convolucional. Consta de los siguientes identificadores:

vec_t Es un tipo que representa un punto junto a sus dimensiones que serán de entrada para la red neuronal.

label_t Es un tipo que representa un vector de enteros que son las etiquetas o clases que pertenecen a un punto.

También tenemos la clase ***network*** que define el modelo de la red neuronal. Para usarlo se define una variable de la siguiente forma ***network<sequential>.*** El tipo ***sequential*** es el tipo de modelo de red neuronal que se va a crear. Esta clase tiene los siguientes métodos:

train<loss_function>(optimizer, input, labels, batch_size, epochs): Realiza el entrenamiento del modelo utilizando los parámetros dados.

- ***loss_function*** Define la función de perdida que va a utilizar el modelo en su entrenamiento. Las funciones de perdida que se pueden usar son: i) mse ii) cross_entropy iii) cross_entropy_multiclass
- ***optimizer*** Algoritmo de optimización que se va a utilizar.
- ***input*** Vector del tipo vec_t. Representa el conjunto de datos que se utiliza en el entrenamiento.
- ***labels*** Vector del tipo label_t. Representa las etiquetas que pertenecen a cada punto en el conjunto de datos.
- ***batch_size*** Número de muestras por actualización de parámetros.
- ***epochs*** Número de épocas en la fase de entrenamiento.

predict_label(in): Realiza la predicción de la clase para un punto en específico. El parámetro ***in*** establece el tipo de vector y el parámetro ***vec_t.*** representa a un punto junto a sus dimensiones.

save(filename, content_type, format): Guarda el modelo en un archivo según los parámetros dados.

- ***filename*** Nombre del archivo con el que se guarda el modelo.
- ***content_type*** Tipo de contenido que tiene el archivo. Puede ser: weights, model, weights_ and_model.
- ***format*** Formato del archivo que se va a guardar. Puede ser file_format::json o file_ format::binary. Por defecto usa file_ format::binary.

load(filename, content_type, format): Carga el modelo desde el archivo especificado y utiliza los parámetros.

- ***filename*** Nombre del archivo con el que se guarda el modelo.
- ***content_type*** Tipo de contenido que tiene el archivo.Puede ser: weights, model, weights_and_model. Por defecto usa weights_and_model.
- ***format*** Formato que tiene el archivo a cargar.Puede ser file_format::json o file_format::binary. Por defecto usa file_format::binary.

Para agregar capas a la red neuronal se hace referencia a namespace nn::layers o ***namespace*** puede hacer uso de la capa con nn:layers::<capa>, donde capa es la capa de la red que se va a utilizar. Para agregar capas a la red hay que hacer uso del operador $<<$.

Las capas que podemos utilizar son las siguientes.

convolutional_layer(out_width, in_height, window_size, in_channels, out_channels): Toma una imagen en dos dimensiones y realiza la operación de filtrado.

- ***in_width*** ancho del tamaño de la imagen.
- ***in_height*** alto del tamaño de la imagen.

- ***window_size*** tamaño del kernel de convolución.
- ***in_channels*** número de canales de la imagen. Escala de grises = 1 y RGB = 3.
- ***out_channels*** canales de imagen de salida.

quantized_convolutional_layer(in_ width, in_height, window_ size, in_channels, out_channels): Toma una imagen en dos dimensiones y realiza la operación de filtrado.

- ***in_width*** ancho del tamaño de la imagen.
- ***in_height*** alto del tamaño de la imagen.
- ***window_size*** tamaño del kernel de convolución.
- ***in_channels*** número de canales de la imagen. Escala de grises = 1 y RGB = 3.
- ***out_channels*** canales de imagen de salida.

max_pooling_layer(in_width, in_height, in_channels, pooling_size): Aplica la operación max-pooling a los datos de entrada. Se puede usar el alias max_pool.

- ***in_width*** ancho del tamaño de la imagen.
- ***in_height*** alto del tamaño de la imagen.
- ***in_channels*** número de canaleso profundidad de la imagen de entrada.
- ***pooling_size*** factor de la reducción de la escala.

average_pooling_layer(in_width, in_height, in_channels, pooling_size): Aplica la operación de average-pooling sobre el conjunto de datos. Se puede usar el alias ave_pool.

- ***in_width*** ancho del tamaño de la imagen.
- ***in_height*** alto del tamaño de la imagen.
- ***in_channels*** número de canaleso profundidad de la imagen de entrada.

- ***pooling_size*** factor de la reducción de la escala.

fully_connected_layer(in_dim, out_dim): Representa la capa completamente conectada de la red neuronal.

- ***in_dim*** número de elementos de entrada.
- ***out_dim*** número de elementos de salida.

Las funciones activación se puede usar importando el ***namespace nn::activation*** . De igual manera para agregar al modelo de la red se necesita utilizar el operador justo después de alguna capa. Para utilizar las funciones de activación utilice ***activation::<function>*** , donde function es el nombre de la función de activación o su respectivo alias.

Las funciones de activación disponibles son:

- ***sigmoid_layer():*** Ejecuta la función sigmoide para los datos de la capa. El alias es ***sigmoid.***
- ***tanh_layer():*** Ejecuta la función tangente para los datos de la capa. El alias es ***tanh.***
- ***relu_later():*** Ejecuta la función ReLu para los datos de la capa. El alias es ***relu.***
- ***softmax_layer():*** Ejecuta la función softmax para los datos de capa. El alias es ***softmax.***

Por último, las funciones de optimización se representan con structs. Para utilizarlas se declara y se hace referencia al sctruct del respectivo optimizador. Los optimizadores disponibles son: i) agrad. ii) RMSprop. iii) adam. iv) adamax. v) gradient_descent. vi) momentum. vii) nesterov_momentum.

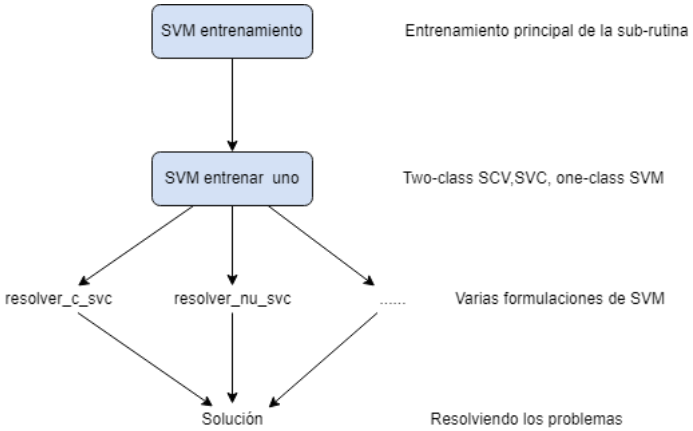
2.1.3.6 SVM (Support Vector Machines).

La implementación de la técnica de SVM se basó en el trabajo de LIBSVM (Chang & Lin, 2011), que es un programa de consola y a

su vez librería que resuelve problemas de clasificación y regresión utilizando el algoritmo de SVM. Soporta distintas formulaciones del algoritmo de SVM, tales como *nu-SVC*y *C-SVC* para clasificación, *épsilon-SVR* y *nu-SVR* para regresión. También tiene soporte para clasificación de múltiples clases o etiquetas. La base de esta aplicación esta desarrollada en el lenguaje C y tiene extensiones para otros lenguajes como Java, Matlab, Python, etc. El funcionamiento general de LIBSVM se muestra en la Figura 17.

Figura 17

Support Vector Machines.



Nota: *Diagrama Support Vector Machines . Fuente: (Chang & Lin, 2011)*

Para agregar esta implementación de *SVM* al código base de DicomClassifier, se utilizaron las funcionalidades principales de LIBSVM y se los añadió a un módulo. Esto permite acceder a las funciones que ayudan al entrenamiento del modelo y así realizar predicciones sobre las clases. Para poder acceder a los métodos se necesita importar el archivo ***"svm/svm.h"*** de la librería DicomClassifier. Esta cabecera brinda el acceso a las siguientes estructuras:

svm_node: Representa a un nodo o dimensión que pertenece a un punto dentro del conjunto de datos. Un vector de svm_node representa a un punto y sus dimensiones en el conjunto de datos. Tiene los atributos:

- ***index*** Posición, columna o dimensión a la que pertenece el valor del nodo. El índice -1 indica el final del vector.
- ***value*** Valor del nodo.

svm_problem: Describe el problema a resolver, en otras palabras, contiene los datos con los que se va a entrenar el modelo. Tiene los atributos:

- ***l*** Es la cantidad de puntos que hay en el conjunto de datos.
- ***y*** Es un array que contiene los valores enteros que representan las etiquetas o clases de cada punto.
- ***x*** Es un array de punteros donde cada puntero representa a un array de ***svm_node*** , en otras palabras, un vector de las dimensiones para los distintos puntos en el conjunto de datos.

svm_parameter: Esta estructura representa los parámetros que se utiliza modelo SVM. Cuenta con los atributos:

- ***svm_type*** Valor entero que representa el tipo de SVM que va a evaluar. Para ello se utiliza un enumerador con los siguientes valores: i) C_SVC ii) NU_SVC iii) ONE_CLASS.
- ***kernel_type*** Representa el tipo de kernel que se va a utilizar. Se puede usar un enumerador con los valores: i) LINEAR ii) POLY iii) RBF iv) SIGMOID v) PRECOMPUTED.
- ***gamma*** Establece el valor degree para la función del kernel. Se utiliza para el kernel POLY.
- ***coef0*** Establece el valor de coef0 para cuando se usa el kernel POLY o SIGMOID.
- ***cache_size*** Es el tamaño del cache del kernel, se expresa en megabytes.

- ***eps*** Es el criterio de parada con el cual converge el algoritmo. Usualmente se usa 0.00001 en nu-SVC y 0.001 para los demás.
- ***C*** Es el costo de violación en las restricciones. Se utiliza para C_SVC.
- ***nr_weight*** Valor entero que indica la cantidad de elementos en los vectores weight_label y weight. El valor 0 indica que no haya penalización para ninguna clase.
- ***weight_label*** Representa un vector con las etiquetas o clases que se utilizan para penalizar a una clase.
- ***weight*** Representa un vector con el peso de penalización para cada clase en el vector weight_label.
- ***nu*** Representa el valor del coeficiente ν para NU_SVC y ONE_CLASS.
- ***probability*** Representa si se quiere obtener la estimación de probabilidad. El valor 1 indica que si y el valor de 0 que no.

svm_model: Representa al modelo de SVM después de la fase de entrenamiento. Tiene las propiedades:

- ***svm_parameter*** Estructura del tipo svm_parameter con los parámetros para entrenar el modelo.
- ***nr_class*** Representa al número de clases.
- ***l*** Representa al número de vectores de soporte.
- ***SV*** Representa al vector de soporte
- ***sv_coeff*** Representa a los coeficientes para cada SV en la función de decisión.
- ***rho*** Constantes en la función de decisión.
- ***label*** array que representa los valores de las clases o etiquetas.

- ***nSV*** Representa a la cantidad de vectores de soporte para cada clase.
- ***free_sv*** Representa como el modelo fue creado. El valor de 1 si fue cargado utilizando un archivo binario o el valor de 0 si el modelo fue creado utilizando la función svm_train.

Aparte de las estructuras que nos permiten definir el modelo junto a la configuración de sus parámetros y tipos de datos a utilizar, también se define las funciones principales ayudan al entrenamiento del modelo o a la predicción de las clases. Las funciones disponibles son:

svm_train(prob, param): Construye el modelo SVM de acuerdo a los datos y parámetros dados. Devuelve una estructura svm_model que define el resultado del entrenamiento.

- ***prob*** Estructura del tipo ***svm_problem*** con los datos que se utiliza para el entrenamiento.
- ***param*** Estructura del tipo ***svmparam_param*** con la configuración de parámetros que utiliza el modelo en el entrenamiento.

svm_save_model(model_filename, model): Guarda el modelo generado en un archivo. Devuelve 0 si se guardo correctamente y -1 si hubo algun error.

- ***model_filename*** Nombre del archivo que se va a guardar.
- ***model*** Estructura svm_model que representa al modelo entrenado.

svm_load_model(model_filename): Carga el modelo SVM desde el archivo establecido. El parámetro ***model_filename*** es nombre del archivo con el modelo a cargar.

svm_predict(model, x): Realiza la predicción o clasificación sobre un punto dado. Para el modelo ONE_CLASS devuelve 1 o -1 como valor de la etiqueta. Tiene los parámetros:

- ***model*** Estructura de tipo ***svm_model*** que representa el modelo SVM.
- ***x*** Representación del vector de dimensiones de un punto, es un array de ***svm_node.***

La utilidad “svm/svm_util.h” contiene funciones para transformar vectores de datos a estructuras que son utilizadas por ***SVM*** . El funcionamiento de los métodos se detallan a continuación.

createSVMNode(points): Crea una estructura del tipo ***svm_node*** que representa a un punto junto con los valores de sus dimensiones, en otras palabras, regresa un array del tipo ***svm_node.*** El parámetro ***points*** es un vector de valores enteros que representa a un punto junto a sus dimensiones.

initProblemNodes(x_space, prob, data): Inicializa el conjunto de datos con los que se va a entrenar el modelo. Los párametros que utiliza son:

- ***x_space*** Es un array del tipo svm_node. Es la representación de los puntos que tiene el dataset.
- ***prob*** Estructura del tipo svm_problem. Para inicializar los valores x del problema a solucionar.
- ***data*** Datos de entrada que se van a convertir. Dataset con los valores de las distintas matrices de imagen DICOM.

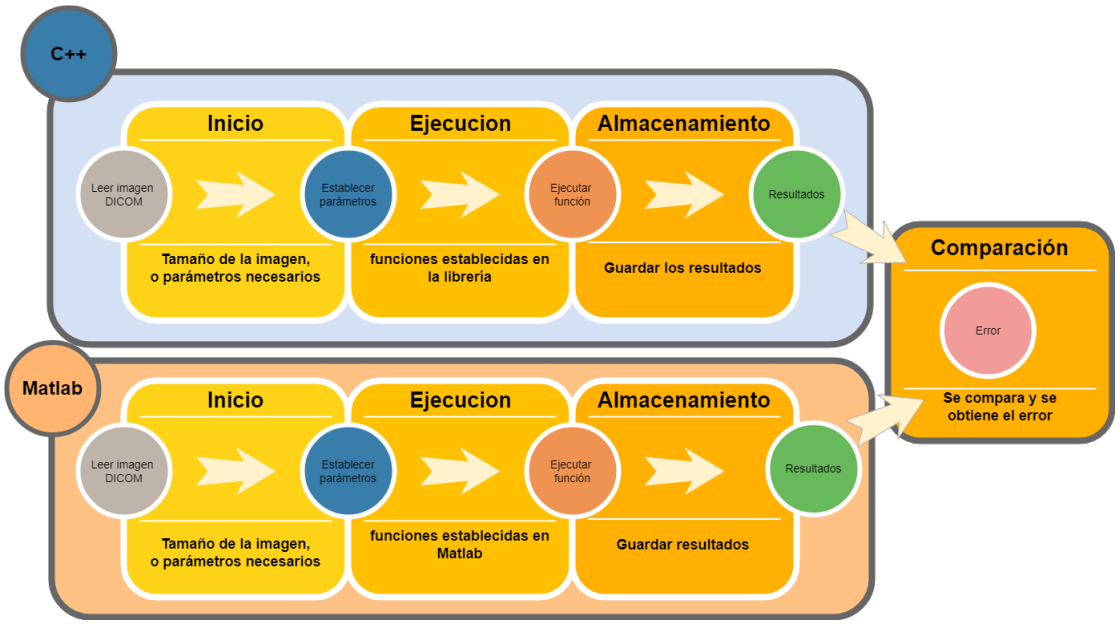
2.1.4 Fase 4: Comprobación con Matlab

Para esta parte del proyecto utilizamos un software fundamental como es Matlab ya que es un software que trabaja con cálculos numéricos y que por defecto viene incorporado gran parte de algoritmos de clasificación. También utilizamos la función ***dicomread*** incorporada en Matlab, esta función nos permite leer datos de una imagen de archivos en formato DICOM la cual facilita la lectura y carga de información fundamental de la imagen a Matlab y poder enviarla como parámetros a los algoritmos en Matlab. El proceso detallado anteriormente es muy similar al desarrollo de la librería DicomClassifier, donde de igual forma leemos la imagen tipo

DICOM, establecemos los parámetros para ejecutar cada uno de los algoritmos, y obtenemos resultados este proceso se lo puede apreciar en la Figura 18 y la librería, donde los resultados se muestran en la siguiente sección.

Figura 18

Proceso de comparación.



Nota: *Comprobación de Matlab y la librería. Elaborado por: Los autores.*

Para la comprobación se realizó la implementación y ejecución sobre los mismos datos tanto en C++ como en Matlab, es importante recalcar que debió al tiempo para el desarrollo de la librería se comprobaron los resultados en Matlab con diez imágenes DICOM proporcionados en el dataset.

Para la obtención de los resultados se procedió a calcular el error absoluto mediante la Ecuación 29 la cual consta de la comparación de los resultados obtenidos en cada imagen aplicando la formula.

$$\varepsilon_{absoluto} = |V_{real} - V_{medios}| \tag{29}$$

De esta forma logramos verificar los resultados que son muy similares en Matlab con un

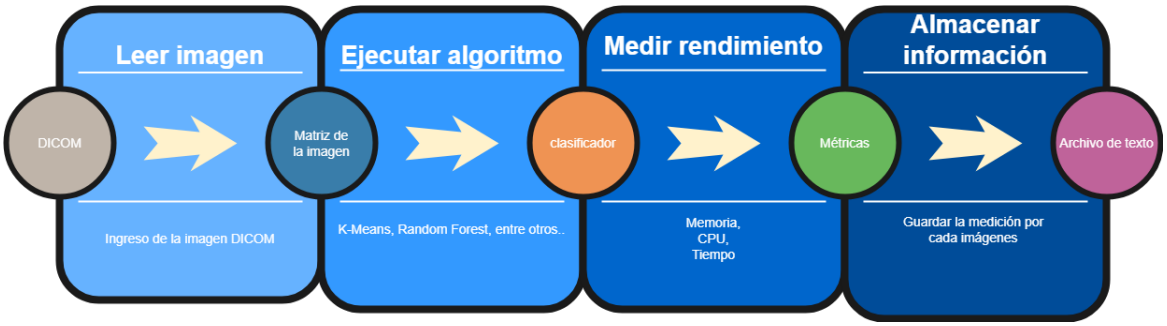
mínimo error en ciertos casos esta información se puede apreciar mejor en el próximo capítulo de resultados.

2.1.4.1 Pruebas de rendimiento y métricas.

Para las pruebas de rendimiento gracias a que Linux nos ayuda con el acceso a esta información, incorporamos en la ejecución de los algoritmos una (función o código) para tomar los valores correspondientes como se puede observar en la Figura 19. Es importante resaltar que los valores que se obtienen en el rendimiento y métricas varía dependiendo los equipos utilizados para la medición. Para medir el rendimiento utilizamos parámetros de CPU para observar el porcentaje de uso durante el proceso de la imagen, el uso de memoria RAM que es donde se almacena de manera temporal los datos que se procesan en la ejecución de cada algoritmo y por último el tiempo de ejecución promedio que ocupan en procesar cada imagen médica en formato DICOM.

Figura 19

Métricas y rendimiento de la librería.



Nota: *Métricas y rendimiento de la librería. Elaborado por: Los autores.*

RESULTADOS

3.1 HERRAMIENTAS

Para el desarrollo de la librería DicomClassifier se utilizó un data set con un total de 150 imágenes mamográficas en formato DICOM, también posee metadatos relacionados con los pacientes, se encuentra dividida en 76 imágenes con vista craneocaudal (CC) y 74 imágenes de Medio Lateral Oblicuo (MLO) en los cuales se puede apreciar la presencia de lesiones como masas, calcificaciones, microcalcificaciones (MCs) y mamas normales como se puede ver en la Tabla 1.

Tabla 1

Detalle del dataset

Detalle		Vista	N° Img
Masas	CC	L	5
		R	5
	MLO	L	2
		R	6
MCs	CC	L	15
		R	21
	MLO	L	22
		R	23
Masas y MCs	CC	L	9
		R	12
	MLO	L	8
		R	9
Normal	CC	L	4
		R	5
	MLO	L	3
		R	1

Nota: *Descripción detallada de las imágenes del dataset. Elaborado por: Los autores.*

3.2 HARDAWARE

Los equipos de cómputo que se utilizaron los detallados en la Tabla 2. Es importante mencionar que para la parte de entrenamiento se utilizó la Máquina 1 y para la parte de testing se utilizó la Máquina 2

Tabla 2

Hardware utilizado

Máquina	Procesador	Núcleos	Frecuencia	Memoria
1	i7-6820	4	2.7 GHz	16 GB
2	Intel(R) Xeon(R) CPU E5-2683 v4	1	2.10 GHz	12 GB

Nota: *Equipos utilizados junto al detalle de cada uno. Elaborado por: Los autores.*

3.3 SOFTWARE

La librería DicomClassifier utiliza como dependencia libdcmtk junto a CMake y g++ para el proceso de compilación. Además, se utilizaron los entornos de desarrollo QT y Code::Blocks. La descripción de las versiones utilizadas se detalla en la Tabla 3.

Tabla 3

Software utilizado

	Nombre	Descripción	Versión
Dependencias	libdcmtk-dev	librería para el manejo de imágenes DICOM.	3.6.1
	g++	Compilador para C++.	9.3
	CMake	Herramienta para automatizar el proceso de compilación.	3.21
Entornos de desarrollo	Code::Blocks	IDE para C/C++.	20.03
	QT	IDE para C/C++.	6.2.2

Nota: *Dependencias y entornos de desarrollo que se utilizaron junto a sus respectivas versiones. Elaborado por: Los autores.*

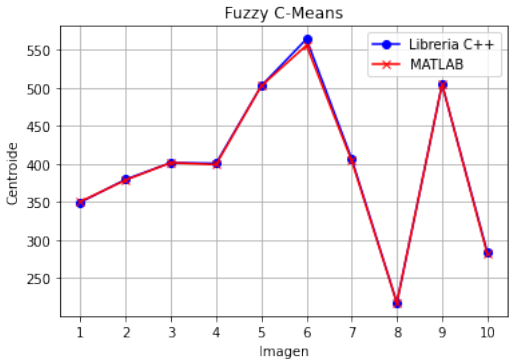
3.4 COMPARACIÓN CON MATLAB

Para verificar que la implementación de los algoritmos de clasificación sea correcta se realizó una comparación con Matlab. En esta fase se utilizó la Máquina 2 que se detalla en la Tabla 2. Para ello se tomó una muestra de 10 imágenes y se realizó un modelo de clasificación con la misma configuración para cada imagen tanto en Matlab como en la librería DicomClassifier. A continuación, se seleccionó un parámetro a evaluar del modelo en el que nos apoyamos para obtener el error en comparación con la implementación en Matlab.

El modelo de Fuzzy C-Means tomo como base 3 clusters para cada una de las imágenes y el parámetro a evaluar fueron los centroides, dándonos como resultado un promedio de 1.6994 % de error. El detalle error para cada imagen se puede observar en la Figura 20.

Figura 20

Comparación Fuzzy C-Means



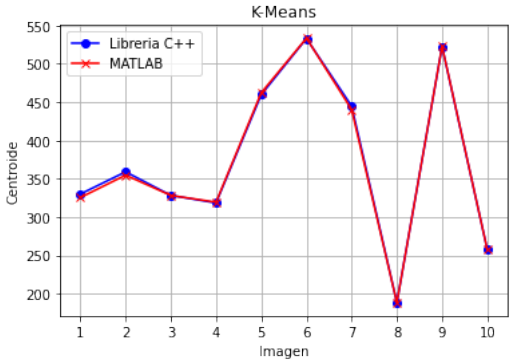
Nota: *Comparación del error para Fuzzy C-Means. Elaborado por: Los autores.*

Para KMeans se realizó un modelo de 3 clusters, al igual que Fuzzy C-Means se evaluaron los centroides para el cálculo del error. La Figura 21 muestra el comportamiento de los centroides para las 10 imágenes evaluadas. El promedio de error obtenido fue de 1.7968 % en comparación a los resultados de Matlab.

En la técnica de Random Forest se tomó como base el valor obtenido del error OOB (out-

Figura 21

Comparación K-Means

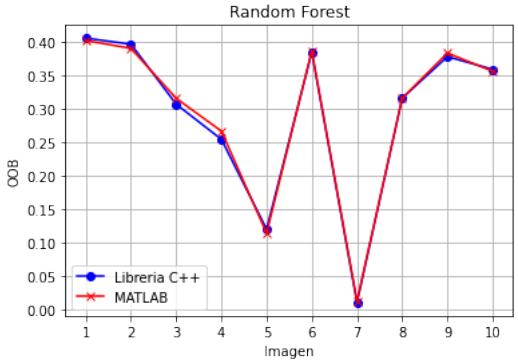


Nota: *Comparación del error para K-Means. Elaborado por: Los autores.*

of-bag), comparándolo con Matlab. La Figura 22 nos indica que la tendencia del error tanto en matlab como en la librería DicomClassifier es muy similar. El promedio de error es del 0.0049 % lo que nos indica resultados muy cercanos a los valores de Matlab.

Figura 22

Comparación Random Forest



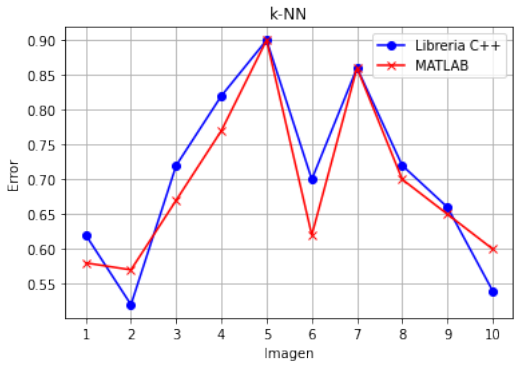
Nota: *Comparación del error para Random Forest. Elaborado por: Los autores.*

En el modelo realizado para KNN se utilizó con un valor de k (número de vecinos cercanos) igual a dos y el parámetro que se evaluó fue porcentaje de predicción para cada clase. En la Figura 23 podemos observar que la tendencia del error por parte de la librería DicomClassifier es

muy cercana a la presentada por Matlab. El valor del error promedio obtenido fue de 0.0316 %.

Figura 23

Comparación KNN

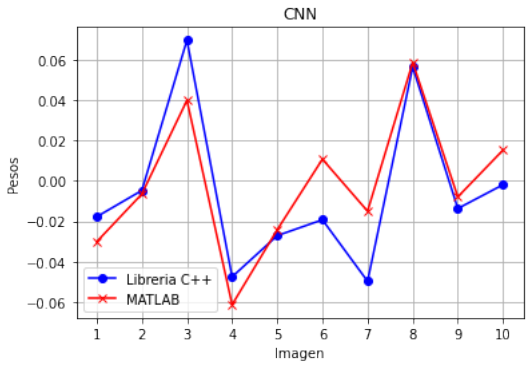


Nota: *Comparación del error para KNN.. Elaborado por: Los autores.*

Para el modelo de redes neuronales convolucionales (CNN) se tomó como parámetro a evaluar los pesos resultantes del modelo. Como se puede observar en la Figura 24 los pesos para ambos casos siguien una tendencia muy similar. Se compararon con los que dio Matlab y el error promedio es de 0.0578 %.

Figura 24

Comparación CNN



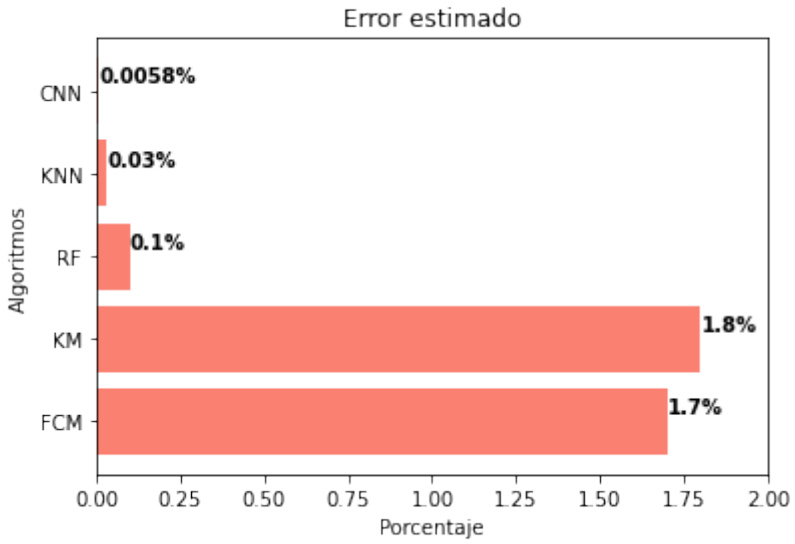
Nota: *Comparación del error para CNN. Elaborado por: Los autores.*

Como resultado se realiza una comparación entre los resultados obtenidos en Matlab contra

los resultados en C++ con 10 imágenes para estimar un error aproximado, es probable que este error cambie al utilizar otros datos para su comparación, en la Tabla 25 se muestran los valores.

Figura 25

Error de Matlab con librería DicomClassifier.



Nota: *Resumen del error en comparación a Matlab con librería DicomClassifier. Elaborado por: Los autores.*

3.5 PRUEBAS DE RENDIMIENTO

Para las pruebas de rendimiento se tomaron como base 100 imágenes del dataset y al igual que en la fase anterior se utilizó la maquina 2 mostrada en la Tabla 2. Las métricas evaluadas para poder medir el rendimiento que tiene la librería DicomClassifier se detalla a continuación:

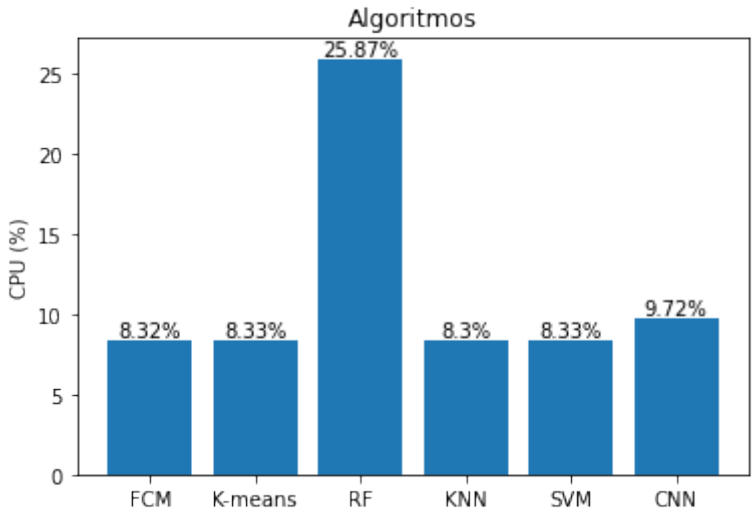
3.5.1 Consumo de CPU

El consumo promedio del CPU que se obtiene de la mayoría de las técnicas utilizadas sobre el conjunto de imágenes DICOM hace uso de un 11.48 % del total del CPU de manera general.

Aunque se puede observar que la técnica de RF hace un mayor consumo sobre KNN, CNN, KM, FCM, con un 25.87 % del CPU, es debido a que este RF se ejecuta bajo múltiples hilos por defecto, lo que hace que el consumo de recursos computacionales aumente. Por otro lado, CNN, KNN, KM, FCM hacen un uso promedio del CPU entre 8 % y 10 %, lo cual indica que está haciendo uso de un único hilo de trabajo como lo puede observar en la Figura 26.

Figura 26

Rendimiento del CPU



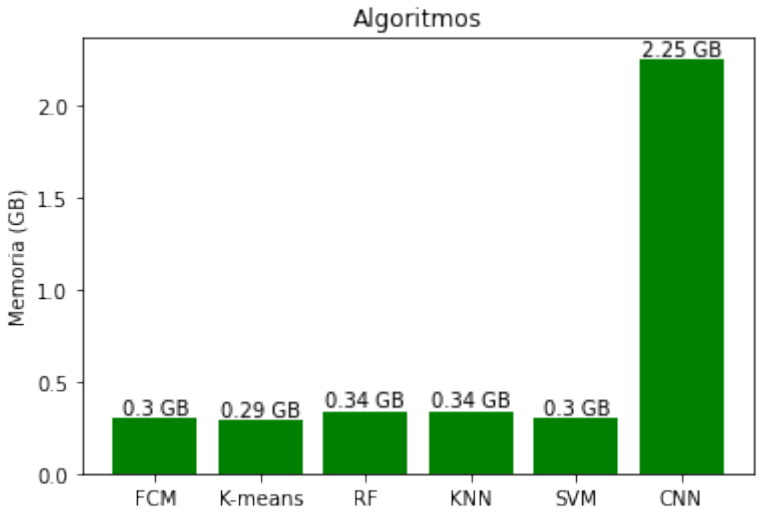
Nota: *Comparación del porcentaje de uso del CPU por cada algoritmo. Elaborado por: Los autores.*

3.5.2 Consumo de memoria RAM

El promedio de memoria RAM consumida por las diferentes técnicas de clasificación de la librería DicomClassifier es de 0.63 GB de manera general. Se puede observar en la Figura 27. El mayor consumo de memoria es por parte CNN con un promedio de 2.25 GB de RAM. Esto se debe que para entrenar utilizando este modelo se necesita reservar espacio no solo para la imagen DICOM, sino que también para las diferentes capas que conforman el modelo de la red convolucional.

Figura 27

Rendimiento de la memoria RAM



Nota: *Comparación de la memoria RAM utilizada por los algoritmos. Elaborado por: Los autores.*

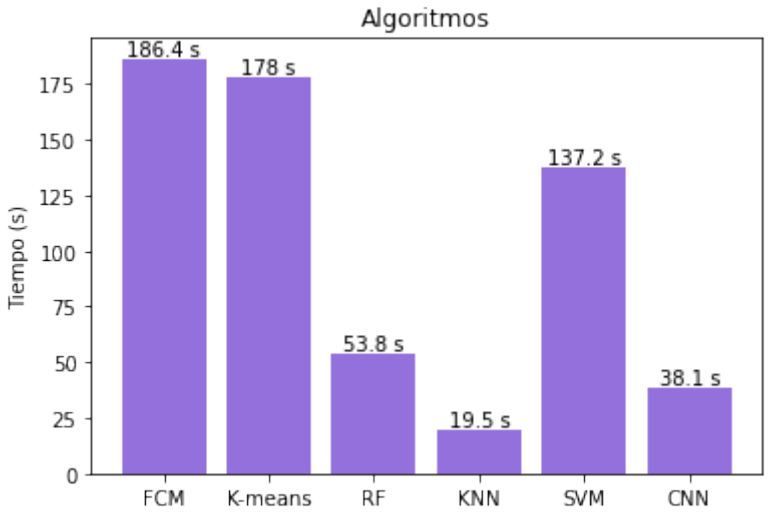
3.5.3 Tiempo de ejecución

Con la finalidad de evaluar y medir el tiempo de entrenamiento de la librería DicomClassifier se utilizaron computadoras con similares características como está representado en la Tabla 2 y así poder medir esta métrica. En la Figura 28 se puede observar que las técnicas que ocupan un mayor tiempo durante su ejecución son: FCM, KM y SVM. Por otro lado, la técnica de Random Forest consume un menor tiempo debido a que hace uso de múltiples hilos durante su ejecución. CNN tienen un tiempo de ejecución menor, lo cual se debe a temas de parametrización, ya que se está utilizando solo una epoch para entrenar al modelo, lo que resulta en un tiempo bajo de ejecución comparado a los demás algoritmos. Por último, KNN se realizó una optimización que consiste en escoger el mejor valor de k (vecinos cercanos), esto resulto en que el algoritmo converja de manera mucho más rápida que FCM, KM, RF, SVM, CNN.

Es importante destacar que los resultados de las pruebas de rendimiento dependen mucho

Figura 28

Tiempo de ejecución



Nota: *Comparación del tiempo utilizado por los algoritmos. Elaborado por: Los autores.*

de los equipos utilizados para los mismos por lo cual estos datos podrían variar, pero se estima en base al error de los algoritmos que los resultados sean similares a los mostrados anteriormente, también es importante considerar que el proceso de imágenes dependerá del detalle de las imágenes, masas, micros, masas con micros y normales por lo cual mostramos los resultados obtenidos en la Tabla 4.

Tabla 4

Resultados

Detalle			Fuzzy C-Means			K-Means			Random Forest			k-NN			SVM_oneclass			CNN		
			T (s)	RAM (Kb)	CPU (%)	T (s)	RAM (Kb)	CPU (%)	T (s)	RAM (Kb)	CPU (%)	T (s)	RAM (Kb)	CPU (%)	T (s)	RAM (Kb)	CPU (%)	T (s)	RAM (Kb)	CPU (%)
Masas	CC	L	161.29	0.27	8.33	116.17	0.79	8.33	55.75	0.29	30.66	16.06	0.28	8.33	102.65	0.23	8.3	32.52	1.95	9.73
		R	161.29	0.27	8.33	116.17	0.79	8.33	55.75	0.29	30.66	16.06	0.28	8.33	102.65	0.23	8.32	32.52	1.95	9.73
	MLO	L	114.12	0.27	8.32	137.58	1.81	8.33	40.34	0.30	23.67	15.99	0.28	8.33	105.76	0.25	8.33	32.38	1.96	9.71
		R	186.85	0.30	8.33	216.77	2.71	8.33	45.69	0.35	21	21.39	0.36	8.33	151.87	0.31	8.33	42.35	2.26	9.80
MCs	CC	L	201.60	0.29	8.33	173.91	2.96	8.33	58.11	0.33	28.33	19.45	0.33	8.33	134.80	0.29	8.33	37.98	2.12	9.74
		R	176.96	0.28	8.33	141.61	1.86	8.33	58.88	0.32	29.24	18.03	0.31	8.33	123.94	0.27	8.33	35.56	2.01	9.73
	MLO	L	182.97	0.29	8.33	158.49	1.93	8.33	28.42	0.33	20.69	19.01	0.33	8.33	130.94	0.29	8.33	37.37	2.12	9.73
		R	183.82	0.29	8.33	175.90	2.53	8.33	44.90	0.33	22.63	19.98	0.32	8.33	133.56	0.29	8.33	36.84	2.13	9.72
Masas y MCs	CC	L	236.57	0.30	8.33	236.74	2.33	8.33	72.33	0.37	28.99	22.86	0.38	8.33	169.18	0.32	8.33	43.09	2.38	9.71
		R	217.35	0.30	8.33	211.36	2.36	8.33	68.60	0.35	29.39	21.48	0.36	8.33	159.76	0.31	8.33	41.54	2.31	9.76
	MLO	L	194.00	0.30	8.33	227.31	2.33	8.33	55.91	0.37	24.14	22.34	0.38	8.33	165.61	0.32	8.33	44.21	2.38	9.67
		R	231.08	0.30	8.33	213.94	2.31	8.33	51.59	0.35	22.94	21.28	0.36	8.33	151.04	0.31	8.33	41.43	2.26	9.70
Normal	CC	L	128.98	0.27	8.33	123.38	1.53	8.33	46.27	0.30	26.98	16.14	0.28	8.33	93.04	0.25	8.32	31.34	1.84	9.74
		R	158.74	0.28	8.33	132.07	2.42	8.33	54.47	0.30	30.35	16.37	0.28	8.33	110.21	0.26	8.33	33.13	1.93	9.77
	MLO	L	145.45	0.30	8.33	217.17	1.54	8.33	38.64	0.35	19.96	21.10	0.36	8.33	146.72	0.31	8.33	40.28	2.26	9.64
		R	97.64	0.27	8.33	122.80	0.44	8.33	27.53	0.28	17.35	16.52	0.28	8.33	109.61	0.26	8.33	31.74	1.93	9.74

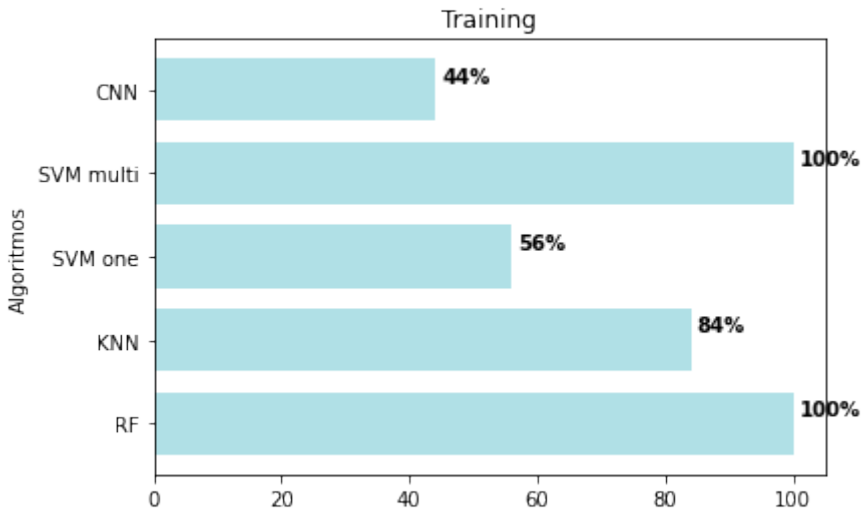
Nota: *Tabla con los resultados de las métricas de rendimiento clasificado por tipo de imagen utilizada. Elaborado por: Los autores.*

3.6 PRECISIÓN DE LOS MODELOS DE CLASIFICACIÓN

Para validar que la librería DicomClassifier cumpla con un nivel de precisión aceptable se realizó un modelo para cada técnica de clasificación y de esta forma analizar la precisión individualmente. Durante esta etapa se utilizó la Máquina 1 que se muestra en la Tabla 2 ya que cuenta con más recursos y poder generar un modelo que permita evaluar la precisión de manera adecuada. Se trabajó con una parte del dataset estableciendo 25 imágenes para la parte del entrenamiento de cada modelo. Se obtiene el porcentaje de precisión (accuracy) para cada técnica sobre el mismo conjunto de imágenes de entrenamiento, obteniendo los resultados de la Figura 29. Al tratar de predecir sobre las mismas imágenes que se utilizaron para entrenar el modelo se debería tener un alto porcentaje de precisión como se puede observar para SVM-MULTI, RF y KNN. Por otro lado, SVM-ONE y CNN tiene como resultado una precisión baja, lo que nos indica que necesita más imágenes para su entrenamiento.

Figura 29

Precisión en el training

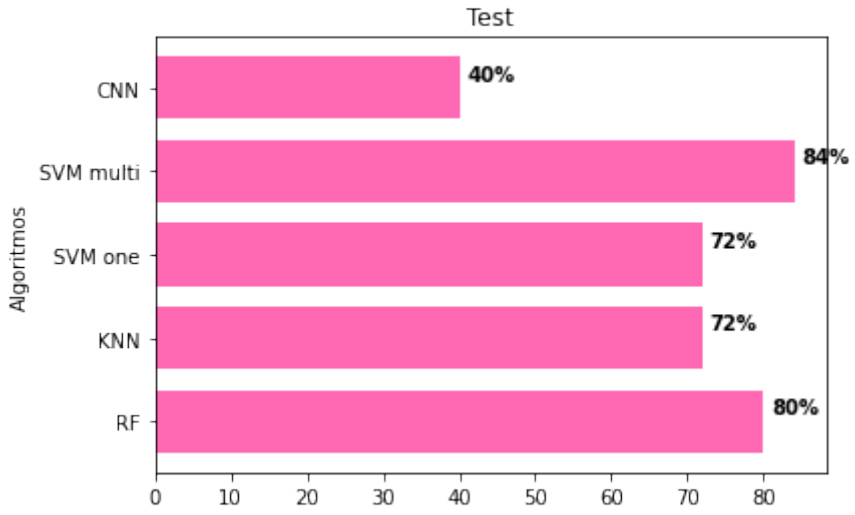


Nota: *Precisión en el training de los algoritmos con las imágenes de entrenamiento.*
Elaborado por: Los autores.

Del dataset se tomó otras 25 imágenes diferentes que nos serviran para evaluar la precision sobre un conjunto de imágenes de prueba. Al tratar de predecir sobre nuevas imágenes el porcentaje de precisión deberia disminuir en cierta medida en comparación a la precisión sobre las imágenes de entrenamiento.

Figura 30

Precisión en el test



Nota: *Precisión en el test de los algoritmos con las imágenes de entrenamiento. Elaborado por: Los autores.*

En la Figura 30 se puede observar que SVM-MULTI y RF siguen teniendo un porcentaje alto al igual que KNN. Mientras que SVM-ONE mejoro su precisión. Por otro lado la precisión para CNN sigue manteniéndose bajo. De manera general observamos que la precision sobre las imágenes de prueba sigue la misma tendencia que la precision sobre las imágenes de entrenamiento.

CONCLUSIONES

- El uso del Machine Learning esta cada vez más presente en nuestra vida diaria. Existen muchas técnicas de clasificación de imágenes pero muy pocas están enfocadas al tratamiento de imágenes médicas DICOM. Gracias a las actuales capacidades de computo es posible realizar algoritmos que ayuden al área de la medicina y aportar en cierta forma al análisis de mamografías.

- La librerías deben ser pensadas y estructuradas para que sean adaptables a diferentes entornos de desarrollo. Por lo que no deben depender de funcionalidades exclusivas de un framework. De esta manera se pueda integrar de una manera sencilla a las herramientas que utiliza un desarrollador.

- Los algoritmos de clasificación obtienen mejores resultados dependiendo de varios parámetros o los datos entrenamiento recibido, al utilizar programación multihilo Random Forest destaca en todas las métricas evaluadas durante la ejecución de la librería DicomClassifier.

- La librería desarrollada tiene un buen rendimiento, debido a que el rendimiento máximo del CPU es de 25.87 %, teniendo en cuenta que se puede optimizar más los algoritmos que presenten.

- El promedio general de rendimiento de la librería DicomClassifier es de 11.45 %, únicamente teniendo el cuenta la métrica de CPU que consideramos la más destacada, seguido de un uso de memoria de 0.63 GB y un tiempo de ejecución promedio de 101 segundos.

RECOMENDACIONES

- Para mejorar el rendimiento de los algoritmos se sugiere adaptar el código fuente para que funcione con programación en paralelo, a demás de la posibilidad de trabajar los bucles o iteraciones con lenguajes como Fortran o ensamblador para mejorar el tiempo de ejecución y aprovechar los núcleos del CPU al máximo.

- Se recomienda realizar el entrenamiento con una mayor cantidad de imágenes y con datasets que contengan imágenes DICOM normalizadas para obtener mejores resultados en la clasificación.

- El presente trabajo se enfoca en la clasificación de imágenes crudas. Se recomienda complementarlo con distintos módulos que abarquen técnicas de pre-procesamiento, segmentación y extracción de características de imágenes DICOM.

- Las imágenes DICOM al tener una resolución muy alta, las distintas técnicas de clasificación tienden a presentar problemas al diferenciar las características esenciales de la imagen. Además de que el consumo de memoria puede ser alto en datasets con muchas imágenes. Por lo que se recomienda aplicar técnicas de extracción de características para los datasets.

REFERENCIAS

Aceves, T. (2015). *Administración de plantas médicas hospitalarias* (Tesis Doctoral no publicada).

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer.

Aguilar, L. J., Zahonero, I., & Martínez, I. Z. (2005). *Programación en C: metodología, algoritmos y estructuras de datos*. Descargado de `http://books.google.com/books?id=9NqxAAAACAAJ&pgis=1`

Álvaro, M., & Stefany, N. (2020). *Desarrollo de una herramienta para detección de tejido anómalo en mamografías digitales usando redes neuronales convolucionales* (Tesis Doctoral no publicada). ESCUELA POLITÉCNICA NACIONAL.

Caffery, L. J., Rotemberg, V., Weber, J., Soyer, H. P., Malvehy, J., & Clunie, D. (2021). The Role of DICOM in Artificial Intelligence for Skin Disease. *Frontiers in Medicine*, *7*. doi: 10.3389/fmed.2020.619787

Cancer.net. (2021, sep). *Mamografía.* https://www.cancer.net/es/desplazarse-por-atención-del-cáncer/diagnóstico-de-cáncer/pruebas-y-procedimientos/mamografía.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 27:1–27:27. (Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`)

Chethan, K. S., Vishwanath, S., Patil, R. V., & Vijetha, K. A. (2020). Segementation and Prediction from CT Images for Detecting Lung Cancer. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*. doi: 10.1109/ICCCNT49239.2020.9225486

Chitarroni, H. (s.f.). *La regresión logística* (Inf. Téc.). Universidad del Salvador. Descargado de `http://www.salvador.edu.ar/csoc/idicso`

CMake. (s.f.). Descargado 2022-02-07, de https://cmake.org/

Code::Blocks - Code::Blocks. (s.f.). Descargado 2022-02-07, de https://www.codeblocks.org/

Cortes, C., & Vapnik, V. (1995, 01 de Sep). Support-vector networks. *Machine Learning*, *20*(3), 273-297. Descargado de https://doi.org/10.1007/BF00994018 doi: 10.1007/BF00994018

Cowley, J. (s.f.). *Redes neuronales convolucionales – IBM Developer.* Descargado 2022-02-06, de https://developer.ibm.com/es/technologies/artificial-intelligence/articles/cc-convolutional-neural-network-vision-recognition/

Cristina Pérez Verona, I., & García, L. A. (2016). Una revisión sobre aprendizaje no supervisado de métricas de distancia. *Revista Cubana de Ciencias Informáticas*, *10*(4), 43–67.

Cruz, J. A., & Wishart, D. S. (2006). *Applications of machine learning in cancer prediction and prognosis* (Vol. 2). doi: 10.1177/117693510600200030

Cruz-Morales, R. A., Villaseñor-Navarro, Y., Pavón-Hernández, C. M., Pérez-Badillo, M. P., Aguilar-Cortázar, L. O., & Pérez-Zúñiga, I. (2012). Breast microcalcification: A challenge for the diagnosis. *Gaceta Mexicana de Oncologia*, *11*(4), 251–259.

DCMTK - DICOM Toolkit. (2022, feb). Descargado 2022-02-07, de https://dicom.offis.de/dcmtk.php.en

Falcou, J. (2015, sep). Designing hpc libraries in the modern c++ world. En *Proceedings of the 2015 international conference on high performance computing and simulation, hpcs 2015* (p. 458-459). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/HPCSim.2015.7237076

Francisco Jose Torres Hoyos. (2012). *ESTUDIO in vivo DE LA DINAMICA DEL CRECI-*

MIENTO TUMORAL CEREBRAL MEDIANTE EL ANALISIS DE ESCALAMIENT (Tesis Doctoral no publicada). UNIVERSIDAD SIMÓN BOLÍVAR.

Géron, A. (2019). *Hands-on Machine Learning whith Scikit-Learing, Keras and Tensorfow*.

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental Concepts of Convolutional Neural Network. En V. E. Balas, R. Kumar, & R. Srivastava (Eds.), *Recent trends and advances in artificial intelligence and internet of things* (pp. 519–567). Cham: Springer International Publishing. Descargado de https://doi.org/10.1007/978-3-030-32644-9_36 doi: 10.1007/978-3-030-32644-9_36

Hung, M. C., & Yang, D. L. (2001). An efficient fuzzy C-means clustering algorithm. En *Proceedings - ieee international conference on data mining, icdm* (pp. 225–232). doi: 10.1109/icdm.2001.989523

Hussain, F. (s.f.). *Your Handbook to Convolutional Neural Networks.* Descargado 2022-02-06, de https://medium.com/analytics-vidhya/your-handbook-to-convolutional-neural-networks-628782b68f7e

Kaestner, C. A. A. (2013). Support Vector Machines and Kernel Functions for Text Processing. *Revista de Informática Teórica e Aplicada*, *20*(3), 130. doi: 10.22456/2175-2745.39702

Kamalakannan, J., Thirumal, T., Vaidhyanathan, A., & Mukeshbhai, K. D. (2015). Study On Different Classification Technique for Mammogram Image. *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, 1-5. doi: 10.1109/ICCPCT.2015.7159456

Kumari, N., & Saxena, S. (2018). Review of Brain Tumor Segmentation and Classification. *Proceedings of the 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT 2018*, 1-6. doi: 10.1109/ICCTCT.2018.8551004

Malacara, D. (s.f.). *LA HOLOGRAFÍA*. Descargado 2022-02-07, de http://

bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_8.htm

MathWorks. (2018). *What Is Machine Learning? | How It Works, Techniques & Applications - MATLAB & Simulink* (n.º May 2016). Descargado 2022-02-06, de https://www.mathworks.com/discovery/machine-learning.html

Medina-Merino, R. F., & Ñique-Chacón, C. I. (2017). Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python. *Interfases*(010), 165. Descargado de https://www.kaggle.com/primaryobjects/voicegender doi: 10.26439/interfases2017.n10.1775

Ministerio de Salud Pública. (2021, sep). *Cifras de Ecuador – Cáncer de Mama.* https://www.salud.gob.ec/cifras-de-ecuador-cancer-de-mama.

Nascimento, S., Mirkin, B., & Moura-Pires, F. (2000). Fuzzy clustering model of data and fuzzy c-means. En *Ieee international conference on fuzzy systems* (Vol. 1, pp. 302–307). IEEE. doi: 10.1109/fuzzy.2000.838676

Ninla Elmawati Falabiba. (2019). *IMPLEMENTACIÓN DE UNA LIBRERÍA QUE PREPROCESE CÓDIGO ESCRITO EN C++ SIGUIENDO EL ESTÁNDAR ISO/IEC 14882 Y CREAR UNA HERRAMIENTA QUE HAGA USO DE ELLA PAÚL* (Tesis Doctoral no publicada).

Okamoto, S., & Kohana, M. (2016). A C++ Header Library for Web Applications. *NBiS 2016 - 19th International Conference on Network-Based Information Systems*, 541-545. doi: 10.1109/NBiS.2016.41

Organizacón Mundial de la Salud. (2021). *Cáncer de Mama.* https://www.who.int/es/news-room/fact-sheets/detail/breast-cancer.

Paolettia, M. E., Haut, J. M., Plaza, J., & Plaza, A. (2019). A comparative study of techniques for hyperspectral image classification. *RIAI - Revista Iberoamericana de Automatica e*

Informatica Industrial, *16*(2), 129–137.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, P., & Saraiva, J. (2021). Ranking programming languages by energy efficiency.

Pirklbauer, K., Plosch, R., & Weinreich, R. (1993). Libraries and tools for object-oriented distributed automation software. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, *4*(October), 403-408. doi: 10.1109/icsmc.1993.390746

QT.io. (s.f.). *About Qt.* Descargado 2021-09-15, de `https://wiki.qt.io/About_Qt`

Raimi, K. (s.f.). *10 stochastic gradient descent optimisation algorithms + cheatsheet.* Descargado 2022-02-07, de `https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9`

Routray, I., & Rath, N. P. (2018). Textural Feature Based Classification of Mammogram Images Using ANN. *2018 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018*, 10–15. doi: 10.1109/ICCCNT.2018.8493957

Sarmiento-Ramos, J. L. (2020, jun). Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica. *Revista UIS Ingenierías*, *19*(4), 1–18. doi: 10.18273/revuin.v19n4-2020001

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471. doi: 10.1162/089976601750264965

Sharma, S., Sharma, S., & Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, *04*(12), 310–316. Descargado de `http://www.ijeast.com` doi: 10.33564/ijeast.2020.v04i12.054

Sociedad Americana Contra el Cáncer. (2022). *¿Qué busca el médico en un mamo-*

grama? https://www.cancer.org/es/cancer/cancer-de-seno/pruebas-de-deteccion-y-deteccion-temprana-del-cancer-de-seno/mamogramas/que-busca-el-medico-en-un-mamograma.html.

Srivastava, A. (2013). *Fundamentals of Medical Imaging Technology Hydrogen sensor View project fundamentals of Physics View project* (Inf. Téc.). Descargado de https://www.researchgate.net/publication/313791570

Sung, H., Ferlay, J., Siegel, R. L., Laversanne, M., Soerjomataram, I., Jemal, A., & Bray, F. (2021). Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries. *CA: A Cancer Journal for Clinicians*, *71*(3), 209-249. doi: 10.3322/caac.21660

Suryansh S. (2018). *Neural Networks: All YOU Need to Know – Towards Data Science.* Descargado 2022-02-06, de https://towardsdatascience.com/nns-aynk-c34efe37f15a

Wen, B., Zeng, W. F., Liao, Y., Shi, Z., Savage, S. R., Jiang, W., & Zhang, B. (2020, nov). Deep Learning in Proteomics. *Proteomics*, *20*(21-22). doi: 10.1002/pmic.201900335

Zoltan, C. (s.f.). *SVM and Kernel SVM.* Descargado 2022-02-06, de https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

ANEXOS

GUÍA DE INSTALACIÓN PARA LIBRERIA DICOMCLASSIFIER

Dicom Classifier

Dicom Classifier es una libreria para manejar imagenes médicas dicom y con diferentes técnicas de clasificación.

Dependencias

La libreria usa: - **libdcmtk-dev**: para el manejo de imágenes dicom.

Para instalarlos puedes ejecutar el script `checker.sh`

```
chmod +x ./checker.sh
./checker.sh
```

Tambien se pueden instalar de manera manual:

```
sudo apt install libdcmtk-dev build-essential cmake
```

Instalación

Antes de usar la libreria se necesita realizar el proceso de compilación para generar los archivos necesarios e importarlos directamente en su proyecto.

QT

Para compilar la libreria se puede usar QT creator.

1. Primero se clona el proyecto en su computadora.
2. Abrir el archivo .pro con QT Creator.
3. Configurar la carpeta donde se compilará
4. Compilar el proyecto

CMake

En caso de compilar con cmake asegurese de tener instalado la version `>3.0`

```
mkdir build
cd build
cmake ..
make
```

Uso

Para usar esta libreria asegurese de haber compilado este proyecto y haber generado los archivos `.o` y `.a`.

La libreria se ha usado con los IDEs QT Creator y CodeBlocks.

> **Nota:** Algunos algoritmos hacen uso de procesamiento multi-hilo, para ellos asegurese de usar el flag `-pthread` e incluir la libreria `-lpthread`.

QT

Para usarlo con QT inicialice un nuevo proyecto y asegurece que su archivo de configuracion `.pro` tenga lo siguiente:

```
INCLUDEPATH += /dicom-classifier/DicomClassifier \ # Aqui es el path del proyecto con las cabeceras.

LIBS += -L/dicom-classifier/build \  # Directorio con los archivos compilados
    -lDicomClassifier \
    -ldcmdata \
    -ldcmimgle \
    -ldcmimage \
    -ldcmjpeg \
    -lpthread
```

Codeblocks

Para usarlo en codeblocks inicie un nuevo proyecto y asegurece de ejecutar el script `checker.sh`.

Dirigirse a configuraciones y compilador, en la pestaña **linker settings** agregue el archivo `libDicomClasifier.a` que se encuentra en la carpeta **build** despues del proceso de compilacion. Tambien incluya las siguientes opciones:

```
-lDicomClassifier
-ldcmdata
-ldcmimgle
-ldcmimage
-ldcmjpeg
-lpthread
```

Ahora en la pestaña `Search directories > compiler` agregue el directorio donde se encuentran los archivos `.h` del proyecto.

```
/home/user/projects/dicom-classifier/DicomClassifier
```

En la pestaña `Search directories > linker` agregue el directorio donde se se encuentran los archivos compilados.

```
/home/user/projects/dicom-classifier/build
```