UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGERNIERÍA EN SISTEMAS COMPUTACIONALES

TEMA:

Modelo Machine Learning para la detección de tumores cerebrales mediante un conjunto de imágenes de resonancias magnéticas

AUTOR:

Castellano Sánchez, Kevin Lester

Trabajo de titulación previo a la obtención del título de

INGENIERO EN SITEMAS COMPUTACIONALES

TUTOR:

Ing. Célleri Mujica, Colón Mario, Mgs

Guayaquil, Ecuador

15 de septiembre de 2022

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENNIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

CERTIFICACIÓN

Certificamos que el presente trabajo de titulación, fue realizado en su totalidad por **Castellano Sánchez, Kevin Lester**, como requerimiento para la obtención del título de **Ingeniero en Sistemas Computacionales**.

TUTOR

f. _____
Ing. Célleri Mujica, Colón Mario, Mgs.

Guayaquil, a los 15 días del mes de septiembre del año 2022

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENNIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Castellano Sánchez, Kevin Lester**

DECLARO QUE:

El Trabajo de Titulación, **Modelo Machine Learning para la detección de tumores cerebrales mediante un conjunto de imágenes de resonancias magnéticas** previo a la obtención del título de **Ingeniero en Sistemas Computacionales**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 15 días del mes de septiembre del año 2022

EL AUTOR

f. _____
Castellano Sánchez, Kevin Lester

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENNIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

AUTORIZACIÓN

Yo, **Castellano Sánchez, Kevin Lester**

Autorizo a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del Trabajo de Titulación, **Modelo Machine Learning para la detección de tumores cerebrales mediante un conjunto de imágenes de resonancias magnéticas**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

Guayaquil, a los 15 días del mes de septiembre del año 2022

EL AUTOR:

f. _____

Castellano Sánchez, Kevin Lester

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRIBUNAL DE SUSTENTACIÓN

f. _____

ING. ANA CAMACHO CORONEL, MGS

DIRECTORA DE CARRERA

f. _____

ING. FERNANDO CASTRO AGUILAR, PhD

DOCENTE DE LA CARRERA

f. _____

ING. GALO CORNEJO GOMEZ, MGS

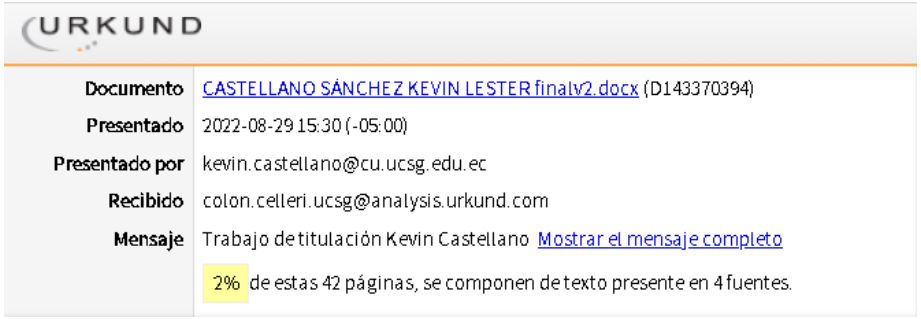
OPONENTE

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENNIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

REPORTE URKUND



URKUND

Documento	CASTELLANO SÁNCHEZ KEVIN LESTER finalv2.docx (D143370394)
Presentado	2022-08-29 15:30 (-05:00)
Presentado por	kevin.castellano@cu.ucsg.edu.ec
Recibido	colon.celleri.ucsg@analysis.urkund.com
Mensaje	Trabajo de titulación Kevin Castellano Mostrar el mensaje completo
	2% de estas 42 páginas, se componen de texto presente en 4 fuentes.

Fecha de elaboración: 29/agosto 2022
Firma:

Nombre del tutor: Mario Célleri Mujica
Tutor de Trabajo de Integración Curricular
Carrera Ingeniería en Computación

AGRADECIMIENTO

Primero agradecerle a Dios por haberme permitido llegar hasta este día, logrando así cumplir con el objetivo. Por la familia que tengo. Por haber guiado a lo largo de este camino y sobre todo haberme dado la fortaleza para poder seguir adelante.

A mis padres, Siempre han sido mis mejores guías de vida. Gracias por su apoyo incondicional, por estar siempre allí, por ser quienes son y por creer en mí. Gracias por sus consejos, sus enseñanzas basadas en ejemplos… muchas gracias… también son parte de logro.

A los docentes de la facultad, muchas gracias por sus enseñanzas no solo las de carácter académico sino aquellas que se basan en la experiencia que tienen como excelentes profesionales. Un agradecimiento especial a mi tutor el Ing. Mario Célleri, muchas gracias por la directriz, la motivación y la guía durante el desarrollo del presente trabajo.

Finalmente, a mis amigos y compañeros que formaron parte de este proceso de formación.

Kevin Lester Castellano Sanchez

DEDICATORIA

Dedico el resultado de este trabajo a toda mi familia. Principalmente a mis padres quienes me apoyaron y estuvieron conmigo en todo momento. Son el pilar fundamental de mis logros y la fuente de mi motivación.

Kevin Lester Castellano Sanchez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El Machine Learning se utiliza en algunos campos del conocimiento, como la medicina, por su predicción de la eficacia en tratamientos de enfermedades y en la toma de decisiones luego de la identificación de patrones. Esto permite crear herramientas de soporte para diagnóstico, como el Modelo ML para la detección de tumores cerebrales mediante resonancias magnéticas que se propone, por medio del cual se permitirá la automatización de dicho proceso, para agilizar el diagnóstico del médico tratante. Se conoció sobre las redes neuronales, el aprendizaje por transferencia, la arquitectura ResNet 50, Deep Learning, además de las redes neuronales convolucionales. Se utilizó la metodología descriptiva con enfoque cualitativo y entrevista a profesionales médicos, además del prototipado evolutivo como metodología de desarrollo. Se conoció que la tecnología tiene relevancia en la medicina, y que sí es conveniente implementar una herramienta que sirva de soporte para la toma de decisiones en cuanto a diagnóstico de tumores cerebrales. De la evaluación del prototipo se obtuvo una tasa de aciertos aceptable para el desarrollo de la herramienta. Al finalizar, se propusieron algunas conclusiones y recomendaciones a considerar para posibles trabajos futuros.

Palabras Clave: *Machine Learning, red neuronal convolucional, aprendizaje por transferencia, Deep Learning.*

ABSTRACT

Machine Learning is used in some fields of knowledge, such as medicine, for its prediction of the effectiveness of disease treatments and decision making after the identification of patterns. This allows the creation of support tools for diagnosis, such as the ML Model for the detection of brain tumors through magnetic resonance imaging that is proposed, which will allow the automation of this process, to speed up the diagnosis of the treating physician. We learned about neural networks, transfer learning, ResNet 50 architecture, Deep Learning, and convolutional neural networks. A descriptive methodology with qualitative approach and interviews with medical professionals was used, in addition to evolutionary prototyping as a development methodology. It was learned that technology has relevance in medicine, and that it is convenient to implement a tool to support decision making in the diagnosis of brain tumors. From the evaluation of the prototype, an acceptable success rate for the development of the tool was obtained. At the end, some conclusions and recommendations to be considered for possible future work were proposed.

Keywords: *Machine Learning, convolutional neural network, transfer learning, Deep Learning****.***

INTRODUCCIÓN

Con el paso del tiempo, la industria de la tecnología ha ido produciendo gran cantidad de herramientas innovadoras y tendencias para dar solución a las crecientes necesidades de empresas y consumidores. El futuro de la tecnología es muy difícil de predecir, ya que las continuas transformaciones a las que se ve sometida provocan una serie de cambios y modificaciones en los productos y/o servicios ofertados, lo que significa que para que un adelanto tecnológico sea aceptado y sea de utilidad al consumidor, deberá ser práctico y rentable.

Las tendencias tecnológicas que se visualizaron desde 2021 para el presente año son variadas. *Robotic Procesing Automatization*, desarrollo de software, Inteligencia Artificial (IA), Aprendizaje Automático o *Machine Learning* (ML) y sistemas inteligentes (Forbes, 2021; Syntonize, 2021) son algunas de las tecnologías que pueden ser capaces de modificar en entorno de la industria y la informática alrededor del mundo.

De las tecnologías mencionadas en el párrafo anterior, cabe mencionarse la utilidad que tiene el ML en distintos ámbitos de la ciencia. El ML es una subárea de la IA muy eficaz, con exitosos resultados de aplicación en los diversos tipos de industrias (Álvarez et al., 2020) y áreas del conocimiento, dentro de las que se puede mencionar la medicina.

En concordancia con lo anterior, se conoce que en años recientes los datos clínicos de los pacientes ya se encuentran almacenados y disponibles de manera electrónica, y esta facilidad permite que el área médica se transforme en el entorno preciso para desarrollar y aplicar nuevas herramientas tecnológicas. En este contexto, el ML es una tecnología que tiene la capacidad de ofrecer mejoras a los sistemas médicos, por cuanto puede analizar miles de datos e información clínica de los pacientes y “crear modelos pronósticos, de tamizaje y diagnóstico” (Álvarez et al., 2020, p. 2). Aunque son indiscutibles las ventajas de los algoritmos del ML como métodos para cambiar los sistemas de salud y mejorar la atención a los pacientes, todavía se requiere del correcto proceso de evaluación para apreciar sus beneficios.

La medicina es un área muy extensa y poco a poco ha ido adquiriendo mayor complejidad. Hoy en día los profesionales de la salud y los científicos ponen a disposición de sus pacientes mayor cantidad de terapias, medicinas, procedimientos médicos, y el procesamiento de tal cantidad de datos supera la mente humana, por lo que se requiere de la búsqueda de nuevas herramientas que ayuden a la labor de integración de todos los datos de los pacientes, reconocimiento de patrones y creación de modelos que sirvan para solucionar las barreras de los doctores, reduciendo la carga de trabajo, mejorando y personalizando la atención, con el consiguiente ahorro de recursos.

De esto se puede entender que existirían espacios no atendidos en las diversas especialidades médicas, que se podrían atender con alguna solución tecnológica, con el fin de apoyar el problema que pueda existir. Este puede ser el caso de profesionales médicos que atienden a pacientes con tumores cerebrales, a quienes se diagnostica a través de resonancias magnéticas para constatar el padecimiento del paciente, por lo que se recomendaría a esos casos implementar un modelo de ML que permita detectar tumores cerebrales a través de imágenes de resonancias magnéticas para facilitar el diagnóstico del médico.

Para un mejor conocimiento del desarrollo de este proyecto, se lo ha dividido en cuatro capítulos. En el **primer capítulo** se explica el problema de investigación y su planteamiento, su ubicación dentro de un contexto, sus causas y consecuencias, su delimitación, la evaluación del problema, las preguntas de investigación, los objetivos (general y específicos), el alcance, la justificación e importancia y las variables de investigación. En el **segundo capítulo** se estructura la parte teórica y conceptual del problema, en donde se revisan temas relacionados con el tema de estudio, incluyendo teorías, definiciones y aspectos legales inherentes. El **tercer capítulo** define la metodología de la investigación, las técnicas para el levantamiento de la información, la población y muestra, los instrumentos de recolección y el análisis de los resultados; en este capítulo se incluye la metodología de desarrollo utilizada en la creación de la solución tecnológica. En el **cuarto capítulo** se describe la propuesta, en donde se especifica el uso de las

herramientas tecnológicas, la metodología de desarrollo utilizada y toda la información tecnológica que se utilizó en la solución. Finalmente, se describen las conclusiones y recomendaciones, además de las referencias bibliográficas.

CAPÍTULO I

EL PROBLEMA

PLANTEAMIENTO DEL PROBLEMA

1.1 Ubicación del Problema en un Contexto

La tecnología aplicada a la ciencia médica se orienta hacia la aplicación de la IA y la subárea de ML, a través de la utilización de modelos diseñados e implementados para la búsqueda de información médica y el descubrimiento de nuevos conocimientos que sirvan para agilizar los diagnósticos y ofrecer a los pacientes mejores y más certeros tratamientos para sus enfermedades. Los avances tecnológicos últimos en cuanto a informática y computación han hecho de la IA una parte importante en la atención médica actual. Los métodos, algoritmos y aplicaciones desarrolladas mediante la IA son utilizados en el área médica para colaborar con los profesionales en su entorno y en investigaciones que se llevan a cabo (IBM, s/f).

Con este preámbulo, cabe resaltarse que la tecnología en favor de la ciencia médica se remite al apoyo al profesional médico en la toma de decisiones en cuanto a diagnóstico de enfermedades para una mejor aplicación de tratamientos y todo lo que conlleva la atención al paciente. La tecnología facilita el acceso inmediato a la información, tanto para consultas sobre patologías y tratamientos, como de datos almacenados de los pacientes (IBM, s/f).

Los diagnósticos de tumores cerebrales son realizados a través de exámenes radiológicos complejos, como lo son las tomografías computarizadas, radiografías o resonancias magnéticas; estas últimas son exámenes que crean “imágenes del cerebro y de los tejidos nerviosos circundantes” (MedlinePlus, s/f, párr. 1). Los resultados de estos exámenes son analizados a través de herramientas de IA para buscar algún tipo de lesión o encontrar algún otro problema cerebral que un profesional de radiología podría no distinguir.

A pesar de las ventajas que ofrece la tecnología en la rama de la radiología, no se conoce con certeza que, en los centros médicos tanto privados como estatales, la utilización de modelos de ML para detectar tumores cerebrales y

agilizar el trabajo del profesional médico sea una realidad, por lo que sería conveniente que se adopten este tipo de modelos para mejores diagnósticos.

1.2 Causas y Consecuencias del Problema

La falta del desarrollo de un modelo de ML que sirva como herramienta para que el profesional de la salud agilice el diagnóstico de tumores cerebrales significaría que el área médica de las instituciones de salud no haría uso extensivo de las ventajas de esta tecnología para la interpretación de los exámenes imagenológicos, o no disponen de sistemas médicos actualizados ni eficientes. Esto puede provocar que los resultados de las imágenes se demoren para la obtención de los informes, lo que es contraproducente para el diagnóstico y tratamiento de los pacientes, que pueden encontrarse en desventaja y que podría evitarse si existiera un diagnóstico inicial y oportuno que marque “la diferencia entre un tratamiento oportuno o un desenlace fatal” (Hidago et al., 2021, p. 1).

1.3 Delimitación del Problema

Campo: Desarrollo de software.

Área: Inteligencia Artificial.

Aspecto: Machine Learning.

Tema: Modelo Machine Learning para la detección de tumores cerebrales. mediante un conjunto de imágenes de resonancias magnéticas cerebrales.

1.4 Formulación del Problema

¿Cómo un modelo de ML puede automatizar el proceso de detección de tumores cerebrales para agilizar el diagnóstico médico?

1.5 Evaluación del Problema

El proyecto a desarrollar es **evidente**, ya que es posible visualizar cómo se lleva a cabo el proceso de diagnóstico de tumores cerebrales. El paciente se realiza la resonancia magnética y los resultados pasan al profesional médico para su revisión e interpretación.

Es **claro**, ya que su planteamiento y redacción son precisos, por lo que se lo puede comprender fácilmente y las ideas que se plantean son concretas.

Es **relevante** ya que el modelo ML busca agilizar el proceso de detección de tumores cerebrales, de manera que los profesionales médicos puedan ofrecer una mejor calidad de atención a los pacientes.

Es **original**, ya que es un campo que tiene poco análisis y aplicación conocida en el medio.

Es **factible**, puesto que a través de esta investigación se conocerán los antecedentes del problema planteado y construir el modelo ML que servirá al profesional de la salud para agilizar el proceso de detección de tumores cerebrales.

1.6 Objetivos

1.6.1 Objetivo General

Desarrollar un prototipo de modelo de ML que permita la automatización del proceso de detección de tumores cerebrales por medio de imágenes de resonancias magnéticas, para agilizar el diagnóstico del médico tratante.

1.6.2 Objetivos Específicos

- Revisar las distintas teorías relacionadas al ML, su importancia y su aplicabilidad en la detección de tumores cerebrales.
- Analizar las herramientas requeridas para el modelo ML para el proceso de automatización de detección de tumores cerebrales.
- Desarrollar el prototipo del modelo ML para la automatización del proceso de detección de tumores cerebrales.
- Evaluar los resultados del prototipo del modelo ML.

1.7 Alcances del Problema

El modelo propuesto, a través del entrenamiento con el algoritmo de ML con base en las redes neuronales, verificará la presencia o no de un tumor cerebral en el paciente. Para el desarrollo del modelo se usará la plataforma web Google Colabs, que va a permitir el uso de las librerías tales como Tensorflow o Keras, así como no depender de la capacidad de cómputo del hardware físico para la construcción del modelo ML. Este trabajo no contempla la clasificación de los distintos tipos de tumores cerebrales que pueda padecer un paciente o si es maligno o benigno; tampoco se especificará la ubicación del tumor en el cerebro del paciente.

Para el entrenamiento del modelo se utilizará un set de datos públicos alojados en la página web de Kaggle, que reúne gran cantidad de datos para la comunidad del Data Science. El modelo presentará una interfaz, que permitirá la carga de la imagen de la resonancia magnética y a través de un botón se podrá predecir si el paciente tiene o no un tumor cerebral con base en el modelo entrenado. Se manejará un usuario administrador.

1.8 Justificación e importancia

El desarrollo de este proyecto se justifica por la exigencia actual de bridar una atención médica de calidad y con los mejores diagnósticos y tratamientos para los distintos padecimientos de salud de los pacientes, por lo que disponer de las más modernas herramientas tecnológicas que ayuden en tal cometido, es una prioridad para los profesionales médicos de las instituciones de salud públicas o privadas.

La IA y su subárea ML son las herramientas actuales de gran utilidad en la ciencia médica, puesto que ayudan en el diagnóstico temprano de enfermedades. Los sistemas médicos que utilizan esta tecnología están en la capacidad de *aprender* de los escenarios que determinan las enfermedades de los pacientes.

El modelo de ML para la detección de tumores cerebrales propuesto en este proyecto podría beneficiar a los profesionales médicos para un diagnóstico más ágil de estos problemas, con el fin de ofrecer mejor atención a los pacientes.

1.9 Preguntas de investigación

Se plantean algunos supuestos que se comprobarán durante la investigación, como unas posibles soluciones. Son:

¿Cuáles son las teorías que se relacionan con el ML?

¿Cuáles son las herramientas que el modelo ML requiere para automatizar la detección de tumores cerebrales?

¿Cómo se implementará el prototipo de modelo de ML para la automatización la detección de tumores cerebrales?

¿Cómo se evaluarán los resultados del prototipo del modelo ML?

CAPÍTULO II
MARCO TEÓRICO

2.1 Teorías y principios sobre Machine Learning (ML)

2.1.1 Antecedentes

El Machine Learning (ML) o Aprendizaje Automático es una disciplina de la Inteligencia Artificial (IA) que integra algunas estrategias de análisis, que tienen como finalidad desarrollar algoritmos que sirvan para obtener información de los datos, con el fin de explicarlos, clasificarlos o predecir. Aunque es frecuente identificar el ML con la IA, cabe mencionarse que aunque la IA "es una clasificación aún más amplia que incluye tanto técnicas para el análisis de datos estructurados como el aprendizaje automático y datos no estructurados como procesamiento de lenguaje natural" (Pedrero et al., 2021, p. 249). Es en los años 50 que aparece el ML como una técnica "para emular, computacionalmente, elementos del proceso cognitivo humano a través de reconcomiendo de patrones y procesos de toma de decisión" (Pedrero et al., 2021, p. 249).

De acuerdo a Maisueche (2019), el ML tiene su fundamento en las matemáticas, estadística y computación. Los avances en informática aparecieron en los años 40 y, antes de esa fecha, "hubo varios métodos y técnicas matemáticas y estadísticas que sirvieron de base para lo que hoy se conoce como ML: el Teorema de Bayes, el ajuste por Mínimos Cuadrados y las Cadenas de Márkov" (Maisueche, 2019, p. 30).

La definición técnica del Teorema de Bayes, de 1763, se refiere a que lo utiliza para el cálculo de la probabilidad de un evento, del que se tiene información previa (J. López, 2021). Según Abdi (2003) referenciado por Maisueche (2019, p. 30) "el ajuste por Mínimos Cuadrados (1805) es una técnica de optimización matemática que busca la función continua que mejor se aproxime a una serie de datos según el criterio de mínimo error cuadrático"; de acuerdo a Espinosa et al. (2016) este método es utilizado "para encontrar la solución de sistemas de ecuaciones lineales" (Espinosa et al., 2016, p. 43). Y la cadena de Márkov (1913) es un método cuantitativo con enfoque estocástico utilizado en la toma de decisiones; este tipo de enfoque plantea

estudio de variables aleatorias que tienen su evolución a través del tiempo. En el modelo de Márkov surge “una sucesión de variables aleatorias tal que el *siguiente* estado del proceso es independiente de los estados anteriores, siempre que sea conocido el estado presente” (E. López & Joa, 2017, p. 45), en donde estas variables constituyen los *estados de transición*.

Como se ha mencionado en párrafos anteriores, el inicio del ML se dio en los años 50. En la Tabla 1 se muestran los aspectos más relevantes desde su surgimiento.

Tabla 1. Inicios del ML

FECHA	EVENTO
1950	Alan Turing crea el Test de Turing para determinar si una máquina era realmente inteligente. El test consistía en que una máquina debía ser capaz de engañar a una persona y hacerle creer que no era un ordenador sino un humano.
1952	Arthur Samuel escribe el primer programa de ordenador capaz de aprender, que fue un juego de damas, que iba mejorando partida tras partida.
1956	Martin Minsky y John McCarthy, junto con Claude Shannon y Nathan Rochester, organizan la conferencia de Darthmouth de 1956, evento en donde se dio a conocer la IA.
1958	Frank Rosenblatt diseña el Perceptrón, la primera red neuronal artificial

Nota: Adaptado de González (2019)

En 1967 “se escribe el algoritmo *Nearest Neighbor*. Este hito está considerado como el nacimiento al campo del reconocimiento de patrones (*pattern recognition*) en computadores” (González, 2019, párr. 8).

Para la segunda mitad de los 70 la IA sufre el primer revés, cuando las agencias que suministraron fundos para su estudio, hacen recorte de fondos, luego de años de pocos avances con expectativas altas. Este período se conoce como el *invierno* de la IA (González, 2019; Maisueche, 2019). En 1979 se crea el *Stanford Cart*, un robot de movimiento autónomo que evitaba obstáculos (González, 2019).

La década de los años 80 estuvo caracterizada por el surgimiento de los sistemas expertos, que se sustentaron en reglas que tuvieron su aceptación inmediata en las corporaciones, generando de este modo una nueva atracción hacia el ML. En la Tabla 2 se presentan otros hitos del surgimiento del ML.

Tabla 2. Otros hitos

FECHA	EVENTO
1981	Gerald Dejong introduce el concepto *Explanation Based Learning* (EBL), donde un computador analiza datos de entrenamiento y crea reglas generales que le permiten descartar los datos menos importantes.
1985	Terry Sejnowski inventa *NetTalk*, que aprende a pronunciar palabras como si fuese un niño.

Nota: Adaptado de González (2019)

Hacia fines de la década de 1980 y primera mitad de 1990, apareció el *segundo invierno* de la IA, cuyas consecuencias duraron algunos años. El renombre de la IA se recuperó durante el año 2000.

Tabla 3. Nuevos hitos del ML

FECHA	EVENTO
1990	El trabajo en ML pasa del enfoque orientado al conocimiento (*knowledge-driven*) hacia uno orientado al dato (*data-driven*). Se empiezan a desarrollar programas para análisis de volúmenes de datos para extraer conclusiones de los resultados.
1997	El ordenador *Deep Blue*, de IBM vence al campeón mundial de ajedrez Gary Kaspárov.
1998	La nueva versión *Deeper Blue* consiguió vencer todas las partidas de ajedrez.

Nota: Adaptado de González (2019) y Maisueche (2019)

El auge y adopción definitiva del ML (desde 2006) se debió a la necesidad de realizar cálculos con un enorme volumen de datos. Grandes organizaciones se encuentran orientando sus estructuras hacia los datos e incorporando técnicas de ML para la ejecución de sus procesos, elaboración de productos y servicios, para ser mejores frente a la competencia y conseguir mayores ventajas (González, 2019).

En la Tabla 4 se muestran nuevos hitos del aparecimiento del ML a partir del año 2006.

Tabla 4. ML desde 2006

FECHA	EVENTO
2006	Geoffrey Hinton acuña el término *Deep Learning* (Aprendizaje Profundo) para explicar nuevas arquitecturas de Redes Neuronales profundas capaces de aprender mucho mejor, modelos más planos.
2011	El ordenador Watson de IBM vence a sus competidores humanos en el concurso Jeopardy que consiste en contestar preguntas formuladas en lenguaje natural.
2012	Google y la Universidad de Stanford lideran el proyecto *GoogleBrain*, que desarrolla una Red Neuronal Profunda utilizando toda la capacidad de la infraestructura de Google para detectar patrones en vídeos e imágenes.
	Geoffrey Hinton lidera el equipo ganador del concurso de Visión por Computador a Imagenet utilizando una Red Neuronal Profunda (RNP). Se dio el nacimiento a la actual explosión de ML basado en RNPs
	Google X utiliza *GoogleBrain* para analizar autónomamente vídeos de Youtube y detectar aquellos que contienen gatos.
2014	Facebook desarrolla *DeepFace*, un algoritmo basado en RNPs para reconocer a personas con la misma precisión que un ser humano.
	Google compra *DeepMind*, que había demostrado las capacidades de las RNPs con un algoritmo capaz de jugar a juegos de Atari sólo viendo los píxeles de la pantalla, tal y como lo haría una persona. El algoritmo, luego de entrenarse, era capaz de batir a humanos expertos en algunos de esos juegos
	Amazon lanza su propia plataforma de ML
2015	Microsoft crea el *Distributed Machine Learning Toolkit*, que permite la distribución eficiente de problemas de ML en múltiples computadores.
	Elon Musk y Sam Altman, entre otros, fundan *OpenAI*, con el objetivo de asegurar que el desarrollo de la IA tenga un impacto positivo en la humanidad
2016	Google DeepMind vence en el juego Go al jugador profesional Lee Sedol por 5 partidas a 1. Jugadores expertos de Go afirman que el algoritmo fue capaz de realizar movimientos *creativos* que no se habían visto hasta el momento
2017	*OpenAI* despunta los avances en agentes conversacionales o *chatbots*, así como en el mundo de los videojuegos
2018	Google desarrolló BERT, la *primera representación de lenguaje bidireccional y sin supervisión* que se puede utilizar en una variedad de tareas de lenguaje natural, como responder preguntas.

Nota: Adaptado de Abeliuk (2021), González (2019) y Maisueche (2019)

2.1.2 Campos de aplicación del ML

Dentro de las aplicaciones del ML se encuentran: a) predicciones, b) detección de intrusos, c) antivirus, d) clasificación de texto, e) productividad, f) recomendación de productos, g) *bots* de soporte, h) *customer churn* (pérdida de clientes), i) lingüística computacional, j) *Deep learning*, k) biométrica (J.

Rodríguez, 2018), l) reconocimiento de voz, m) servicio al cliente, n) visión por computadora, o) motores de recomendación, p) negociación de acciones computarizada, q) detección de fraude (IBM, 2022). En cuanto a sus campos de aplicación, se conocen algunos (Hinestroza, 2018):

2.1.2.1 Medicina

En el ámbito médico, la IA integrada por medio del ML trata de mejorar los sistemas y equipos médicos que se encargan de los diagnósticos de las enfermedades, con el fin de eliminar errores de los profesionales médicos cuando se analizan los datos de los exámenes, de manera que se puedan reducir los costos en las investigaciones. Los avances en este campo se los puede apreciar en el uso de ciertos programas y *chatbots* para la interacción con los pacientes y de esta forma determinar el probable estado de salud del paciente, o para llevar a cabo algún tipo de seguimiento.

También se ha conocido que el ML ha facilitado el descubrimiento de células cancerígenas en etapa de surgimiento y desarrollo. Gracias a esta cualidad, los tratamientos para los pacientes pueden ser formulados de mejor forma, específicamente orientados de acuerdo a su tipo de cáncer (Hinestroza, 2018).

Debido a la crisis sanitaria mundial a inicios de diciembre de 2019, originada por el SARS-CoV-2 que causa la COVID-19, el entorno de salud pública se declaró en emergencia por el brote del nuevo virus y por su rápida expansión alrededor del mundo. Luego de ser declarada pandemia y declararse la emergencia se hizo necesario acceder “modelos precisos de predicción de brotes para obtener información sobre la probabilidad de propagación y las consecuencias de la reciente enfermedad infecciosa” (Muñoz & Romero, 2021, p. 92). El virus, por su condición de no lineal y complejo, planteó un grave problema en cuanto a la creación de modelos epidemiológicos a gran escala y esto ha motivado para que el ML sea la herramienta que permita desarrollar “modelos de predicción de brotes, pronóstico de casos, estimación de muertes y proyección de recuperaciones” (Muñoz & Romero, 2021, p. 92).

Los modelos de predicción de ML fueron utilizados en años anteriores para el modelamiento y monitoreo de otros brotes epidémicos (Hinestroza, 2018)

como “ébola, cólera, influenza H1N1, fiebre del dengue, zika o norovirus de ostras” (Muñoz & Romero, 2021, p. 92).

2.1.2.2 Educación

La aplicación del ML en la educación se encuentra en sistemas de evaluación de los estudiantes, mediante la creación de planes que contemplen sus necesidades para que el docente pueda analizar cuáles son las deficiencias que tiene cada uno. También se podría llegar a pensar que los sistemas de IA serían capaces de sustituir a los docentes, para eliminar barreras de lugar y que la educación llegue a cualquier parte del mundo, para lo cual el ML ofrece los recursos que faltan, sobre todo la calidad.

El ML podría tener algunos beneficios prácticos en el campo educativo como:

- Permitiría supervisar de manera más fácil y ágil el “progreso educativo de los alumnos prácticamente en tiempo real permitiendo incluso prever futuros itinerarios pedagógicos para ellos a partir de su actividad previa en las plataformas que cuentan con ML” (Aulaplaneta, 2020, párr. 5).
- Facilitaría un alto nivel de autoformación al estudiante que haga uso de esta herramienta, de manera que pueda recibir trabajos y ejercicios frecuentemente, adaptado a sus posibilidades y que permitan mejorar las deficiencias que tenga en el aprendizaje, lo que también hace más efectiva la labor del docente.
- A lo anterior se suma que el ML aportaría a los docentes distintas opciones para que proporcionen a los estudiantes, tareas diversas, con base en anteriores realizadas en clase, lo que agilizaría el proceso de enseñanza (Aulaplaneta, 2020).

2.1.2.3 Construcción

El ML ha permitido la automatización de procesos que anteriormente eran ejecutados por personas, que ocupaban cortos horarios de trabajo y, por consiguiente, poca producción y reducida eficiencia. Estos inconvenientes se mejorarían con el surgimiento del ML combinado con la robótica, herramientas que ayudan a las grandes empresas a mayor producción con mejor rentabilidad (Hinestroza, 2018).

2.1.2.4 Finanzas

En esta área, el ML tuvo su primera aparición, ya que las empresas requerían plataformas intuitivas que permitan a los usuarios realizar revisiones y verificaciones de sus operaciones, transacciones o trámites. A través de softwares que permitían recolectar datos de clientes de instituciones financieras, en la actualidad la aplicación del ML tiene mucha importancia en mercados financieros, como la bolsa de Wall Street, puesto que ésta lleva a cabo múltiples operaciones (Hinestroza, 2018).

2.1.2.5 Tipos

Dentro de las categorías de aprendizaje de ML se encuentran: a) supervisado, b) semisupervisado, c) no supervisado, d) por refuerzo (IBM, 2021).

2.1.2.6 Aprendizaje supervisado

En cuanto al *aprendizaje supervisado*, éste requiere de un conjunto de datos etiquetados a los que se debe comprender cuál es su clasificación, lo que significa que al modelo se le deberá indicar lo que se necesita que aprenda durante su entrenamiento (Gobierno de España, 2020; IBM, 2021). Este tipo de aprendizaje busca conseguir "patrones en datos que se pueden aplicar a un proceso de analítica. Estos datos tienen características etiquetadas que definen el significado de los datos" (IBM, 2021, p. 4).

Según el sitio web Algotive (2022, párr. 23) en el aprendizaje supervisado "la máquina aprende a partir del ejemplo guiado de un humano", lo que significa que la máquina es suministrada de un alto volumen de datos, siendo el operador el que le asigna etiquetas al algoritmo de ML con el fin de realice el reconocimiento de esos datos. En caso de que en algún momento el algoritmo pueda equivocarse, el operador de la máquina lo canaliza nuevamente, de manera que pueda alcanzar un mayor nivel de precisión y rendimiento, para la máquina se optimice al punto de que ésta reconozca nuevos datos y los identifique sin necesidad de la participación del hombre.

Los modelos basados en aprendizaje supervisado tienen un buen funcionamiento en la ejecución de trabajos específicos y su aplicación se la encuentra en la detección de spam, "analíticas de anuncios digitales, reconocimiento de voz, y detección de imágenes" (Algotive, 2022, párr. 25).

Dentro de este tipo de aprendizaje se encuentran dos algoritmos de entrenamiento: a) clasificación, b) regresión. Mediante el **algoritmo de clasificación** se espera que éste indique el grupo al que pertenece el componente que se está estudiando; existen algunos "algoritmos de clasificación para la predicción: Naive Bayes, regresión logística, máquina de vectores de soporte, perceptrón multicapa, árbol de decisión y bosque aleatorio" (Jiménez Alfaro & Díaz Ospina, 2021, p. 115).

El **algoritmo de regresión** espera como resultado un número, que no se sitúa dentro de un grupo, sino que regresa un valor específico (Sandoval, 2018). De acuerdo con Vannieuwenhuyze (2020) algunos de los algoritmos de aprendizaje supervisado por regresión son: a) regresión lineal simple, b) regresión lineal múltiple, c) método de descenso de gradiente, d) regresión polinomial, e) regresión logística, f) árboles de decisión, g) random forest, máquina de vectores de soporte (SVM), entre otros. En la Tabla 5 se describen cuatro de estos algoritmos.

Tabla 5. Descripción de algoritmos de clasificación por regresión

ALGORITMO	DESCRIPCIÓN
Regresión lineal simple	Busca establecer, en la forma de una recta, una relación entre una variable explicada y una variable explicativa. En otras palabras, los datos de una serie de observaciones se representan en la forma de una nube de puntos y se busca encontrar una recta que pase lo más cerca posible de estos puntos.
Método de descenso del gradiente	Consiste en encontrar mediante sucesivas iteraciones el mínimo global de la función de coste (error).
Regresión polinomial	Permite encontrar un vínculo entre las variables con ayuda de una curva añadiendo pliegues a la curva usando elementos llamados polinomios
Árbol de decisión	Ayuda a la decisión o de exploración de datos permite representar un conjunto de opciones en forma de árbol. Se trata de un conjunto de pruebas realizadas con el objetivo de predecir un resultado; las decisiones están en las extremidades de las ramas del árbol.

Nota: Adaptado de Vannieuwenhuyze (2020)

2.1.2.7 Aprendizaje no supervisado

Sobre al aprendizaje no supervisado utiliza datos no etiquetados, puesto que no existe una etiqueta que se deba predecir. Los algoritmos no supervisados tienen su utilidad en trabajos que requieren de un análisis de datos de los que se obtendrá nuevos conocimientos, o también para unir

entidades que son afines; además, se aplican en la reducción de la dimensionalidad o reducir grupos de datos (Gobierno de España, 2020).

2.1.2.8 Aprendizaje semi-supervisado

En referencia al aprendizaje semi-supervisado contiene características de los dos aprendizajes antes mencionados (Algotive, 2022; Gobierno de España, 2020) en donde los algoritmos se los puede alimentar con datos de entrenamiento, pero con la destreza de explorar los datos y los comprenda. Su característica principal es la generación de modelos predictivos cuyo funcionamiento es mejor que los modelos supervisados y sirven para proyectos con un gran volumen de datos no etiquetados (Algotive, 2022).

2.1.2.9 Aprendizaje por refuerzo

El aprendizaje por refuerzo

> Es un modelo de aprendizaje conductual. El algoritmo recibe retroalimentación del análisis de datos, conduciendo al usuario hacia el mejor resultado. El aprendizaje de refuerzo difiere de otros tipos de aprendizaje supervisado, porque el sistema no está entrenado con el conjunto de datos de ejemplo. Más bien, el sistema aprende a través de la prueba y el error. Por lo tanto, una secuencia de decisiones exitosas conduce al fortalecimiento del proceso, porque es el que resuelve el problema de manera más efectiva. (IBM, 2021, párr. 6).

En la Tabla 6 se muestran los cuatro tipos de ML y sus aplicaciones.

Tabla 6. Tipos de ML y sus usos

Tipos de ML	Usos
Aprendizaje supervisado	Detección de spam Reconocimiento de voz Analíticas de anuncios
Aprendizaje no supervisado	Sistemas de recomendación Registro de conductas Detección de anomalías
Aprendizaje semisupervisado	Análisis de voz Detección de fraude Investigaciones médicas
Aprendizaje por refuerzo	Robótica Videojuegos Gestión de recursos

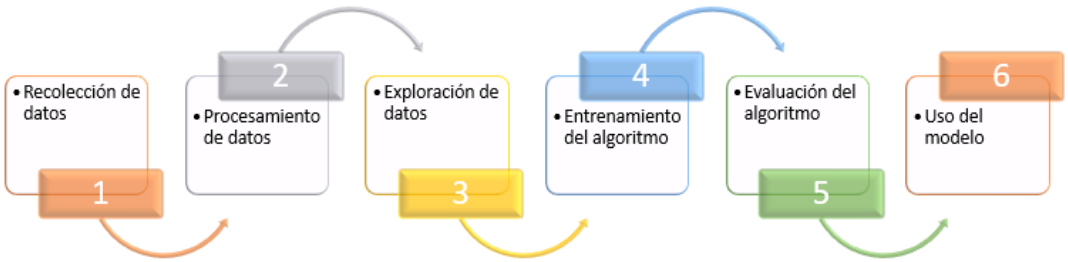
Nota: Tomado de Algotive (2022)

Todos los tipos de aprendizaje antes referidos tienen sus propios algoritmos para la construcción de los modelos ML. La elección del tipo de aprendizaje ML dependerá del tipo de proyecto que se piensa desarrollar, así como el algoritmo que ayudará a generar el modelo ML para la detección de tumores cerebrales mediante un conjunto de imágenes de resonancias magnéticas.

2.1.3 Flujo de trabajo de ML

Según Manrique Rojas (2020) el proceso de creación de un modelo de ML no se restringe solamente al uso de un algoritmo o una biblioteca, sino que sigue seis pasos (ver Figura 2).

Figura 1. Proceso de creación de modelo ML



Nota: Adaptado de Manrique Rojas (2020)

2.1.3.1 Recolección

La recolección se la realiza de distintas fuentes (sitio web, a través de una API o base de datos). Es la fase más difícil del proceso y necesita de un determinado tiempo.

2.1.3.2 Preprocesamiento

Una vez que se dispone de los datos, se deberá confirmar que todos se encuentren con el formato debido. Estos datos servirán para cargarlos al algoritmo de aprendizaje. El preprocesamiento es una fase que requiere ser realizada en varias tareas antes de que los datos puedan ser utilizados.

2.1.3.3 Exploración

Consiste en realizar un estudio anticipado que permita la corrección de “los casos de valores faltantes o tratar de encontrar a primera vista cualquier patrón en ellos que facilite la construcción del modelo. (…) se deben detectar valores atípicos; o (…) las características que tienen más influencia para hacer una predicción” (Manrique Rojas, 2020, p. 589).

2.1.3.4 Entrenamiento

Para el entrenamiento, se requiere que los datos preprocesados alimenten el algoritmo. La finalidad del entrenamiento es que el algoritmo extraiga la información de mayor utilidad de los datos para posteriormente realizar predicciones.

2.1.3.5 Evaluación

Se la realiza mediante “las pruebas de la información que genera el conocimiento del entrenamiento previo que se obtuvo a través del algoritmo” (Manrique Rojas, 2020, p. 589). Se evalúan los resultados de las predicciones del algoritmo y si aún no reflejan lo que se espera, se deberá regresar a la fase anterior.

2.1.3.6 Uso

Se pone en funcionamiento el modelo ML.

2.2 Modelos de ML

Sandoval (2018) mencionó que los algoritmos de ML pueden ser agrupados en los modelos que se muestran en los párrafos siguientes.

2.2.1.1 Modelos lineales

Estos modelos buscan “una línea que se *ajuste* bien a la nube de puntos que se disponen” (Sandoval, 2018, p. 38). Entre estos modelos se encuentran la regresión lineal y la logística, que tienen como inconveniente su ajuste excesivo a los datos que se disponen, arriesgando la integridad de los datos que pudieran llegar. Sus resultados no son muy adecuados para problemas complejos, por su simplicidad.

2.2.1.2 Modelos de árbol

Estos modelos son “modelos precisos, estables y más sencillos de interpretar básicamente porque construyen unas reglas de decisión que se pueden representar como un árbol” (Sandoval, 2018, p. 38). Mediante estos modelos se pueden solucionar problemas puesto que representan relaciones no lineales. Entre estos modelos se encuentran los árboles de decisión random forest. Aunque se pierde rendimiento, se gana capacidad de predicción, puesto que estos modelos tienen mayor precisión y elaboración.

2.2.1.3 Redes neuronales

Los modelos de redes neuronales artificiales son una réplica del funcionamiento del cerebro humano, que es el conjunto de millones de neuronas interconectadas en red a través de las cuales se envían mensajes entre ellas. Este modelo es muy utilizado “por las habilidades cognitivas de razonamiento que adquieren. El reconocimiento de imágenes o vídeos, por ejemplo, es un mecanismo complejo y una red neuronal es lo mejor para realizarlo” (Sandoval, 2018, p. 38). El inconveniente que presentan es que su entrenamiento es lento y requieren alta capacidad de cómputo

2.3 ML aplicado en la medicina

Según González-Islas (2019) el ML se lo ha utilizado en el diagnóstico clínico, la predicción de la eficacia en nuevos tratamientos para enfermedades y en la toma de decisiones que surgen luego de identificar patrones. A pesar de que los métodos de ML se han utilizado en pocas ocasiones en sistemas médicos reales, el interés a esta herramienta se ha incrementado en los últimos años, aplicándose en la mejora de la observación del progreso de una enfermedad, para predecirla, para la autogestión del paciente y en el proceso clínico.

Aunque el modelo de *big data* se encuentra cambiando la ciencia médica, hay que considerar que todos los datos y la información que se obtiene de los pacientes y las investigaciones por sí mismos no poseen una utilidad importante a menos que se los analice, interprete y se puedan tomar decisiones; esto significa que los datos no son los que se aplican a la medicina, sino los algoritmos que ejecutan los procesos; por esto es que se requiere orientar los esfuerzos en desarrollar herramientas de ML para el área médica. Conforme el ML vaya implementándose en la ciencia médica, con el tiempo será el encargado de ejecutar las tareas de los profesionales de radiología y patología, con base en la digitalización de las imágenes ya digitalizadas (González-Islas, 2019).

Los sistemas basados en ML superan la precisión de los radiólogos que leen exámenes médicos, por lo que la seguridad de los pacientes se la confiará con mayor frecuencia a estos sistemas, puesto que éstos tienen la capacidad de monitorear constantemente y en tiempo real la información de

los pacientes, lo que no puede realizar la mano del hombre. A esto se suma información sobre investigaciones realizadas que han demostrado la reincidencia de equivocados diagnósticos de los profesionales médicos, por lo que se requeriría que el ML permita que los diagnósticos de las enfermedades sean más precisos (González-Islas, 2019).

En palabras de Goecks et al. (2020) el ML utiliza algoritmos complejos que funcionan con grupos de datos diferentes a gran escala, de manera que se puedan identificar patrones que tendrían utilidad, cuya identificación podría ser difícil o imposible hasta para expertos. El potencial que tiene el ML en la recopilación y análisis de grandes volúmenes de datos referentes resultados de exámenes y tratamientos médicos aplicados asegura el cambio de la ciencia médica y la orienta hacia una ciencia que ofrezca resultados y se sustente en datos y vasto alcance para detectar, diagnosticar y tratar cualquier padecimiento de salud. Los algoritmos de ML utilizados en el reconocimiento de patrones, realización de predicción, aprendizaje de datos y toma de decisiones incluyen “árboles de decisión, regresión para análisis estadístico y predictivo, redes generativas adversarias, agrupación en clústeres basada en instancias, bayesiana, red neuronal, etc.” (Jurado -Sánchez et al., 2021, p. 17).

Goecks et al. (2020) manifestaron que la recolección de información molecular y fenotípica se ha extendido, incluyendo pruebas de genoma humano para tratar el cáncer, imágenes anatómicas bidimensionales y tridimensionales de altar resolución de órganos, análisis histológicos de biopsias de tejidos y relojes inteligentes que monitorean la frecuencia cardíaca y notifican a los usuarios de irregularidades. Esta información levantada ofrece los insumos para poder realizar diagnósticos tempranos y con mayor precisión, además de tratamientos personalizados y monitoreo continuo.

2.4 Deep Learning

Según Vannieuwenhuyze (2020) “Deep Learning es una rama de ML basada en el uso de neuronas artificiales que se inspiran en el cerebro humano. Estas neuronas están organizadas en capas que le dan una noción de profundidad (…) a la red de neuronas”.

De acuerdo a Janiesch et al., (2021) en la actualidad los sistemas inteligentes que brindan contenidos de IA, por lo general se sustentan en ML, el mismo que se refiere a la forma que tienen los sistemas para realizar un aprendizaje de los datos específicos de entrenamiento, que provienen de problema a resolver, para la automatización del proceso de creación de modelos de análisis y solución de los trabajos a ella asociadas.

Para Sáez de la Pascua (2019) el Deep Leaning o aprendizaje profundo es un concepto derivado del ML, que se sustenta en redes neuronales artificiales (RNA). Se dice profundo debido a la cantidad de capas de dichas técnicas; las redes neuronales de Deep Learning tienen algunas capas, frente a la capa de las convencionales, que se encarga de hacer una simulación la forma de aprender de los seres humanos cuando se busca alcanzar algún grado de conocimiento.

Se puede considerar al Deep Learning una manera de llevar a cabo la automatización del análisis predictivo mediante algoritmos complejos y de extracción, los cuales aplican transformación no lineal de entrada, y crea un modelo estadístico de salida con los datos del aprendizaje; las iteraciones del aprendizaje se mantienen hasta que se ha conseguido una precisión admisible en la salida.

Con Deep Learning el aprendizaje del modelo es realizado por sí mismo, y de esta forma podrá explorar las diversas relaciones que existen entre las distintas variables, sin que intervenga la mano humana, ya que el modelo será capaz de seleccionar las características (Sáez de la Pascua, 2019).

2.5 Redes Neuronales Artificiales (RNA)

Según lo manifestado por Quiñones Huatangar et al., (2020, p. 110) “la RNA es un modelo computacional capaz de imitar las características básicas del cerebro humano como la autoadaptabilidad, autoorganización y la tolerancia a errores”. Las aplicaciones de las RNA han aumentado en el transcurso del tiempo en la mayoría de las áreas del conocimiento, puesto que son modelos que sirven para la solución de problemas relacionados con la estadística convencional, por lo que los modelos que surgen de las RNA sirven para la descripción de “pronósticos para tiempos más prolongados, además de relaciones no lineales, por ende, es difícil prescribir la relación matemática

exacta entre los parámetros" (Quiñones Huatangar et al., 2020, p. 110). Su adopción se orienta para identificar, analizar, pronosticar y reconocer sistemas y optimizar los modelos.

Las Redes Neuronales Artificiales (RNA) son parte de la IA, además de ser un conjunto de técnicas para procesar información, que se inspiran en el sistema nervioso de un ser humano y tratan de imitar el funcionamiento del cerebro, con el fin de solucionar "funciones altamente no lineales en un tiempo corto porque aprenden de los datos que son difíciles de expresarlas matemáticamente" (Rivas-Asanza et al., 2018, p. 12). Se las considera como herramientas eficaces para analizar señales y modelar y controlar sistemas complejos por

> Su capacidad para aproximar funciones f: $\mathbb{R}^N \rightarrow \mathbb{R}^M$. De forma general están formadas por una capa de entrada, un conjunto de capas intermedias ocultas, y una capa de salida. En cada una de las capas hay un conjunto de neuronas. Las salidas de las neuronas de cada capa se encuentran conectadas a las entradas de la capa siguiente. El enlace entre cada dos neuronas tiene asociado un coeficiente denominado peso. Cada neurona implementa una función matemática f: $\mathbb{R}^N \rightarrow \mathbb{R}$, donde N es el número de neuronas de la capa anterior. Además, cada neurona puede tener coeficientes que modifica el algoritmo de aprendizaje (…). Cada capa puede tener un número de neuronas diferente. Habitualmente todas las neuronas de la misma capa son del mismo tipo. El algoritmo de aprendizaje ajusta los pesos de la red y los coeficientes de las neuronas para que la red implemente la función esperada (Sierra-García & Santos, 2021, p. 328).

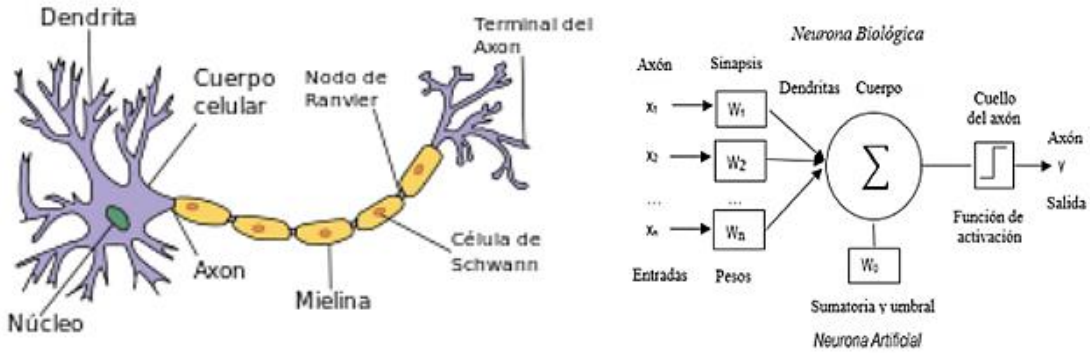
Entre las ventajas que tienen las RNA se encuentran:

> Su capacidad de generalización, que permite dar una respuesta adecuada a entradas nunca vistas anteriormente; su naturaleza distribuida, que permite la construcción de sistemas eficientes; su capacidad de aproximar funciones no lineales, que resuelven problemas no complejos y su adaptabilidad frente a cambios en el entorno. (Campos Wright & Trujillo Casañola, 2021, p. 185).

El entrenamiento de una RNA “consiste en iterativamente suministrar a la red una secuencia de patrones de entrada, y ajustar los pesos de las conexiones en función de las salidas obtenidas” (Campos Wright & Trujillo Casañola, 2021, p. 185). El proceso iterativo refina la solución hasta que su grado de operación sea lo bastante bueno. “La mayoría de los métodos de entrenamiento utilizados en las redes neuronales con conexión hacia delante consisten en proponer una función de error que mida el rendimiento actual de la red en función de los pesos” (Quiñones Huatangar et al., 2020, p. 110).

El entrenamiento de la RNA sirve para descubrir el grupo de pesos para maximizar o minimizar la función. El método de entrenamiento de optimización para la RNA ofrece una medida para actualizar los pesos, de manera que se llegue al estado óptimo (ver Figura 2).

Figura 2. Similitudes entre la red neuronal biológica y RNA.



Nota: Adaptado de Areli-Toral Barrera (2015) y Quiñones Huatangar et al., (2020)

En la Tabla 7 se muestra una comparación entre una neurona biológica y una artificial.

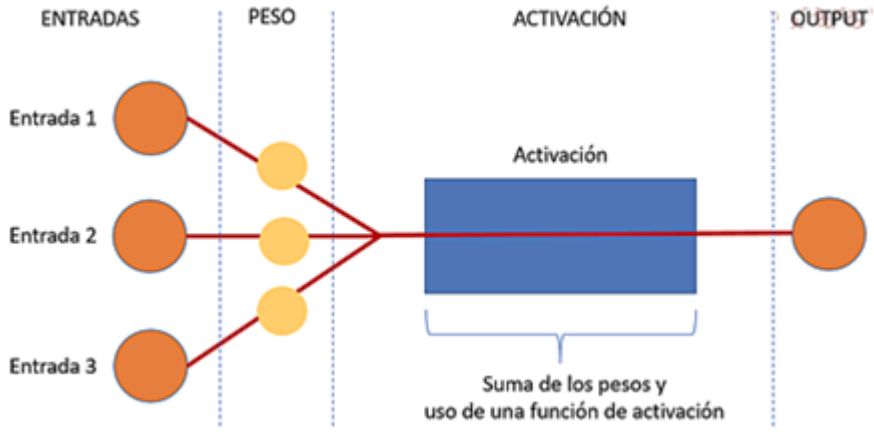
Tabla 7. Comparativa de una neurona biológica y una artificial

NEURONA BIOLÓGICA	NEURONA ARTIFICIAL
Dendritas	Entradas (input)
Sinapsis	Peso
Axón	Salida (output)
Activación	Función de activación

Nota: Tomado de Vannieuwenhuyze (2020)

2.6 Arquitectura de una RNA

Figura 3. Arquitectura de la RNA



Nota: Tomado de Vannieuwenhuyze (2020)

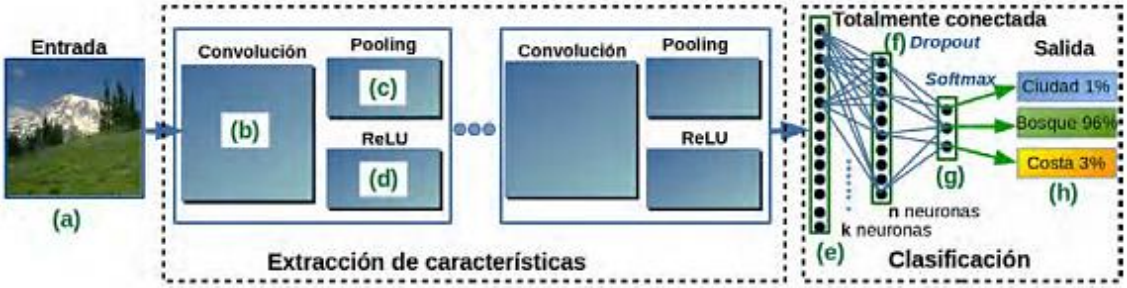
2.7 Elementos de una RNA

De acuerdo con Varela Arregocés & Campbells Sánchez (2011) las partes más importantes para que una RNA funcione son:

- Un conjunto de unidades de procesamiento (neuronas).
- Un estado de activación (variable de estado).
- Una función de salida para cada unidad.
- Un conjunto de conexiones (patrón de conectividad).
- Un conjunto de reglas para propagar las señales de salida a través de la RNA.
- Una regla de combinación.
- Una regla de activación.
- Una regla de modificación.
- Un ambiente en el cual opera la RNA. (Varela Arregocés & Campbells Sánchez, 2011, pp. 21–22).

De acuerdo a López Saca (2019), el proceso que sigue una CNN se visualiza en la Figura 4.

Figura 4. Proceso de la CNN



Nota: Tomado de López Saca (2019)

Una especificación más concisa de las partes o elementos de una RNA la dio Aguado López (2020, pp. 9–10):

- Un *conjunto de entradas* x_j.
- Conjunto de pesos sinápticos w_{ij}, con $j = 1,\ldots n$. Representa la interacción entre la neurona presináptica *j* y la postsináptica *i*.
- Una *regla de propagación* h_i, definida a partir del conjunto de entradas y los pesos sinápticos. Es decir:

$$h_i(X_1, \ldots, X_n, W_{i1}, \ldots, W_{in})$$

La regla de propagación que normalmente se usa es la combinación lineal de las entradas y los pesos, obteniéndose:

$$h_i(X_1, \ldots, X_n, W_{i1}, \ldots, W_{in}) = \sum_{i=1}^{n} W_{ij} X_j$$

También es posible adicionar otro parámetro θ (umbral), que se resta al potencial postsináptico.

$$h_i(X_1, \ldots, X_n, W_{i1}, \ldots, W_{in}) = \sum_{i=1}^{n} W_{ij} X_j - \theta$$

- Una *función de activación*, la cual representa simultáneamente la salida de la neurona y su estado de activación. Normalmente los valores de la salida están comprendidos en un rango (0,1) o (-1,1). Si se denota por y_i dicha función de activación, se obtiene:

$$y_i = (f_i) = f_i(\sum_{j=0}^{n} W_{ij} X_j - \theta)$$

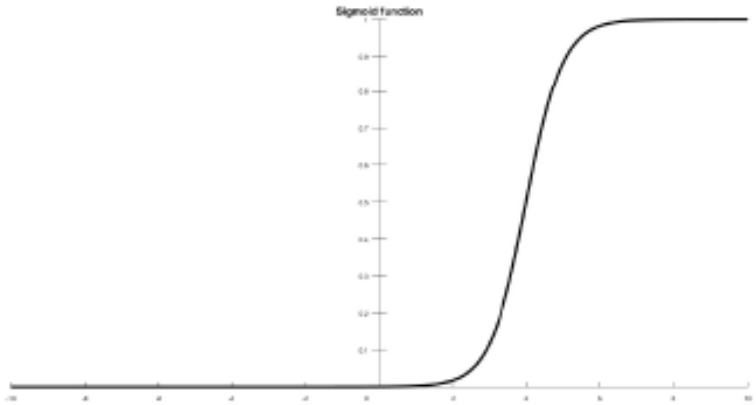
De todas las funciones de activación que existen siempre se intentará buscar aquella cuya derivada sea simple ya que esto hará que se minimice el coste computacional. Las más utilizadas son:

Sigmoid-Sigmoide

Transforma los valores que se introduzcan a una escala (0,1), donde aquellos valores altos tienden de manera asintótica al valor 1 mientras que los valores muy bajos lo hacen al valor 0.

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

Figura 5. Función Sigmoidal



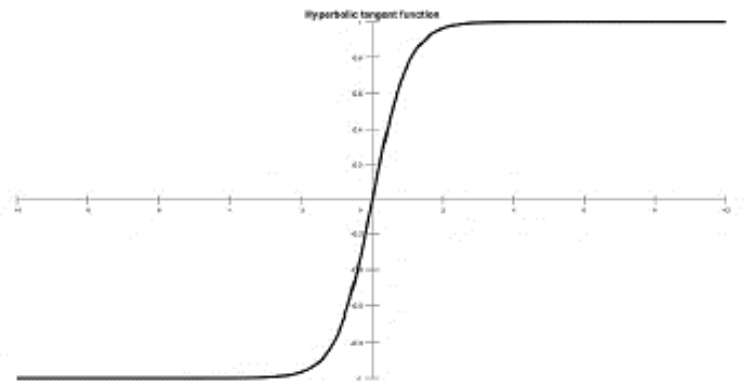
Nota: Tomado de Barrera (2018)

Tanh: Tangente hiperbólica

Realiza la misma transformación anterior, pero a una escala (-1,1).

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

Figura 6. Función de activación tangente hiperbólica



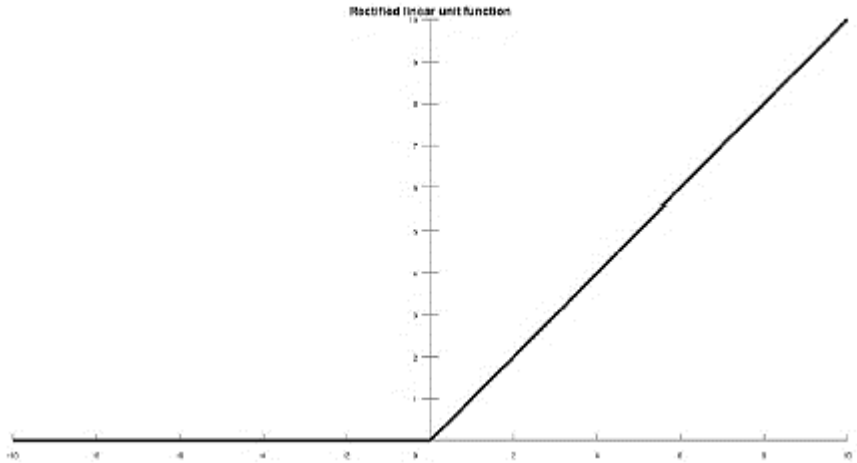
Nota: Tomado de Barrera (2018)

ReLU: Rectified Lineal Unit

Iguala a 0 los valores negativos y deja los positivos tal y como están.

$$f(x) = max(0, x) = \begin{cases} 0 \ \ si\ x\ < 0 \\ x \ \ si\ x >= 0 \end{cases}$$

Figura 7. ReLU



Nota: Tomado de Barrera (2018)

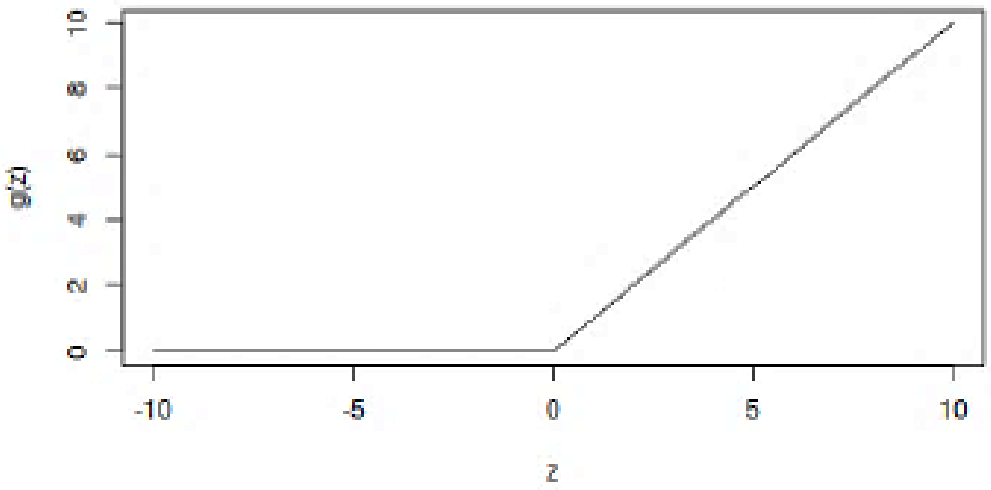
Leaky Rectified Lineal Unit

A diferencia de la función ReLU simple, esta multiplica los valores negativos por un coeficiente rectificativo (Aguado López, 2020).

$$f(z)_j = \begin{cases} a * x \ \ si\ x < 0 \\ x \ \ si\ x >= 0 \end{cases}$$

En la Figura 8 se muestra el valor que toma la ReLU de acuerdo al valor de z_i (Charte Luque, 2017).

Figura 8. Función ReLU



Nota: Tomado de Charte Luque (2017)

Softmax

Transforma las salidas a una representación en forma de probabilidades, de modo que el sumatorio de todas las probabilidades de las salidas dé 1 (Aguado López, 2020, pp. 9–10).

$$f(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{z}}$$

2.8 Topología de RNA

Según García Sánchez (2019) las RNA tienen su clasificación de acuerdo a su topología, que se define de acuerdo a "sus funciones de activación, la función de propagación, su regla de aprendizaje y su número de capas" (García Sánchez, 2019, p. 8). Sus conexiones definen dos tipos de RNA:

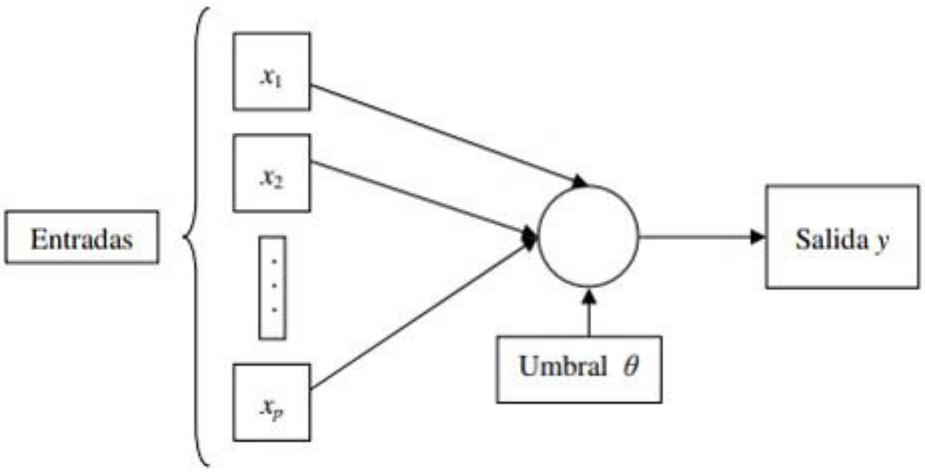
2.8.1 Redes Feed-forward propagation (forward)

Son estrictamente hacia delante (García Sánchez, 2019). A este grupo pertenece el perceptrón simple, perceptrón multicapa, Adaline, entre otros (Cebrián Aldana, 2020).

2.8.2 Perceptrón simple

Dicho de forma sencilla, el perceptrón es una única neurona, que tiene pesos (sinápticos) y umbral ajustables (Aguado López, 2020; García Sánchez, 2019). En la Figura 9 se puede apreciar la estructura de un perceptrón de una sola neurona.

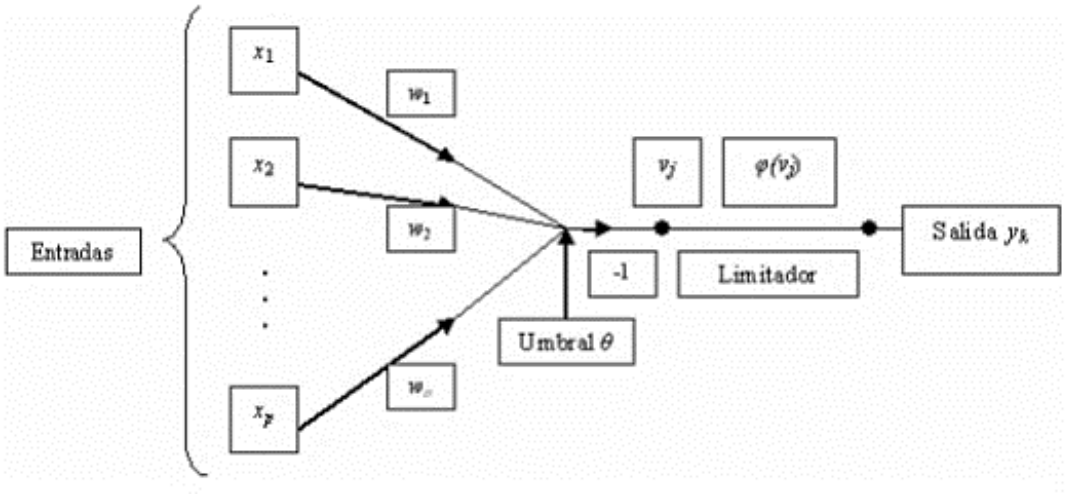
Figura 9. Representación del perceptrón de una sola neurona



Nota: Tomado de Aguado López (2020)

El perceptrón simple es una red neuronal simple, la más antigua y básica, que es utilizada por su eficacia en la solución de problemas de clasificación de patrones linealmente separables; esto significa que los patrones están situados en los dos lados de un hiperplano (Aguado López, 2020; Cebrián Aldana, 2020; García Sánchez, 2019). En la Figura 10 se muestra la estructura de un perceptrón simple.

Figura 10. Representación de un perceptrón simple



Nota: Tomado de Aguado López (2020)

2.8.3 Perceptrón multicapa

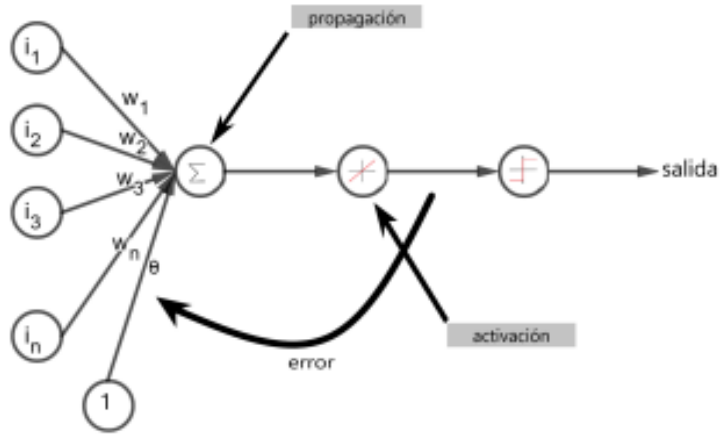
Es una variación del perceptrón simple, pero posee “una o más capas ocultas entre sus capas de entrada y salida. Las conexiones entre neuronas siempre se dirigen desde las capas inferiores a las superiores, y las neuronas en la misma capa no están interconectadas” (Cebrián Aldana, 2020, p. 8). La cantidad de neuronas que tiene la capa de entrada tiene la misma cantidad de atributos de entrada; lo mismo sucede con la cantidad de neuronas que tiene la capa de salida, que es la misma cantidad de número de clases. El aprendizaje consiste en adaptar “los pesos de las conexiones para obtener una diferencia mínima entre la salida obtenida y la esperada, y para esto, la técnica más utilizada es la retropropagación” (Cebrián Aldana, 2020, p. 8).

2.8.4 Adaptative Linear Neuron (ADALINE)

Red de tipo forward, que tiene su similitud al perceptrón simple, que se diferencia de éste por la regla de aprendizaje utilizado. ADALINE usa “como salida el número real obtenido por la red, por lo que se tiene en cuenta cuánto

se ha equivocado, se va a utilizar la diferencia entre el valor real esperado y el producido por la red" (Cebrián Aldana, 2020, p. 9). En la Figura 11 se aprecia la estructura de ADALINE.

Figura 11. Estructura de ADALINE

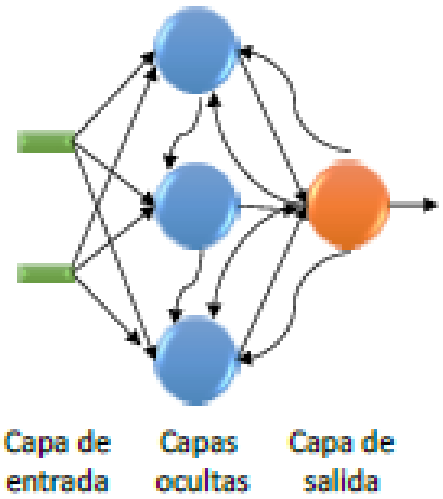


Nota: Tomado de García Sánchez (2019)

2.8.5 Redes recurrentes (Feed back-forward propagation o back-forward)

Las conexiones pueden ser entre las redes que tienen una misma capa; esto incluye "la misma neurona y hacia capas anteriores. En este tipo de redes se pueden alcanzar estados de estabilidad para ciertas neuronas, donde la activación de esta neurona no cambiará" (García Sánchez, 2019, p. 7). En la Figura 12 se muestra la representación de una red recurrente.

Figura 12. Red recurrente



Nota: Adaptado de Manjarrez (2014)

2.9 Teoría de las Redes Neuronales Convolucionales (CNN)

Las CNN se han constituido en el modelo de RN con mayor representatividad en el aprendizaje profundo. La visión artificial sustentada en CNN ha facilitado el reconocimiento facial, vehículos autónomos, autoservicio en supermercados y los tratamientos médicos inteligentes (Li et al., 2021).

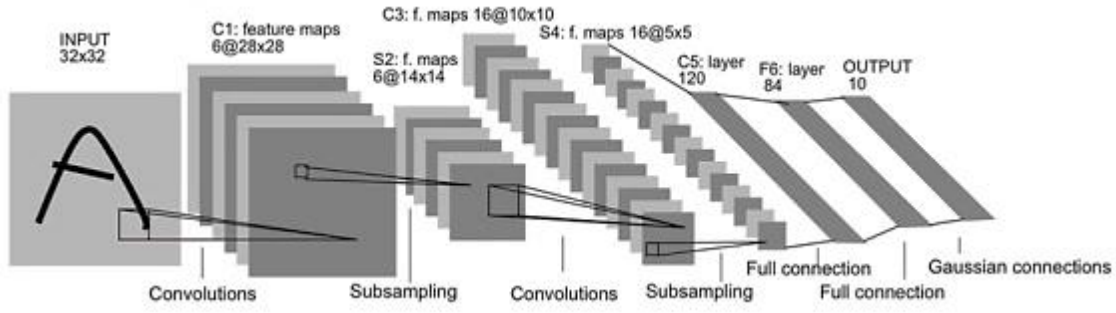
La CNN, en palabras de Cifuentes et al., (2019, p. 5) “es un tipo de RNA con aprendizaje supervisado que procesa sus capas imitando el cortex visual del ojo humano para identificar distintas características en la entrada que definitiva hacen que puedan identificar objetos”. Las CNN realizan: “detección y/o categorización de objetos, clasificación de escenas, clasificación de imágenes en general” (Cifuentes et al., 2019, p. 5).

De acuerdo a Rodríguez (2020) el proceso que sigue esta red tiene dos fases: 1) extracción de las características más importantes que parten de datos, y 2) con base en la extracción de las características y en el problema que se estudia, retorna un resultado lo más cercano posible a lo que se espera como salida para la solución del problema. Se puede trabajar como clasificación de imágenes, que devuelve “un vector de probabilidades que representará la probabilidad de pertenencia a cada una de las posibles clases predichas” (I. Rodríguez, 2020, p. 7). También esta arquitectura puede realizar la “segmentación semántica de una imagen, donde habitualmente se devuelve la probabilidad de pertenencia a cada clase para cada pixel de la imagen de entrada” (I. Rodríguez, 2020, p. 7).

Las CNN reciben este nombre debido a su función de extracción de características, la que se puede describir como varias capas, cada una de las cuales emplean diferentes transformaciones a los datos de entrada de manera secuencial, es decir, “la capa *i*-ésima opera sobre la salida generada por la capa *i* – 1-ésima” (I. Rodríguez, 2020, p. 7).

En la Figura 13 se muestra el esquema genérico de una CNN.

Figura 13. Esquema de una CNN



Nota: Tomado de Erroz Arroyo (2019)

2.9.1 Capas de convolución

Las capas convolucionales son filtros conocidos como campos receptivos adaptables para poder extraer las características de una señal. Difiere de una RNA

> Cada neurona de una capa se conecta a todas las neuronas de la otra, (…), en las CNN se comparten las neuronas a través de filtros que permiten extraer información de las imágenes de entrada. Cada capa de la CNN es un bloque con tres principales variables: entrada, pesos y salida. en estas capas se presenta una característica fundamental como donde la salida de una capa se convierte en la entrada de la próxima. Este proceso es secuencial y puede ser no lineal, en cada capa se realiza una función específica. (Cifuentes et al., 2019, p. 5).

El más importante trabajo de una CNN es la extracción de patrones (Cifuentes et al., 2019). De acuerdo a Rodríguez (2020), en las aplicaciones que procesan imágenes, la convolución de los llamados *filtros* que se encuentran sobre la imagen, representa una herramienta común que se utiliza en inconvenientes de “segmentación, detección de bordes o reducción de imagen. No obstante, los valores de dichos filtros deben ser establecidos a priori, lo que requiere experiencia en el campo y a menudo largos procesos de ajuste de parámetros al problema en cuestión” (I. Rodríguez, 2020, p. 7). En contraste a lo anterior, las redes de convolución inician a partir de filtros eventuales que se adaptan automáticamente cuando la red está en aprendizaje. Las capas convolucionales se componen de N filtros W de dimensión $k * k * c$.

> Dada una entrada *X* de dimensión *h* × *w* × *c*, cada uno de los filtros de convolución se coloca sobre cada ventana de tamaño *k* × *k* × *c* centrada en cada píxel $x_{m,n}$ de *X*. El filtro de convolución se va desplazando a lo largo de toda la imagen, tras aplicar la convolución a cada una de las ventanas de la imagen. (I. Rodríguez, 2020, p. 7).

La salida $Y_{m,n}$ que se genera para la ventana centrada en el pixel $x_{m,n}$ es calculada de acuerdo a la fórmula señalada a continuación:

$$Y_{m,n} = \sum_{d=1}^{c}\sum_{j=1}^{k}\sum_{i=1}^{k} X_{m+1-\frac{k-1}{2}-1,n+j-\frac{k-1}{2}-1,d} \cdot W_{i,j,d}$$

2.9.2 Pooling

Esta capa reduce "la salida de la capa de convolución, de modo que la cantidad de características también sean reducidas, manteniendo la información más representativa de la imagen" (López Saca, 2019, p. 28). De esta manera disminuye el valor de las propiedades y las convierte en robustos a la distorsión y el ruido. "El parámetro usado es una máscara *M* de tamaño *k* × *k* y paso ρ" (López Saca, 2019, p. 28). Por lo tanto, el objetivo primordial de esta capa es la reducción progresiva del tamaño espacial de la imagen de entrada para así reducir el número de cálculos en la red (Lakkavaram et al., 2019).

Entre los tipos de pooling de mayor utilización se encuentran:

2.9.2.1 Max-Pooling

De acuerdo con López Saca (2019, p. 28) "obtiene el valor máximo de una región R_i de tamaño *k* × *k* de una imagen *I*(*x, y*) de tamaño *N x N*". Se la define como:

$$Y_2 = máx\,\{R_i\}\; i \in [k * k]$$

En donde

$R_i \in I\,(x, y)\; \forall\, \{x, y\} \in \{1 \ldots N\}$

2.9.2.2 Average-Pooling

Según López Saca (2019, p. 28) "obtiene el valor promedio de los elementos j de una región Ri de una imagen I(x, y) de tamaño N × N". Se la define de la siguiente forma:

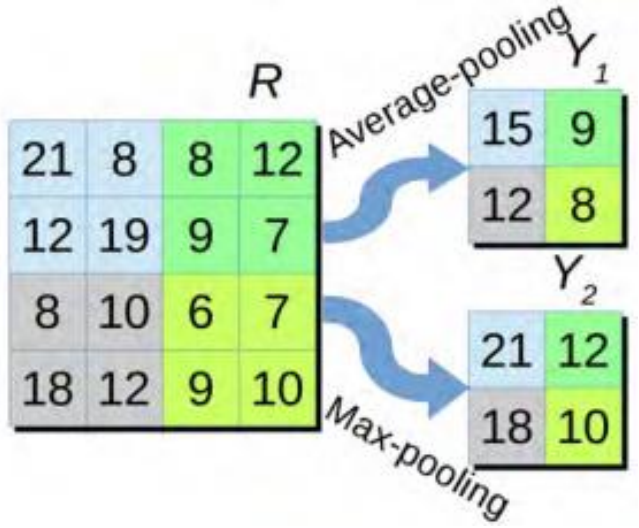
$$Y_1 = \frac{1}{k\ x\ k} \sum_{i \in [k\ x\ k]} R_i$$

En donde

$R\_i \ \in I\ (x, y)\ \forall \{x, y\}\ \in \{1 \ldots N\}$

En la Figura 14 se muestra un ejemplo de Max-Pooling y Average-Pooling

Figura 14. Ejemplo de max-pooling y average-pooling en una matriz R



Nota: Tomado de López Saca (2019)

2.9.3 Entrenamiento de una CNN

En las CNN las neuronas de las capas que las forman, se unen solamente con algunas de las neuronas de la capa subsiguiente, lo que quiere decir que se facilita el aprendizaje y se disminuyen los costos informáticos (Del Hoyo Dorado, 2020).

Los algoritmos para detectar objetos que realizan las CNN tienen dos fases: a) extracción de características de la imagen, y b) búsqueda de objetos con base en dichas características para poder clasificarlas; de este modo se facilita el aprendizaje de la red. A las CNN se las puede entrenar partiendo de cero, aunque este proceso es bastante difícil, puesto que es difícil conseguir grupos de datos que sean grandes y que puedan servir para conseguir adecuada precisión en la predicción, ya que las RN sufren presentan sobreajuste, además de implicar un elevado costo informático y tiempo de entrenamiento extenso (Del Hoyo Dorado, 2020; Quintero et al., 2018).

El problema antes mencionado requiere de la aplicación de la técnica conocida como ***transferencia*** de aprendizaje o de conocimiento, que es un proceso que se ajusta a modelos entrenado previamente (Del Hoyo Dorado, 2020; Quintero et al., 2018), en el que se empieza "con una red previamente entrenada y se le proporcionan datos nuevos con clases desconocidas para la red. Gracias a este método el tiempo de entrenamiento se ve reducido notablemente" (Del Hoyo Dorado, 2020, p. 23). Las características de bajo nivel, como los bordes, se obtienen de las RN de las primeras capas; en las capas siguientes las capturas son de más alto nivel. Con el uso de modelos ya entrenados, se pueden aprovechar las características de bajo nivel, pudiendo resolverse el inconveniente del sobreajuste; también se disminuye la carga de entrenamiento (Quintero et al., 2018).

2.9.3.1 AlexNet

Esta arquitectura se compone de "cinco capas de convolución y tres capas de conexión completa" (Izquierdo-Valladarez & Cuenca-Tapia, 2019, p. 7). También se conoce que otra particularidad de esta arquitectura "es que la imagen de entrada tiene que tener un tamaño de 227x227. Las dos primeas capas de convolución son seguidas de capas overlapping max pooling, las cuales son usadas para reducir el ancho y alto, pero mantienen la profundidad" (Izquierdo-Valladarez & Cuenca-Tapia, 2019, p. 7). Estas capas tenían 60 millones de parámetros y 650 mil neuronas (Moreno Díaz-Alejo, 2020).

Posterior a la convolución, se presenta la función *ReLu,* la misma que efectúa una rectificación lineal. Además

> De la tercera capa a la quinta están conectadas directamente. Luego de la quinta capa, están conectada a un overlapping max layer, las cuales van a dos series de fully connected layers. Finalmente, estas dos están conectadas a softmax classifier con 1000 clases de etiquetas. (Izquierdo-Valladarez & Cuenca-Tapia, 2019, pp. 7–8).

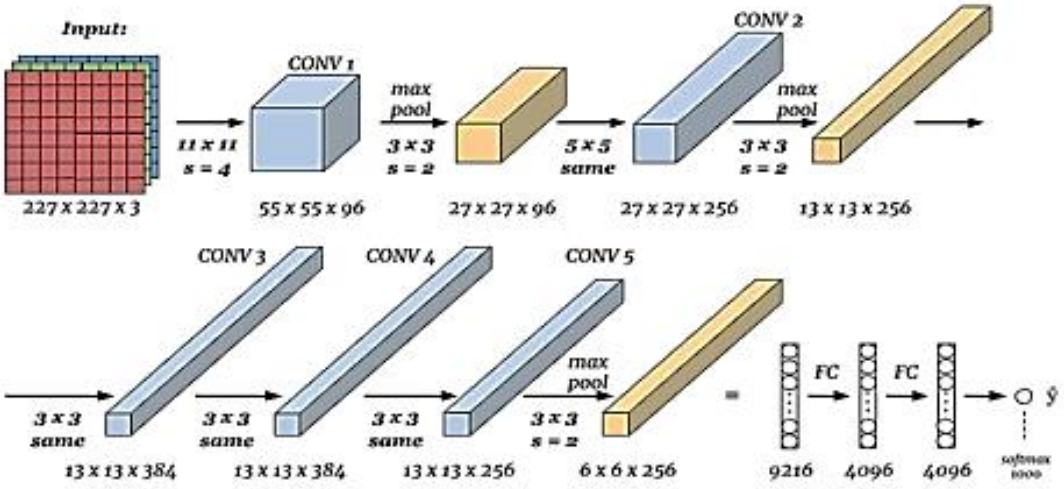
El modelo AlexNet redujo la reducción de la tasa de error, que estaba por sobre el 25%, al 17%. Su arquitectura tiene mayor complejidad y es más profunda que las redes utilizadas hasta 2012 (Moreno Díaz-Alejo, 2020).

AlexNet fue un punto de referencia para las tareas de visión artificial. Desde ahí "para las tareas de clasificación de imágenes se fueron construyendo

modelos de CNN con distintas arquitecturas, cada vez más complejos y más profundos y mejorando cada vez más los resultados" (Moreno Díaz-Alejo, 2020, p. 19). Aunque tiene su antigüedad, aún se la continúa utilizando, puesto que su tiempo de entrenamiento es bastante corto.

En la Figura 15 se muestra la arquitectura de AlexNet.

Figura 15. AlexNet



Nota: Tomado de Erroz Arroyo (2019)

2.9.3.2 Red ZF

Esta arquitectura implementa otros avances importantes, como lo es la reducción de la dimensión "introduce mejoras significativas reduciendo el tamaño del filtro de la primera capa convolucional de 11x11 a 7x7, con pasos de 2 pixeles, en lugar de 4. (…) retiene mucha más información en las características de la primera y segunda capa" (Lozada-Portilla et al., 2021, p. 4). Su tasa de error fue del 16%.

2.9.3.3 VGGNet

Esta arquitectura implementa una nueva funcionalidad al diseño se la CNN, para aumentar permanentemente la profundidad de la red y agrega mayor cantidad de capas convolucionales. Esto es factible "debido al uso de filtros 3x3 en todas las capas, logrando un mejor resultado en la métrica de precisión en tareas de clasificación de imágenes, en dos de sus modelos: VGG16 Y VGG19" (Lozada-Portilla et al., 2021, p. 4), siendo la VGG16 una de las más usadas para extraer características en imágenes (Erroz Arroyo, 2019). Este tipo de arquitectura confirma que "la profundidad del mapa de características

es beneficiosa para la precisión de la clasificación" (Lozada-Portilla et al., 2021, p. 4).

En la Figura 16 se muestra la arquitectura VGG-16.

Figura 16. VGG-16

Nota: Tomado de Erroz Arroyo (2019)

2.9.3.4 ResNet

Es una arquitectura novedosa, surgida con el objetivo de solucionar el problema presente entre "la necesidad de modelos muy profundos capaces de resolver tareas más complicadas y también a la vez incrementar o mejorar la precisión de la clasificación" (Erroz Arroyo, 2019, p. 18), puesto que era un difícil inconveniente debido a que si el modelo es más profundo, es más complicado el aprendizaje.

En la Figura 17 se muestra la arquitectura ResNet.

Figura 17. ResNet

Nota: Tomado de Erroz Arroyo (2019)

2.10 Entornos de trabajo para CNN

Actualmente se pueden encontrar entornos de desarrollo para las CNN como, por ejemplo:

- Caffe: desarrollado por la Universidad de Berkeley.
- Caffe2: Sucesor de Caffe y soportado por Facebook.
- Torch: Soportado por ingenieros de Facebook, Twitter y Google. Desarrollado en lenguaje *Lua*.
- PyTorch: Una versión de Torch para Python soportada por Facebook.
- CNTK: Soportado por Microsoft.

- DSSTNE: Soportado por Amazon. (Picazo Montoya, 2018, p. 6).
- TensorFlow: Soportado por Google. Ejecuta rápida y eficientemente gráficos de flujo que se componen de operaciones matemáticas representadas sobre nudos y cuyas entradas y salidas son vectores multidimensionales de datos (tensores) (Erroz Arroyo, 2019, p. 24; Picazo Montoya, 2018).

De acuerdo con Erroz Arroyo (2019) otra de las herramientas para ML es *Google Colab*, que es un framework gratuito de Google, sin necesidad de configuración y su ejecución es directamente en la nube. Con *Colab* es posible importar "un conjunto de datos de imágenes, entrenar un clasificador de imágenes con dicho conjunto de datos y evaluar el modelo con tan solo usar unas pocas líneas de código" (Colaboratory, s/f, párr. 10), por lo que es un entorno interactivo al poder combinar código con imágenes.

Una de las ventajas de utilizar *Colab* es que se puede "ejecutar código en GPU potentes que ofrece Google" (Erroz Arroyo, 2019, p. 24). Algunas de las aplicaciones de *Colab* son:

- Iniciarse con *TensorFlow*.
- Desarrollo y entrenamiento en RN.
- Probar con TPUs.
- Publicar investigaciones de IA.
- Creación de tutoriales (Colaboratory, s/f).

2.11 Metodologías de desarrollo de software

Los procesos de desarrollo de software en dos: a) *procesos dirigidos por un plan*, que "son aquellos son aquellos donde todas las actividades del proceso se planean por anticipado y el avance se mide contra dicho plan" (Sommerville, 2011, p. 29), y b) *procesos ágiles*, se plantean de forma incremental, de manera que cualquier modificación se la puede realizar más fácilmente para que los cambiantes requisitos del usuario final se reflejen en el producto de software. Cada uno de estos enfoques se adecúa a distintos tipos de proyectos.

Las metodologías de desarrollo de software son aquellos métodos y técnicas utilizadas para el diseño de un producto de software. Son importantes por cuestiones de organización del trabajo, puesto que toda la planificación

deberá tener un orden y conocer su funcionamiento, además de controlar el trabajo, con el fin de anticiparse a posibles errores y poderlos corregir (Universitat Carlemany, 2020).

De acuerdo con Reynaldo Molina (2017) aplican procesos disciplinados al desarrollo de software para que sea más predecible y eficiente, “definiendo una representación que permite facilitar la manipulación de modelos y el intercambio de información entre todas las partes involucradas en la construcción de un sistema” (Reynaldo Molina, 2017, p. 14).

Las metodologías de desarrollo de software se clasifican en tradicionales y ágiles.

Figura 18. Metodologías de desarrollo de software

Nota: Tomado de Santander Universidades (2020)

2.11.1 Metodologías Tradicionales

Se orientan al control de los procesos, definiendo de forma total y rígida todos los requerimientos, actividades, herramientas y notaciones al inicio de cualquier proyecto. No se permite realizar cambios en los ciclos de desarrollo, con una organización lineal, lo que significa que cada una de las etapas del desarrollo siguen una tras otra, debiendo terminar una para continuar a la siguiente, sin poder regresar a la anterior cuando se ha avanzado a la otra etapa. No tienen buena adaptación a los cambios (Reynaldo Molina, 2017; Santander Universidades, 2020).

En la Tabla 8 se presentan algunas de las metodologías tradicionales.

Tabla 8. Tradicionales

Metodología	Definición
Waterfall (cascada)	Ciclo de vida clásico, sugiere un enfoque sistemático y secuencial6 para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado
Prototipado	Se basa en la construcción de un prototipo de software que se construye rápidamente para que los usuarios puedan probarlo y aportar feedback. Así, se puede arreglar lo que está mal e incluir otros requerimientos que puedan surgir. Es un modelo iterativo que se basa en el método de prueba y error para comprender las especificidades del producto.
Espiral	El proceso de software se representa como una espiral, y no como una secuencia de actividades con cierto retroceso de una actividad a otra. Cada ciclo en la espiral representa una fase del proceso de software. El ciclo más interno puede relacionarse con la factibilidad del sistema, el siguiente ciclo con la definición de requerimientos, el ciclo que sigue con el diseño del sistema, etcétera.
Incremental	Se va construyendo el producto final progresivamente. En cada etapa incremental se agrega una nueva funcionalidad, lo que permite ver resultados de una forma más rápida. El software se puede empezar a utilizar incluso antes de que se complete totalmente y, en general, es mucho más flexible que las demás metodologías.
Diseño rápido de aplicaciones (RAD)	Metodología orientada a objetos para el desarrollo de sistemas, la cual incluye un método de desarrollo junto con herramientas de software.

Nota: Adaptado de Santander Universidades (2020), Pressman (2013) y Sommerville (2011)

2.11.2 Metodologías Ágiles

Se fundamentan en la metodología incremental, esto significa que, en el desarrollo de cada una de las etapas del desarrollo que son más rápidas y cortas, se agregan nuevas y pequeñas funcionalidades al producto final en lugar de cambios grandes (Santander Universidades, 2020). A más de fundamentarse en el desarrollo incremental, se orientan hacia la interacción con el cliente, a quienes presentan entregas parciales en determinados intervalos de tiempo, para su evaluación y retroalimentación (Reynaldo Molina, 2017).

En la Tabla 9 se muestran algunas de las metodologías ágiles.

Tabla 9. Metodologías Ágiles

Metodología	Definición
Kanban	Consiste en dividir las tareas en porciones mínimas y organizarlas en un tablero de trabajo dividido en tareas pendientes, en curso y finalizadas. De esta forma, se crea un flujo de trabajo muy visual basado en tareas prioritarias e incrementando el valor del producto.
Scrum	Metodología incremental que divide los requisitos y tareas de forma similar a Kanban. Se itera sobre bloques de tiempos cortos y fijos (entre dos y cuatro semanas) para conseguir un resultado completo en cada iteración. Las etapas son: planificación de la iteración, ejecución, reunión diaria y demostración de resultados. Cada iteración por estas etapas se denomina también sprint.
Lean	Está configurado para que pequeños equipos de desarrollo muy capacitados elaboren cualquier tarea en poco tiempo. Los activos más importantes son las personas y su compromiso, relegando así a un segundo plano el tiempo y los costes. El aprendizaje, las reacciones rápidas y potenciar el equipo son fundamentales.
Programación extrema (XP):	Integra un rango de buenas prácticas de programación, como las liberaciones frecuentes del software, el mejoramiento continuo del software y la participación del cliente en el equipo de desarrollo. Una fortaleza particular es el desarrollo de pruebas automatizadas. Todas las pruebas deben ejecutarse con éxito cuando un incremento se integra en un sistema.

Nota: Adaptado de Santander Universidades (2020)

2.12 Herramientas de desarrollo

A continuación, se hace referencia a algunas herramientas para el desarrollo de software.

2.12.1 Lenguajes

Entre los lenguajes de desarrollo utilizados para ML se encuentran:

2.12.1.1 Python

Lenguaje de desarrollo interpretado, cuya característica principal es que sea legible para personas con conocimientos básicos de programación. También tiene algunas otras características que lo han convertido en uno de los lenguajes de mayor difusión:

- Es open source, por lo que es gratuito.
- Tiene el respaldo de una gran comunidad de desarrolladores, por lo que está en constante desarrollo de librerías y aplicaciones y la solución de problemas se los puede encontrar en los foros.

- Multiparadigma, es decir utiliza distintos paradigmas de programación, facilitando su flexibilidad y facilidad de aprendizaje, independientemente de los conocimientos de quien lo utiliza.
- Por ser multiparadigma puede ser utilizado en áreas completamente diferentes.
- Se puede utilizar en varias plataformas, ejecutarlo en distintos sistemas operativos o a través de un intérprete (Visus, 2020)

El objetivo de Python es automatizar procesos, puesto que eso permite un ahorro de tiempo y posibles dificultades en el desarrollo, ya que cada uno de esos procesos se traducirán en pocas líneas de código para su inserción en sistemas operativos y plataformas.

Python se utiliza en varias industrias y campos científicos:

- Ciencia de datos
- ML
- Desarrollo web
- Educación en ciencias de la computación
- Visión por computador y procesamiento de imágenes
- Juegos
- Medicina y farmacología
- Biología y bioinformática
- Neurociencia y psicología
- Astronomía (Universia, 2020)

2.12.1.2 R

En palabras de Fernández Lizana (2020, p. 100) “R es un entorno y lenguaje de programación empleado primordialmente para efectuar análisis estadístico de datos y construcción de gráficos (…) R es un software libre que se asienta dentro del proyecto GNU y se distribuye bajo licencia GNU GPL”. Por ser lenguaje, R utiliza código para desarrollo. El comienzo de R al departamento de Estadística de la Universidad de Auckland en 1993 y posteriormente las nuevas versiones las controla el R Development Core Team, y como consecuencia, existe gran cantidad de librerías que se pueden aplicar a un sinnúmero de problemas. Es preciso señalar que R es una herramienta de gran utilidad para la ciencia de datos; también para extraer

conocimiento tomando información de distintas bases de datos (Main Yaque et al., 2019).

La principal característica de R “es que forma un entorno de análisis estadística para la manipulación de datos, su cálculo y la creación de gráficos. (…) puede considerarse como otra implementación del lenguaje de programación S, con la particularidad de que es un software GNU (Contreras García et al., 2011). El entorno de R incluye:

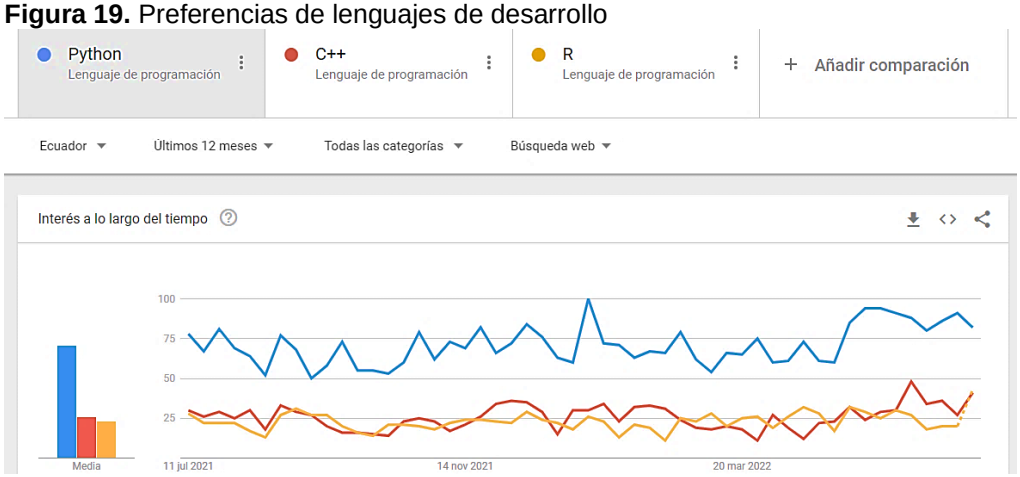
- Instalación segura para manejar y almacenar datos.
- Operadores para matrices.
- Colección de herramientas intermedias que sirven para el análisis de datos.
- Gráficos para análisis y visualización de los datos.
- Incluye bucles, condicionales, recursividad y facilidad de entrada y salida (R Foundation, 2022)

2.12.1.3 C++

Lenguaje orientado a objetos que se deriva de C de propósito general, que surgió para adicionarle algunas funcionalidades y características, como “nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería” (EcuRed, 2019, párr. 1).

En la Figura 19 se muestran las preferencias de los lenguajes de programación antes mencionados.

Figura 19. Preferencias de lenguajes de desarrollo



Nota: Tomado de Google Trends (2022)

2.12.2 Frameworks

Existen algunos frameworks que facilitan el desarrollo aplicaciones, hemos elegidos Django, Laravel, CodeIgniter como objetos de estudios de aquí seleccionaremos uno para el desarrollo de nuestra propuesta.

2.12.2.1 Django

Es un marco de trabajo web para Python para el desarrollo de sitios web seguros y fáciles de mantener. Facilita el desarrollo al permitir escribir el código de la aplicación web sin tener que reinventar nada; es de código abierto, soportado por una comunidad que ofrece muy buena documentación y opciones gratuitas y de pago (MDN Web Docs, 2022), escrito en Python y cumple de cierta forma el paradigma Modelo Vista Controlador (MVC) (EcuRed, 2013). Entre sus características se pueden mencionar:

- Realiza mapeo objeto-relacional.
- Las aplicaciones se pueden instalar en cualquier página que funcione con Django.
- Tiene una robusta API de base de datos.
- Sistema de vistas genéricas.
- Sistemas de plantillas con base en etiquetas.
- Despachador de URLs con base en expresiones regulares.
- Middleware para el desarrollo de características adicionales.
- Soporte.
- Documentación accesible (EcuRed, 2013, párr. 7).

2.12.2.2 Laravel

Laravel es un marco de aplicación web de PHP con una sintaxis expresiva y elegante, de mayor utilización y con una numerosa comunidad (Desarrollo Web, s/f; Laravel, 2022). Es de fácil mantenimiento y escalable, facilitando el desarrollo de todo tipo de proyectos; también favorece y facilita el trabajo en equipo e impulsa las mejores prácticas (Desarrollo Web, s/f).

Algunas de sus características son:

- Sistema de rutas.
- Sistema de abstracción de base de datos.
- Sistema de creación de colas de trabajo.

- Diversidad de configuraciones para envío de correos.
- Notificaciones a usuarios.
- Abstracción del sistema de archivos.
- Gestión de sesiones.
- Sistema de autenticación. (Desarrollo Web, s/f, párr. 7).

2.12.2.3 CodeIgniter

Framework de PHP potente, poderoso y liviano, destinado a programadores que requieren de herramientas simples y elegantes, para que puedan crear aplicaciones web con todas sus funcionalidades, superando permanentemente a su competencia. Cumple con el paradigma MVC, aunque no obliga en el momento del desarrollo; integra protección contra ataques CSRF y XSS, y tiene una guía de usuario completa (CodeIgniter, 2022).

CodeIgniter es adecuado para desarrolladores que requieran un framework que ocupe poco espacio, tenga un excelente rendimiento, compatibilidad con alojamiento que ejecutan versiones de PHP, poca configuración, sin líneas de comando, sin necesidad de bibliotecas monolíticas ni aprender lenguaje de plantillas, que favorezca soluciones simples y con excelente documentación.

CAPÍTULO III
METODOLOGÍA

3.1 Tipo de investigación

Se pueden mencionar algunos tipos de investigación, de los que se requiere analizar sus particularidades para determinar el más adecuado al problema de investigación que se pretende solucionar. A pesar de no existir una concordancia entre los autores sobre los tipos de investigación, se pueden señalar los siguientes: a) descriptiva, b) documental, c) correlacional, d) explicativa o causal, f) estudio de casos, g) experimental, h) histórica, i) otros tipos (Bernal Torres, 2016, p. 143).

Para efectos de este proyecto, el tipo de investigación que más se ajusta al problema es la *descriptiva*, la misma que especifican los rasgos distintivos de un problema objeto de estudio (Bernal Torres, 2016). Según Hernández Sampieri et al. (2014) la investigación descriptiva trata de encontrar las características más relevantes de un determinado fenómeno que se pretende analizar, realizando una descripción de las tendencias de la población motivo de estudio.

Este proyecto es descriptivo, ya que se identificarán las características de la construcción de los modelos ML más adecuados para el desarrollo de una herramienta que ayude a los profesionales de la salud en la detección de tumores cerebrales mediante resonancias magnéticas y el diseño del prototipo.

3.2 Enfoque de investigación

El siglo XXI ha traído grandes reflexiones en cuanto a la ciencia. Los paradigmas que existían en el siglo pasado, hallaron su interrogante a finales del mismo (Baena Paz, 2017), surgiendo corrientes de pensamiento que han encaminado a los investigadores en la búsqueda del conocimiento, polarizando dichas corrientes “en dos aproximaciones principales de la investigación: el enfoque cuantitativo y el enfoque cualitativo” (Hernández Sampieri et al., 2014, p. 4).

En la Tabla 10 se presentan las similitudes de los enfoques cualitativo y cuantitativo.

Tabla 10. Paradigmas de investigación

Similitudes	Paradigma cualitativo	Paradigma cuantitativo
Centro de interés de la investigación	Cualidad del objeto de investigación	Cantidad (cuánto, cuántos)
Raíces filosóficas, conceptos	Fenomenología e interacción simbólica	Positivismo, empirismo lógico
Conceptos asociados	Trabajo de campo, etnografía naturalista	Experimental, empírica, estadística.
Objetivo de la investigación	Comprensión, descripción, descubrimiento, generadora de preguntas de investigación	Predicción, control, descripción, confirmación, comprobación de hipótesis
Características del diseño	Flexible, envolvente, emergente	Predeterminado, estructurado
Marco o escenario.	Natural, familiar	Desconocido, artificial.
Muestra	Pequeña, no aleatoria, teórica	Grande, aleatoria, representativa
Levantamiento de información	El investigador como instrumento primario, entrevistas, observaciones	Instrumentos Inanimados (escalas, pruebas, encuestas, cuestionarios, ordenadores).
Tipo de análisis	Inductivo (por el investigador).	Deductivo (por métodos estadísticos)
Resultados	Comprensivos, expansivos	Precisos, limitados, reduccionistas

Nota: Adaptado de Gallardo (2017)

El enfoque metodológico que se aplica a este proyecto es el *cualitativo*, el mismo que se fundamenta en el levantamiento y estudio de los datos, con la finalidad de perfeccionar las preguntas de investigación o descubrir nuevas preguntas durante el proceso de interpretación. El levantamiento de información se sustenta en métodos no estandarizados y se refiere a alcanzar el punto de vista, experiencias y todo lo relacionado con los involucrados en el problema de estudio; además, son importantes las relaciones de las personas o grupos. Se puede cuestionar de forma más abierta a los participantes, de los que se puede levantar información por medio de lenguaje escrito, verbal, no verbal o visual, permitiendo describirla, analizarla y convertirla en temas vinculantes al estudio (Hernández Sampieri et al., 2014).

Este estudio es cualitativo porque se pretende analizar la necesidad de diseñar la herramienta informática desde el punto de vista de los profesionales

de salud, para desarrollar un modelo ML que detecte tumores cerebrales con base en resonancias magnéticas.

3.3 Técnicas e instrumentos de recolección de datos

Para el levantamiento de información se utilizará **técnica** de la *entrevista estructurada*, la que se aplicará a tres profesionales médicos para conocer su opinión sobre el desarrollo de esta herramienta informática.

El **instrumento** de recolección de datos es la *guía de entrevista*, la misma que consta de las siguientes preguntas:

- ¿Cuál es el procedimiento para detectar un tumor cerebral?
- ¿Considera usted que los factores raza(sexo) puedan afectar la detención de los tumores cerebrales?
- ¿Considera interesante el desarrollo un modelo predictivo para la detección de tumores?
- ¿Cree usted que una herramienta tecnología podría agilizar el proceso de detección de tumores?

Además, se utilizará la **técnica** de *análisis documental*, y el **instrumento** utilizado será un *resumen* en referencia a las características de las imágenes de las resonancias magnéticas, que se tomarán de la base de imágenes Kaggle.

Para el levantamiento de información no se requirió establecer la población y muestra, dada la existencia de elementos teóricos y documentales que sirvieron de base para complementar los resultados. Sin embargo, para el caso de la entrevista se realizó una selección de carácter intencional, de profesionales médicos, que aceptaron participar en este estudio.

3.4 Metodología de desarrollo

La metodología de desarrollo utilizada para este proyecto es el *prototipado evolutivo*, que es ciclo de vida iterativo de desarrollo de software, mediante el cual se pueden determinar los requisitos del usuario final, por medio de la construcción prototipo que simulan las funcionalidades del producto a crear. Si este producto no responde a las necesidades del cliente, se procede a la construcción de otro prototipo con las mejoras realizadas, lo que significa que el diseño evoluciona acoplándose a los requerimientos, a pesar de que su

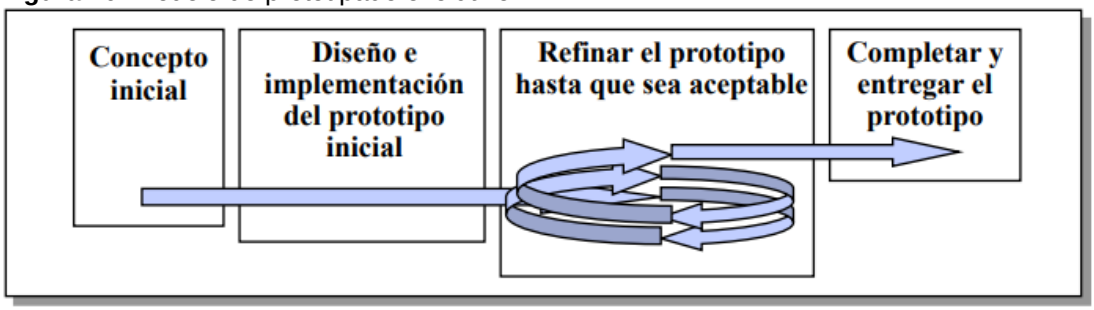
funcionalidad es simulada hasta que se completan todos los requisitos al validarse el último prototipo (Zumba Gamboa & León Arreaga, 2018).

Entre las características del prototipado evolutivo se encuentran:

- Este enfoque de desarrollo es utilizado cuando se desconoce con certeza el producto a construir.
- Se inicia con el diseño e implementación de los elementos de mayor importancia del sistema.
- La evaluación del prototipo ofrece al desarrollador la retroalimentación que se requiere para su mejoramiento.
- La evolución del prototipo se convierte en el sistema final (García Peñalvo et al., 2020).

En la Figura 20 se muestra un bosquejo del proceso de desarrollo de un prototipo.

Figura 20. Modelo de prototipado evolutivo



Nota: Tomado de García Peñalvo et al. (2020)

3.5 Técnicas para el procesamiento y análisis de resultados

Se procede hablar con tres profesionales médicos en el área de la neurología, para tener una perspectiva médica frente a la tecnológica. El resultado de la entrevista se analiza en los párrafos a continuación.

En cuanto a la primera pregunta sobre el procedimiento para la detección de los tumores cerebrales, los entrevistados estuvieron de acuerdo en que el diagnóstico se puede realizar por medio de imágenes; de forma particular, sobre resonancias magnéticas, en algunos casos puede acompañar de una espectroscopía, aunque las resonancias son la primera opción.

Para la segunda pregunta, relacionada con los factores que podrían afectar la detección de tumores cerebrales tales como raza o sexo, los profesionales

respondieron que éstos no son factores influyentes para este tipo de afectaciones.

Sobre la tercera pregunta, que tiene relación con el interés del desarrollo de un modelo predictivo para la detección de tumores, los entrevistados coincidieron que, desde el punto de vista tecnológico, si es interesante el contar con este tipo aplicación, ya que estos avances son útiles para el diagnóstico temprano de los tumores.

Finalmente, la cuarta pregunta sobre la creencia de que una herramienta tecnológica puede ser útil para agilizar el proceso de detección de tumores cerebrales, los profesionales de la salud supieron manifestar que, desde el ámbito de la tecnología, cualquier herramienta o programa informático que ayude a mejorar el diagnóstico de una enfermedad, en este caso, la detección de los tumores cerebrales, debería ser probada y evaluada. Los profesionales coincidieron que la tecnología avanza vertiginosamente y, por ende, no todos se encuentran actualizados en el manejo de la misma, pero si están de acuerdo que implementar un programa para servir de soporte en la detección de los tumores, si sería conveniente.

Finalizada la entrevista se procedió a mostrar algunas imágenes de resonancias magnéticas que forman parte del conjunto de entrenamiento, con la finalidad de validar si las máscaras que acompañan a estas resonancias correspondían a la presencia o ausencia de la afectación. De las resonancias observadas se concluyó que estuvieron correctas. Sin embargo, debido a que el conjunto de imágenes fue numeroso no se pudo validar la totalidad de las resonancias, por lo que no se pudo concluir que todas las imágenes estuvieron correctamente acompañadas con su máscara.

Por último, de las entrevistas realizadas y la revisión de las imágenes para validación, se concluye que el desarrollo de este proyecto es de interés para la comunidad médica, además que representa un aporte desde un punto de vista tecnológico, puesto que esta herramienta podría utilizarse como un soporte en la clasificación de tumores cerebrales en función de las imágenes por resonancias magnéticas.

CAPÍTULO IV

PROPUESTA TECNOLOGICA

En este capítulo se presenta el modelo de clasificación para la detección de tumores cerebrales, además de la metodología de desarrollo para la construcción de la interfaz del sistema.

4.1 Construcción del algoritmo

Se desarrolla una solución informática mediante un modelo ML-DL para determinar la presencia de tumores cerebrales en resonancias magnéticas, solución que será de apoyo al profesional médico como herramienta de soporte en el diagnóstico de los tumores cerebrales.

Para el desarrollo y entrenamiento del modelo, se utilizó la plataforma Google Colab, mediante la cual se puede programar y ejecutar el lenguaje Python en el navegador, sin necesidad de configuraciones.

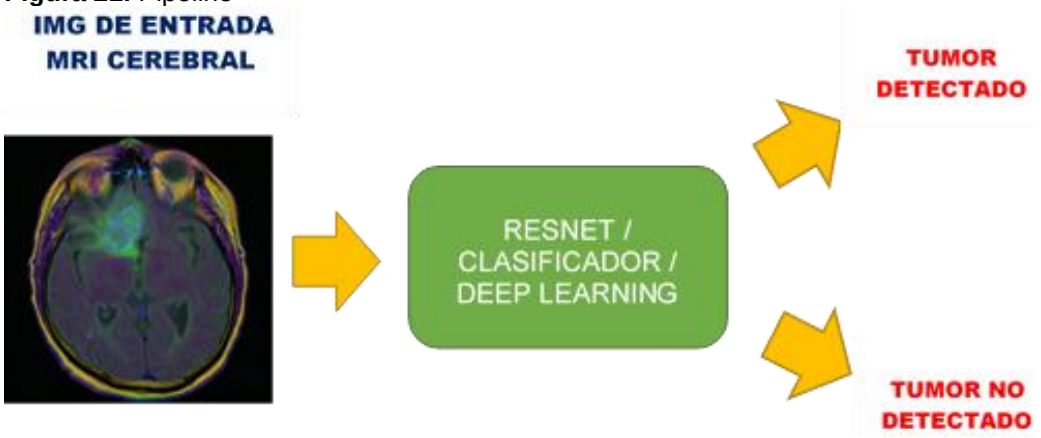
Para el entrenamiento del modelo se procedió a la búsqueda de un conjunto de datos de la base de datos Kaggle, que presenta código y datos para trabajar con ciencia de datos. Existen alrededor de 50.000 conjunto de datos públicos para su revisión y análisis, en el menor tiempo (Kaggle, 2022).

Una vez desarrollado el modelo a través de Google Colab, se exporta el archivo h5 y json para proceder a la construcción del prototipo de interfaz web para la presentación de las predicciones. En los párrafos siguientes se presenta el proceso de construcción.

4.1.1 Construcción del modelo

Pipeline de aprendizaje profundo en capas para el modelo de clasificación

Figura 21. Pipeline



Para la construcción y el entrenamiento del modelo se procedió a descargar un conjunto de imágenes de resonancias magnéticas de la plataforma de kaggle (Kaggle, s/f) . En dicho repositorio se pueden encontrar un total de 3929 imágenes ya clasificadas; 2556 sanas y 1373 con tumores.

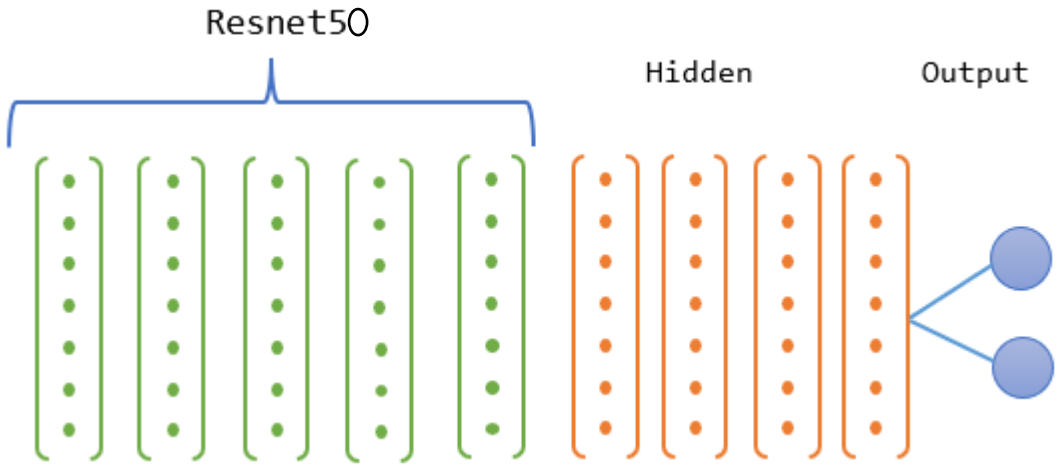
Kaggle es una plataforma web que facilita el trabajo para temas relacionado con ML ya que reúne diferentes sets de datos, los mismos que pueden ser utilizados en la construcción de múltiples modelos predictivos para aplicarlos en diferentes problemáticas.

Para el desarrollo del modelo se utilizó la librería keras que permiten el uso de redes (arquitecturas) previamente entrenadas con grandes bases de datos, para posteriormente adaptarlas al objeto de estudio mediante la técnica del aprendizaje por transferencia. A medida que las CNN son más profundas se presenta el inconveniente denominado como el desvanecimiento del gradiente lo cual afecta a la eficacia de la red. La llamada red neuronal residual (ResNet) presentó una arquitectura novedosa, que permite entrenar una red de 152 capas sin presentar el problema por el desvanecimiento del gradiente logrando como resultado, en un conjunto de estas redes (RESNET), una tasa de error del 3.57% en el conjunto de test de *Imagenet* frente al resto de arquitecturas tales como VGGNet, AlextNet, entre otras; esto ha demostrado el gran avance de este tipo de redes.

4.1.2 Arquitectura del modelo

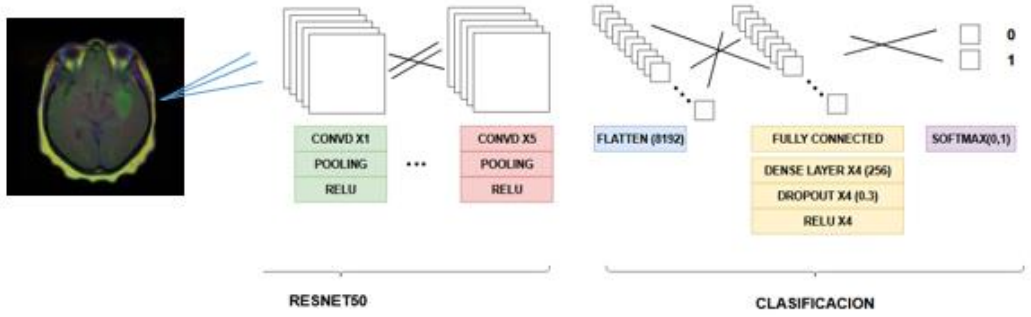
La construcción del modelo se lleva a cabo en función del uso de dos CNN

Figura 22. Arquitectura del modelo



La primera red, la restnet50 es la encargada de extraer las características de la imagen por medio de los kernel, a través de 5 capas de convoluciones con sus respectivas capas de pooling, average pooling, relu, entre otras propias de la arquitectura resnet50. Para la segunda red, la cual añadimos al final de este modelo por medio de la capa de flatten que sirve como input para nuestra red de clasificación en donde utilizamos 4 capas densas con sus respectivos dropout y funciones de activación relu.

Figura 23 Diagrama del modelo



Se procede a cargar el modelo ResNet50 ya entrenado con los pesos de enteramientos utilizados en el *dataset* de *Imagenet*, el cual fue entrenado con aproximadamente 8 millones de imágenes; se excluye la capa final, ya que es la que se utilizará para realizar el modelo de clasificación.

A continuación, se muestra el código generado para el modelo de ResNet50.

```
# Obtenemos el modelo base de ResNet50
basemodel = ResNet50(weights = 'imagenet', include_top = False,
input_tensor = Input(shape=(256, 256, 3)))
```

A continuación, se muestran los parámetros del modelo base:

```
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

Posterior a esto se procede a congelar los pesos. A continuación, se muestra el código para este proceso.

```
# Congelamos los pesos del modelo base

for layer in basemodel.layers:
```

```
layers.trainable = False
```

4.1.3 Entrenamiento del modelo

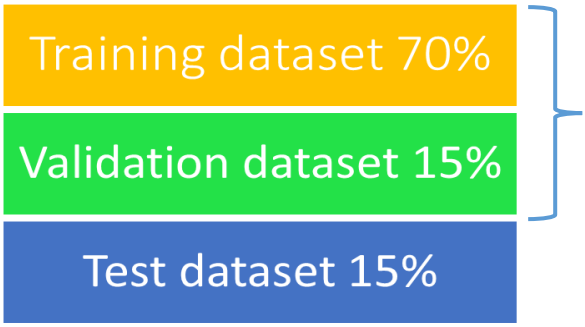
Establecida la arquitectura, se procede con el entrenamiento del modelo; el total de imágenes a usar es de 3929: 2556 sanas y 1373 con tumores.

Para el entrenamiento de la red se utiliza el aprendizaje por transferencia, teniendo en cuenta que para evitar que el modelo caiga en el *overfiting* se procede a dividir el conjunto de entrenamiento de la siguiente manera:

- 85% Training.
- 15% Validation.
- 15% Testing.

Del conjunto del training se reserva el 15% para la validación cruzada durante él la fase del training, mientras que le otro 15% se lo utilizará en la fase de *testing*. Todo esto con la finalidad de evaluar qué tan bien está generalizando el modelo por medio de imágenes que no forman parte del training, las cuales son proporcionadas al final de cada *epochs* pertenecientes al conjunto del set de *validation*.

Figura 24. Conjunto de entrenamiento



Se procede a dividir los datos de entrenamiento, *testing* y validación con librería *sklearn* utilizando la clase *train_test_split*. Para poder procesar las imágenes es necesario crear un generador de imágenes para procesarlas por lotes. El código generado se presenta a continuación.

```
# Dividir los datos en entrenamiento y testing

from sklearn.model_selection import train_test_split

train, test = train_test_split(brain_df_train, test_size = 0.15)

# Creamos el generador de imágenes
from keras_preprocessing.image import ImageDataGenerator
```

```
# Creamos un generador de datos que escale los datos de 0 a 1 y h
aga una división de validación de 0,15
datagen = ImageDataGenerator(rescale=1./255., validation_split =
0.15)
```

El código generador de imágenes para *train*, *test* y *validation* se presenta a continuación.

```
#generar conjunto de entrenamiento
train_generator=datagen.flow_from_dataframe(
dataframe=train,
directory= './',
x_col='image_path',
y_col='mask',
subset="training",
batch_size=16,
shuffle=True,
class_mode="categorical",
target_size=(256,256))

#generar conjunto de validacion al final de cada epochs
valid_generator=datagen.flow_from_dataframe(
dataframe=train,
directory= './',
x_col='image_path',
y_col='mask',
subset="validation",
batch_size=16,
shuffle=True,
class_mode="categorical",
target_size=(256,256))

# Creamos un generador de datos para imágenes de prueba entra en
accion al final del algoritmo
test_datagen=ImageDataGenerator(rescale=1./255.)

test_generator=test_datagen.flow_from_dataframe(
dataframe=test,
directory= './',
x_col='image_path',
y_col='mask',
batch_size=16,
shuffle=False,
class_mode='categorical',
target_size=(256,256))
```

Posteriormente se agregan unas capas adicionales del modelo para poder realizar la clasificación binaria 1 y 0, presencia o ausencia, capas densas con la función de activación *relu, droupouts* y una función *softmax* para la salida.

Inicialmente probamos con dos capas densas, dos droupouts con sus respectivas funciones relu y durante 25 epochs, se obtuvo una tasa de acierto del 91% y en la fase de testing una tasa de acierto del 85%, por lo que se decidió agregar dos capas más a cabecera final y aumentar los epochs 60

A continuación, el código que genera las capas.

```
# Agregamos una cabecera de clasificación al modelo base de RESNE
T50

headmodel = basemodel.output #headmodel da una salida
headmodel = AveragePooling2D(pool_size = (4,4))(headmodel) # aver
age pool
headmodel = Flatten(name= 'flatten')(headmodel) # faltten
headmodel = Dense(256, activation = "relu")(headmodel) #capa dens
a con su función relu
headmodel = Dropout(0.3)(headmodel) # dropout 30% tecnica de regu
larización, solo en la f train
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dense(2, activation = 'softmax')(headmodel)

model = Model(inputs = basemodel.input, outputs = headmodel)
```

Se procede a visualizar la cantidad de parámetros que se agregaron al nuevo modelo.

```
Total params: 26,146,178
Trainable params: 26,093,058
Non-trainable params: 53,120
```

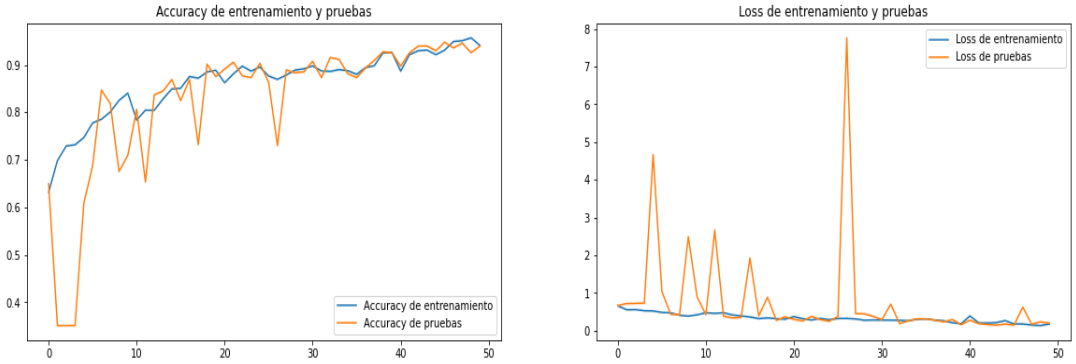
En la Tabla 11 se muestra la descripción de los parámetros del modelo.

Tabla 11. Parámetros del modelo

MODELO	PARAMETROS	HIPERPARAMETROS	DATASET
INICIAL	23,587,712		
FINAL	26,146,178	**Optimizador**: Adam **Función activación**: ReLu **Función perdida**: Binary Crossentropy	Dataset 3929

Se procede a graficar *loss* y *accuracy* del entrenamiento, en donde se puede observar cómo el error va disminuyendo. Se grafica *val_loss* y *val_accuracy* que ocurre después de cada *epochs*.

Figura 25. val_loss y val_accuracy



4.1.4 Evaluación del modelo entrenado

Para poder evaluar el modelo se utiliza la matriz de confunción y los indicadores de *Accuracy, Precision y Recall*. Para evitar volver a entrenar el modelo, se procede a cargar el modelo ya preentrenado.

```
# Cargamos el modelo preentrenado (en lugar de entrenar el modelo
 durante más de 1 hora)
with open('classifier-resnet-model0108.json', 'r') as json_file:
    json_savedModel= json_file.read()
# Cargar el modelo
model = tf.keras.models.model_from_json(json_savedModel)
model.load_weights('classifier-resnet-weights010822.hdf5')
model.compile(loss = 'categorical_crossentropy', optimizer='adam'
, metrics= ["accuracy"])
```

Se realiza la predicción y se la guarda en la variable *test_predict*, gracias a la función de *softmax*. El resultado que se obtiene este dato como la probabilidad entre la clasificación. La función *argmax* devuelve la probabilidad más alta de las columnas disponibles; para este caso va a tener solo dos columnas.

```
# Hacemos la predicción
test_predict = model.predict(test_generator, steps = test_generat
or.n // 16, verbose =1)
test_predict.shape # objeto que contiene la predicción con cero y
 uno... Dos salidas, dos respuestas
```

Figura 26. Resultado

```
test_predict #cual prediccion es mas grande, la probabilidad que no tenga y que si no tenga

array([[9.9954993e-01, 4.5006198e-04],
       [2.0431106e-04, 9.9979573e-01],
       [1.1987397e-11, 1.0000000e+00],
       ...,
       [9.3862498e-01, 6.1374985e-02],
       [1.8005748e-08, 1.0000000e+00],
       [8.3786285e-01, 1.6213720e-01]], dtype=float32)
```

```
# Obtenemos la clase predicha a partir del modelo
predict = []

for i in test_predict:
  predict.append(str(np.argmax(i))) #argmax devuelve la posibilid
ad max de las posibles columnas

predict = np.asarray(predict)
# Dado que usamos el generador de prueba, se limita el número de
imágenes a leer (predecir), debido al tamaño del lote
original = np.asarray(test['mask'])[:len(predict)]
len(original)
```

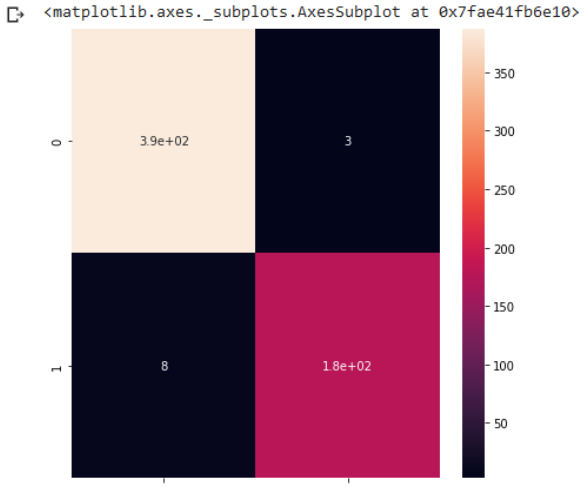
Se procede a obtener la *Accuracy* del modelo con el data set de pruebas y la matriz de confusión (ver Figura 26).

```
# Obtenemos la tasa de acierto del modelo
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(original, predict)
accuracy
```

```
0.9756944444444444
```

```
# Representamos la matriz de confusión
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(original, predict)
plt.figure(figsize = (7,7))
sns.heatmap(cm, annot=True)
```

Figura 27. Matriz de confusión



Para poder evaluar más aún el modelo se utilizan los siguientes indicadores que están disponible en una clase de la librería `classification_report` para poder mostrar `precisión, recall, f1-score`

```
from sklearn.metrics import classification_report

report = classification_report(original, predict, labels = [0,1])
print(report)
```

Figura 28. Reporte de precision, recall, score

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       368
           1       0.98      0.97      0.97       208

   micro avg       0.98      0.98      0.98       576
   macro avg       0.98      0.98      0.98       576
weighted avg       0.98      0.98      0.98       576
```

4.2 Desarrollo

En esta sección se desarrolla la interfaz para poder utilizar el modelo elaborado. Para poder usar el modelo se debe exportar un archivo h5 y el archivo Json del modelo.

4.2.1 Fase 1: Concepto inicial

La idea de la construcción de este prototipo surgió de la necesidad de que los profesionales de la salud, tanto de instituciones públicas como privadas, dispongan de una herramienta apoyo y soporte para la detección de tumores cerebrales a través de resonancias magnéticas. El desarrollo de esta

herramienta tecnológica puede representar un complemento al diagnóstico de los tumores cerebrales

Para la construcción de esta interfaz se utilizaron algunas herramientas de desarrollo. La selección de las mismas dependió de las funcionalidades que tienen y a su utilidad para el desarrollo.

En los párrafos a continuación se describen las herramientas utilizadas.

4.2.1.1 Python

Es un lenguaje de programación de alto nivel que se caracteriza por su simplicidad en la legibilidad de su sintaxis lo que facilita el aprendizaje y el entendiendo del lenguaje. Además, está basado en el paradigma de la programación orientada a objetos convirtiéndolo en un lenguaje muy demandado como se menciona en capítulos anteriores. Está disponible de manera gratuita y es multiplataforma.

Comparado con otros lenguajes principalmente utilizados en la ciencia de datos, Python, resulta muy adecuado debido a que permite tener un código más riguroso y fácil de mantener logrando cada vez más un impulso progresivo para el uso de este lenguaje en diferentes campos de acción tales como: visión computación, estadística, ML, entre otras (Yu et al., 2022).

4.2.1.2 Django

Es un framework de desarrollo web open source desarrollado en Python. Su principal característica se basa en la facilidad que otorga al permitir construir sitios webs de manera ágil y fáciles de mantener, sin dejar a un lado la seguridad de la aplicación.

Las características para resaltar de este framework son las que se describen en la Tabla 12.

Tabla 12. Características de Django

Características	Descripción
MVT (Model View Template)	Respeta el tradicional patrón de diseño (Model View Controler)
Entity	Facilita la creación de objetos tipos entity.
Templates	Sistema de plantillas para la creación de interfaces
Seguridad	Provee de mecanismos que protegen las bases de datos, los formularios, entre otros.

Documentación	Tiene una comunidad muy activa y su documentación es exhaustiva.
Escalabilidad	Se basa en aplicaciones modulares.
ORM (Object Relational Mapping)	Provee de un sitio administrativo que facilita el acceso a la interfaz de la base de datos

Se eligió Django como entorno de trabajo por las características mencionadas anteriormente. Además de que su uso es gratuito, tiene una gran comunidad activa lo que puede llegar a facilitar la resolución de problemas por medio de las interacciones que se realicen en estos blogs.

4.2.2 Otras herramientas

4.2.2.1 Sublime Text

Sublime Text es un editor de texto que permite escribir código fuente para casi todos los lenguajes de programación más demandados en la actualidad. Es ligero, multiplataformas y fácil de usar; disponible en dos versiones: portable e instalable. Aunque no sea totalmente un software open source, su versión de evaluación es completamente funcional, ya que cuenta con un período de prueba ilimitado.

Se eligió este entorno de desarrollo por su simplicidad la cual fue una característica muy importante a la hora de seleccionar, optándose por una versión portable, que soporta de manera nativa el lenguaje de programación Python a utilizar para el desarrollo del proyecto.

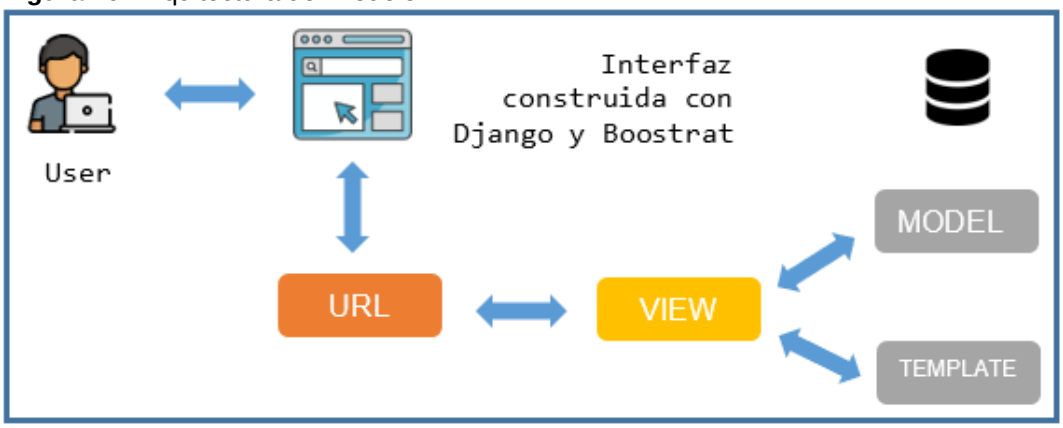
4.2.2.2 SQLite

SQLite es una plataforma de open Source, fácil de instalar y sencilla para ejecutar debido a que comparada con otros gestores de base de datos requiere menos configuraciones (Yuvaraj et al., 2020). Python incluye soporte nativo desde la versión SQlite 2.5 haciendo que sea fácil de integrar este tipo de bases con Django. Cabe destacar que la integración de esta base se la realiza desde Django por medio del archivo "`settings.py`" y la creación del diseño de la base también se realiza por medio del framework bajo el concepto de modelo, en donde se definirá el nombre de las tablas y el tipo de dato.

4.2.3 Fase 2: Diseño y desarrollo del concepto inicial

4.2.3.1 Arquitectura de la solución

Figura 29. Arquitectura del modelo ML

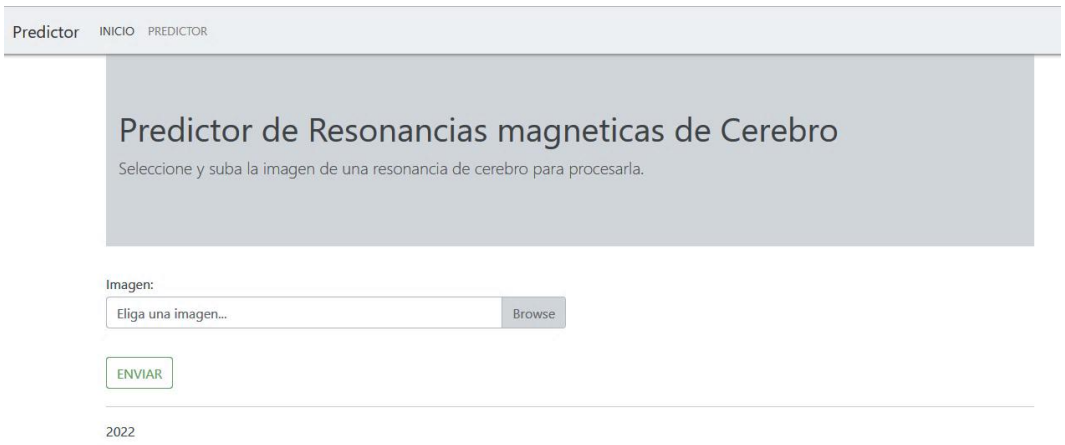


Se empieza con una petición *url* por parte del usuario a través de una interfaz, de la que pasa a la *view* (que en Django hace la función del controlador); ésta se encarga de mostrar el template (vista), que retorna al *user*. También se encarga de enviar a guardar los resultados a la base.

4.2.3.2 Prototipo inicial:

De acuerdo con la característica principal que se busca, se plantea el primer modelo de la interfaz, el que permite realizar la carga de una imagen señalando la ruta del archivo a través del botón *browser*.

Figura 30. Selección de imagen

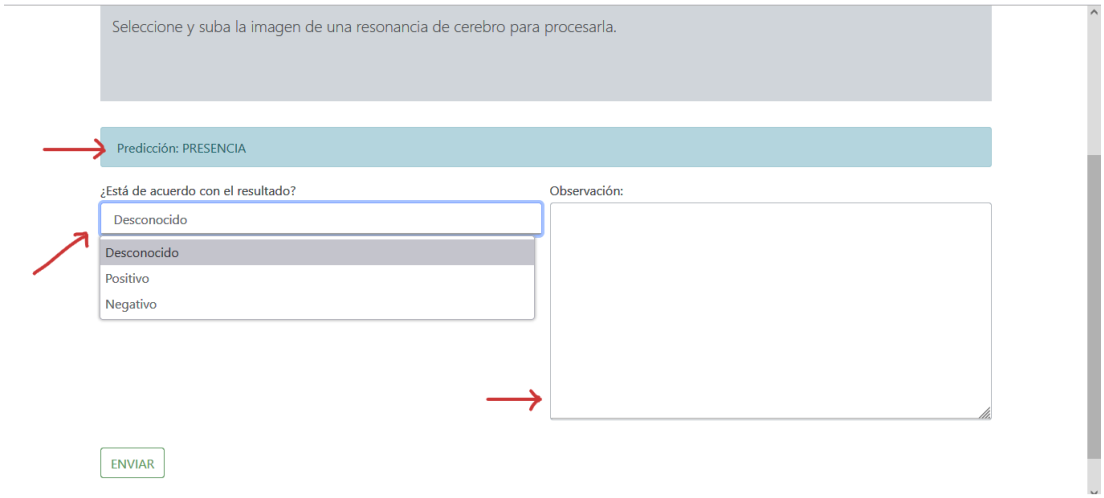


Una vez seleccionada la imagen se habilita el botón Visualizar Imagen, que tiene la función de previsualizar la imagen cargada.

Figura 31. Visualización

Enviada la imagen a procesar, la interfaz devuelve el resultado de la **predicción**, y se muestra una lista para indicar si está de acuerdo con el resultado. Además, se añade un campo de observación que permitirá añadir una retroalimentación al resultado que posteriormente se la enviará a guardar.

Figura 32. Predicción



4.2.4 Fase 3: Refinar el prototipo hasta que sea aceptable

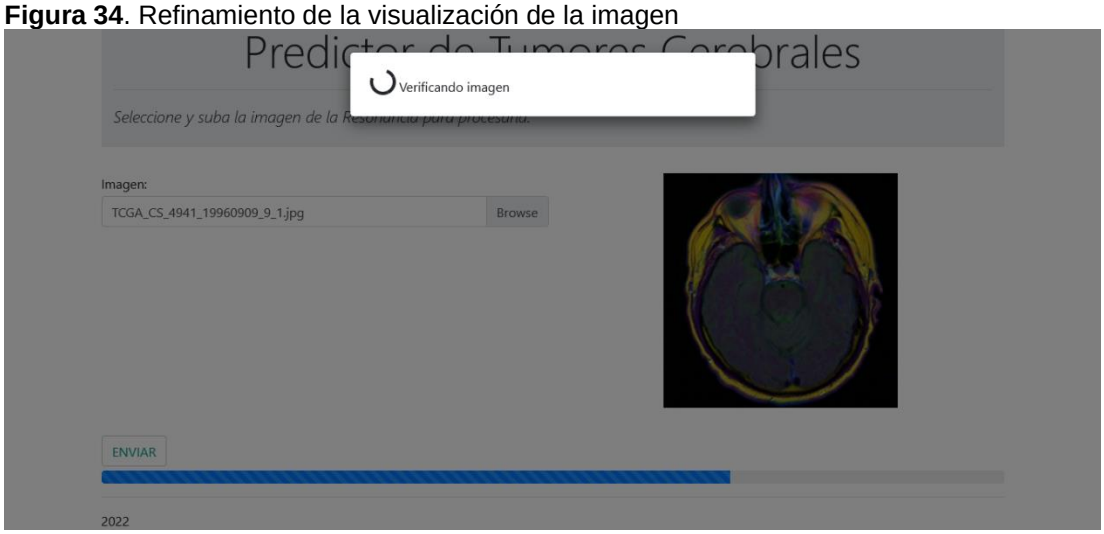
Esta fase está representada por modelo 2 del prototipo,

Figura 33. Refinamiento

Se procede a añadir una interfaz de inicio en donde se agrega una descripción general del proyecto, se proporciona estilo y funcionabilidad al *navbar* que permite la navegación entre las interfaces.

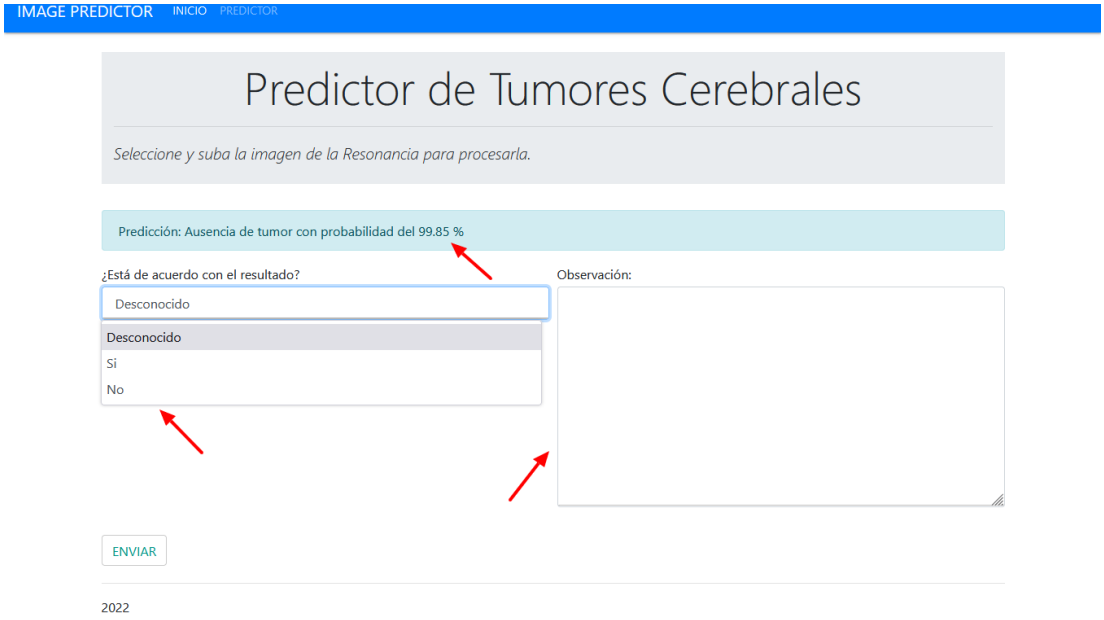
Se realizaron ajustes al procesar la imagen, se añade un *progress bar* para visualizar el progreso que tarda en mostrar el resultado. Además, se elimina el botón de visualizar imagen, para que una vez que se cargue la imagen se muestre directamente en pantalla

Figura 34. Refinamiento de la visualización de la imagen



Se realiza ajuste para que se muestre un porcentaje de probabilidad, remplazamos las opciones de positivo, negativo por sí o no en el list que se consulta si está de acuerdo con el resultado, se mantiene el botón de observaciones con la finalidad de tener feedback del resultado.

Figura 35. Refinamiento de la pantalla de predicción



Para poder revisar los resultados, se procede a utilizar sitio administrativo que ya proporciona Django. Para ello se debe crear el usuario *admin* a través de los siguientes comandos:

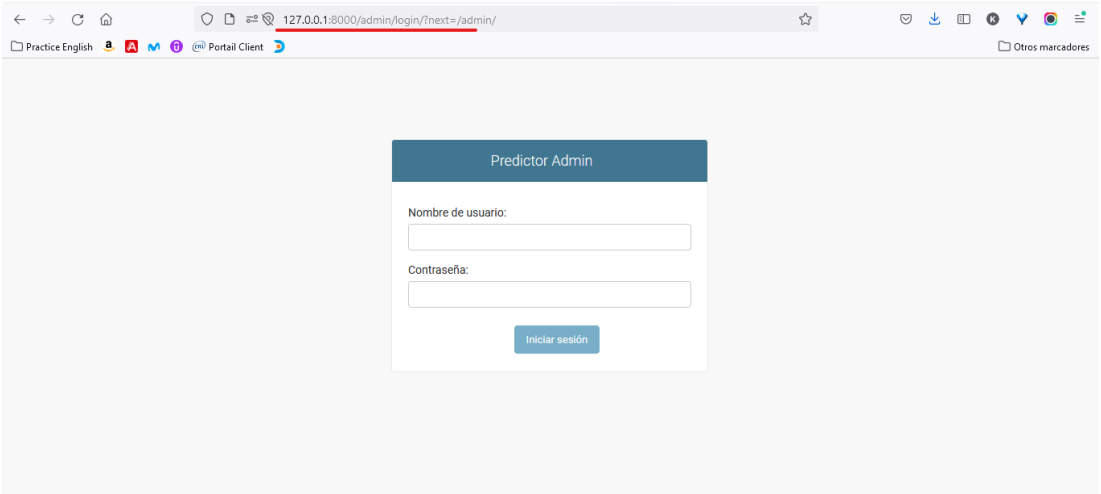
- Se ingresa al entorno virtual
- Se ejecuta el comando "`python manage.py createsuperuser`"
- Se deben seguir los pasos para crear el *super-user*,
- Para poder ir a la pantalla del *login* se debe acceder a la ruta: `http://localhost:8000/admin/`

Figura 36. Levantamiento e implementación del aplicativo

```
kevin@DESKTOP-9SPCFA7: ~/project/myenv
(myenv) kevin@DESKTOP-9SPCFA7:~/project/myenv$ python manage.py createsuperuser
2022-08-16 09:28:56.218991: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic libra
ry 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open shared object file: No such file or directory
2022-08-16 09:28:56.219098: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do n
ot have a GPU set up on your machine.
Nombre de usuario: admin1
Dirección de correo electrónico: admin1@gmail.com
Password:
Password (again):
Esta contraseña es demasiado corta. Debe contener al menos 8 caracteres.
Esta contraseña es demasiado común.
Esta contraseña es completamente numérica.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(myenv) kevin@DESKTOP-9SPCFA7:~/project/myenv$
```

Una vez en la pantalla del *login*, se procede a ingresar con las credenciales creadas.

Figura 37. Pantalla de ingreso



Se utilizará el sitio de administración de Django para poder crear, consultar, eliminar o actualizar datos de la aplicación.

Figura 38. Pantalla Admin resultados

4.2.5 Fase 4: Completar y entregar el prototipo

Una vez terminado el prototipo, comprobando su total funcionalidad, se lo deja a disposición de la comunidad médica y cualquier profesional que quiera ponerlo en funcionamiento para poder verificar las características de esta herramienta, como soporte y apoyo en la detección de tumores cerebrales a través de resonancias magnéticas.

CONCLUSIONES

Se desarrolló un prototipo de ML para la detección de tumores cerebrales mediante el uso resonancias magnéticas con el objetivo de que sirva como una herramienta de soporte a los profesionales de la salud.

Para el desarrollo de este proyecto se revisaron las distintas teorías relacionadas al ML, su importancia y su aplicabilidad en la detección de tumores cerebrales. A través de la investigación se conoció todo lo relacionado con ML, desde su definición, sus inicios, su fundamento, además de sus campos de aplicación: medicina, educación, construcción, finanzas; para este estudio, su aplicación se encuentra orientado en el ámbito de la salud para el diagnóstico clínico, la predicción de la eficacia en nuevos tratamientos para enfermedades y en la toma de decisiones que surgen luego de identificar patrones; dentro de la rama del ML encontramos el Deep learning y que por medio de las redes neuronales convoluciones encontramos las más apropiadas para el proyecto.

El desarrollo del proyecto requirió de la selección de las herramientas de desarrollo, para lo que se realizó una selección de lenguajes, frameworks y otras herramientas, con las que se podía llevar a cabo la construcción del prototipo. De estas herramientas se escogió Python como lenguaje de desarrollo y su librería keras, puesto que es adecuado para ámbito de ciencias de datos y el Deep Learning; junto a este lenguaje se utilizó el framework Django para la construcción de la interfaz web, por la flexibilidad, documentación exhaustiva y la comunidad activa que tiene este framework que permiten el desarrollo de aplicaciones sencillas y escalables. Además, se utilizaron otras herramientas como Sublime Text como entorno de desarrollo y SQLite como base de datos de acuerdo con la facilidad de integración que tiene con el framework seleccionado, como complementos del desarrollo.

Se desarrollo el prototipo de ML para automatizar el proceso de detección de tumores cerebrales mediante una interfaz web, que facilitó la puesta en marcha del modelo ML construido.

Se evaluaron los resultados del prototipo del modelo ML, por medio de sus tasas de aciertos, lo que significa que, para llegar a la mejor precisión del modelo, se dividió el conjunto de datos en 15% para testing y el 85% para el

entrenamiento; de este 85% se utilizó el 15% para la validación durante el entrenamiento. Se realizó el modelo de entrenamiento, iniciándose con 25 epochs y dos capas densas de 256 con la función de activación ReLu, y dos capas de dropout, teniendo una tasa de acierto del 91%. Sin embargo, en la fase del testing no se obtienen buenos resultados. Se refinó el entrenamiento añadiendo dos capas densas adicional de 256 con su respectiva función de activación ReLu y de dropout, logrando una tasa de acierto del 98% en el entrenamiento. Posteriormente en los resultados de testing Obtuvimos una tasa del 97% acierto. Debido a que la tasa de acierto no puede ser concluyente, se utilizaron otros indicadores tales como: precision, recall, f1 score, para la clasificación 0 (ausencia) con valores de 98%, 99% y 99% respectivamente. Y, para la clasificación 1 (presencia), los valores de 98%, 96% y 97% antes expuestos.

RECOMENDACIONES

Sería conveniente complementar el prototipo del modelo desarrollado para que no solamente prediga la presencia de un tumor, sino que permita localizar la región de la afectación en la resonancia magnética, para lo que sería conveniente la utilización de un conjunto de entrenamiento balanceado y otro modelo de red neuronal.

Para la puesta en marcha del aplicativo, el ambiente adecuado de implementación debería ser Linux. Si se encuentra trabajando con Windows, puede utilizarse la herramienta WSL para simular el ambiente.

Para realizar la carga de la imagen a procesar esta deberá tener un formato jpg. o png.

Sería conveniente la creación del Super Usuario antes de comenzar a utilizar el prototipo, con los pasos mencionados anteriormente.

REFERENCIAS BIBLIOGRAFICAS

Abeliuk, A., & Gutiérrez, C. (2021). Historia y evolución de la inteligencia artificial. *Revista Bits de Ciencia*, *21*(14–21). https://www.dcc.uchile.cl/Bitsdeciencia21.pdf

Aguado López, I. (2020). *Deep Learning: Redes Neuronales Convolucionales en R* [Grado en Matemáticas, Uniersidad de Sevilla]. https://idus.us.es/bitstream/handle/11441/114957/GM%20Aguado%20L%c3%b3pez%20Inmaculada.pdf?sequence=1&isAllowed=y

Algotive. (2022). *Machine Learning: ¿Qué es el aprendizaje automático y cómo funciona?* https://www.algotive.ai/es-mx/blog/machine-learning-que-es-el-aprendizaje-automático-y-cómo-funciona

Álvarez, M., Quirós, L., & Cortés, M. (2020). Inteligencia artificial y aprendizaje automático en medicina. *Revista Médica Sinergia*, *5*(8), e557–e557. https://doi.org/10.31434/rms.v5i8.557

Areli-Toral Barrera, J. (2015). *Redes Neuronales*. http://www.cucei.udg.mx/sites/default/files/pdf/toral_barrera_jamie_areli.pdf

Aulaplaneta. (2020). *Machine Learning: ¿El futuro de la educación?* https://www.aulaplaneta.com/2020/10/29/recursos-tic/machine-learning-el-futuro-de-la-educacion

Baena Paz, G. (2017). *Metodología de la investigación* (Tercera). Grupo Editorial Patria, S.A. de C.V. http://www.biblioteca.cij.gob.mx/Archivos/Materiales_de_consulta/Drogas_de_Abuso/Articulos/metodologia%20de%20la%20investigacion.pdf

Barrero, G. (2018). *Predicción de la calidad de software desarrollado en IBM RPG usando Deep Learning*. https://doi.org/DOI: 10.13140/RG.2.2.22500.22409

Bernal Torres, C. (2016). *Metodología de la Investigación* (Cuarta). Pearson Educación de Colombia S.A.S. https://www.academia.edu/44228601/Metodologia_De_La_Investigaci%C3%B3n_Bernal_4ta_edicion

Campos Wright, W., & Trujillo Casañola, Y. (2021). Redes Neuronales Artificiales en la estimación del esfuerzo. *Revista Cubana de Ciencias Informáticas*, *15*(2), 183–198.

Cebrián Aldana, M. (2020). *Análisis de datos mediante técnicas de Deep Learning aplicadas a casos médicos* [Grado en Biotecnología, Universidad de León]. https://buleria.unileon.es/bitstream/handle/10612/13386/TFG_Biotecnologia_CebrianAldana_Miguel.pdf?sequence=1&isAllowed=y

Charte Luque, F. (2017). *Reducción de la dimensionalidad en problemas de clasificación con Deep Learning: Análisis y propuesta de herramienta en R* [Doble Grado en Ingeniería Informática y Matemáticas, Universidad de Granada]. https://doi.org/10.13140/RG.2.2.16155.57123/1

Cifuentes, A., Mendoza, E., Lizcano, M., Santrich, A., & Moreno-Trillos, S. (2019). Desarrollo de una red neuronal convolucional para reconocer patrones en imágenes. *Investigación y desarrollo en TIC*, *10*(2), 7–17.

CodeIgniter. (2022). *CodeIgniter*. https://www.codeigniter.com/

Colaboratory. (s/f). *Google Colaboratory*. Recuperado el 7 de julio de 2022, de https://colab.research.google.com/?hl=es#scrollTo=OwuxHmxllTwN

Contreras García, J., Molina Portillo, E., & Arteaga Cezón, P. (2011). *Introducción a la progrmación estadística con R para profesores*. https://www.ugr.es/~batanero/pages/ARTICULOS/libroR.pdf

Del Hoyo Dorado, E. (2020). *Diseño, implementación y evaluación de una red neuronal convolucional para la detección de puntos principales en naranjas* [Máster en Ingeniería Industrial, Universitat Politénica de

Valencia]. https://riunet.upv.es/bitstream/handle/10251/162073/72897452X_TFM_1600273505745680063200213477611O.pdf?sequence=1

Desarrollo Web. (s/f). *Laravel*. Recuperado el 9 de julio de 2022, de https://desarrolloweb.com/home/laravel

EcuRed. (2013). *Django*. https://www.ecured.cu/Django

EcuRed. (2019). *C++*. https://www.ecured.cu/C%2B%2B

Erroz Arroyo, D. (2019). *Visualizando neuronas en Redes Neuronales Convolucionales* [Grado en Ingeniería Informática, Universidad Pública de Navarra]. https://academica-e.unavarra.es/bitstream/handle/2454/33694/memoria_TFG.pdf?sequence=1&isAllowed=y

Espinosa, A., Espinosa, C., & Roberto, M. (2016). Comparativo de los Métodos de Mínimos Cuadrados y Eliminación de Gauss-Jordan para la Resolución de Sistema de Ecuaciones en el tema de Regresión Lineal. *Conciencia Tecnológica*, *52*, 42–50.

Fernández Lizana, M. (2020). Ventajas de R como herramienta para el Análisis y Visualización de datos en Ciencias Sociales. *Revista Científica de la UCSA*, *7*(2). https://doi.org/10.18004/ucsa/2409-8752/2020.007.02.097

Forbes. (2021). *Atención a estas diez tendencias tecnológicas para 2022*. Forbes Centroamérica • Información de negocios y estilo de vida para los líderes de Centroamérica y RD. https://forbescentroamerica.com/2021/07/22/atencion-a-estas-diez-tendencias-tecnologicas-para-2022/

Gallardo, E. (2017). *Metodología de la Investigación. Manual Autoformativo Interactivo* (Primera). Universidad Continental. https://repositorio.continental.edu.pe/bitstream/20.500.12394/4278/1/DO_UC_EG_MAI_UC0584_2018.pdf

García Peñalvo, F., García Holgado, A., & Vázquez Ingelmo, A. (2020). *Modelo de proceso*. https://repositorio.grial.eu/bitstream/grial/1940/1/IS_I%20Tema%203%20-%20Modelos%20de%20Proceso.pdf

García Sánchez, E. (2019). *Introducción a las redes neuronales de convolución. Aplicación a la visión por ordenador* [Trabajo de fin de grado en Matemáticas, Universidad de Zaragoza]. https://core.ac.uk/download/pdf/290002463.pdf

Gobierno de España. (2020). *¿Cómo aprenden las máquinas? Machine Learning y sus diferentes tipos*. datos.gob.es. https://datos.gob.es/es/blog/como-aprenden-las-maquinas-machine-learning-y-sus-diferentes-tipos

Goecks, J., Jalili, V., Heiser, L. M., & Gray, J. W. (2020). How Machine Learning Will Transform Biomedicine. *Cell*, *181*(1), 92–101. https://doi.org/10.1016/j.cell.2020.03.022

González, V. (2019). *Una Breve Historia del Machine Learning*. Think Big. https://empresas.blogthinkbig.com/una-breve-historia-del-machine-learning/

González-Islas, J. (2019). Aprendizaje Automático en Aplicaciones Fisioterapéuticas. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, *7*(Especial), 104–110. https://doi.org/10.29057/icbi.v7iEspecial.4473

Google Trends. (2022). *Comparar Lenguajes de programación*. Google Trends. https://trends.google.es/trends/explore?geo=EC&q=%2Fm%2F05z1_,%2Fm%2F0jgqg,%2Fm%2F0212jm

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (Sexta). McGraw-Hill Education. https://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf

Hidago, M., Hayes, B., Delgadillo, I., & Goyo, M. (2021). Deep learning aplicado para la detección de hemorragias y tumores cerebrales. *AtoZ: novas práticas em informação e conhecimento*. https://doi.org/10.5380/atoz.v11i.81284

Hinestroza, D. (2018). *El Machine Learning a través de los tiempos, y los aportes a la humanidad* [Trabajo de Grado - Ingeniería, Universidad Libre Seccional Pereira]. https://repository.unilibre.edu.co/bitstream/handle/10901/17289/EL%20MACHINE%20LEARNING.pdf?sequence=1&isAllowed=y

IBM. (s/f). *Inteligencia Artificial en la Medicina*. Recuperado el 2 de junio de 2022, de https://www.ibm.com/ar-es/topics/artificial-intelligence-medicine

IBM. (2021). *¿Qué es Machine Learning?* https://www.ibm.com/ar-es/analytics/machine-learning

IBM. (2022). *What is Machine Learning?* https://www.ibm.com/cloud/learn/machine-learning

Izquierdo-Valladarez, J., & Cuenca-Tapia, J. (2019). Reconocimiento de objetos del hogar, usando redes neuronales convolucionales para personas con discapacidad visual. *Polo del Conocimeinto*, *5*(01), 563–580. https://doi.org/10.23857/pc.v5i01.1238

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*, 685–695. https://doi.org/10.1007/s12525-021-00475-2

Jiménez Alfaro, A., & Díaz Ospina, J. (2021). Revisión sistemática de literatura: Técnicas de aprendizaje automático (Machine Learning). *Cuaderno Activa*, *13*(1), 113–121.

Jurado -Sánchez, M., Pedroza -Charris, E., & Rolón -Rodríguez, B. (2021). ¿Cómo ha ayudado la inteligencia artificial en la medicina? *Revista CONVICCIONES*, *8*(16), 6–20.

Kaggle. (s/f). *Brain MRI segmentation*. Recuperado el 2 de agosto de 2022, de https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation

Kaggle. (2022). *Kaggle: Your Machine Learning and Data Science Community*. https://www.kaggle.com/

Lakkavaram, V. R., Satish Kumar, C., Kumar, C., Sai Sri, G., & Habeeb, S. (2019). A review on Practical Diagnostic of Tomato Plant Diseases. *Suraj Punj Journal For Multidisciplinary Research*. https://www.researchgate.net/profile/Lakkavaram-Raghuveer-2/publication/332570921_A_review_on_Practical_Diagnostic_of_Tomato_Plant_Diseases/links/5cbea6aaa6fdcc1d49a872c0/A-review-on-Practical-Diagnostic-of-Tomato-Plant-Diseases.pdf

Laravel. (2022). *The PHP Framework For Web Artisans*. Laravel. https://laravel.com/

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE*, 1–21. https://doi.org/10.1109/TNNLS.2021.3084827

López, E., & Joa, L. (2017). Cadenas de Markov aplicadas al análisis de la ejecución de proyectos de investigación. *Revista Cubana de Informática Médica*, *9*(1), 44–51.

López, J. (2021). *Teorema de Bayes*. Economipedia. https://economipedia.com/definiciones/teorema-de-bayes.html

López Saca, F. (2019). *Clasificación de imágenes usando redes neuronales convolucionales* [Maestría en Ciencias de la Computación, Universidad Autónoma Metropolitana]. http://zaloamati.azc.uam.mx/bitstream/handle/11191/6123/Clasificacion_de_imagenes_Lopez_Saca_F_2019.pdf?sequence=1

Lozada-Portilla, W., Suarez-Barón, M., & Avendaño-Fernández, E. (2021). Aplicación de redes neuronales convolucionales para la detección del tizón tardío Phytophthora infestans en papa Solanum tuberosum.

U.D.C.A Actualidad & Divulgación Científica, *24*(2e1917). https://doi.org/10.31910/rudca.v24.n2.2021.1917

Main Yaque, P., Navarro Veguillas, H., & Morales Fernández, A. (2019). *Simulación con ejercicios en R* (Primera). Ediciones Complutense. https://bit.ly/3AxLMzn

Maisueche, A. (2019). *Utilización del Machine Learning en la Industria 4.0* [Máster en Ingeniería Industrial, Universidad de Valladolid]. https://uvadoc.uva.es/bitstream/handle/10324/37908/TFM-I-1372.pdf?sequence=1&isAllowed=y

Manjarrez, L. (2014). *Relaciones Neuronales Para Determinar la Atenuación del Valor de la Aceleración Máxima en Superficie de Sitios en Roca Para Zonas de Subducción* [Maestría en Ingeniería, Universidad Nacional Autónoma de México UNAM]. https://bit.ly/3yoxZIQ

Manrique Rojas, E. (2020). Machine Learning: Análisis de lenguajes de programación y herramientas para desarrollo. *Revista Ibérica de Sistemas e Tecnologias de Informação*, *E28*, 586–599.

MDN Web Docs. (2022). *Django introduction*. https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

MedlinePlus. (s/f). *Resonancia magnética de la cabeza*. Recuperado el 2 de junio de 2022, de https://medlineplus.gov/spanish/ency/article/003791.htm

Moreno Díaz-Alejo, L. (2020). *Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes* [Máster Universitario en Inteligencia Artificial, Universidad Internacional de La Rioja UNIR]. https://reunir.unir.net/bitstream/handle/123456789/10008/Moreno%20D%C3%ADaz-Alejo%2C%20Lara.pdf?sequence=1&isAllowed=y

Muñoz, N., & Romero, J. (2021). Optimización de los hiperparámetros de una máquina de regresión de soporte vectorial utilizando enjambre de partículas para el pronóstico de casos de COVID-19 en Bogotá.

Investigación e Innovación en Ingenierías, *9*(2), 91–111. https://doi.org/10.17081/invinno.9.2.4475

Pedrero, V., Reynaldos-Grandón, K., Ureta-Achurra, J., & Cortez-Pinto, E. (2021). Generalidades del Machine Learning y su aplicación en la gestión sanitaria en Servicios de Urgencia. *Revista Médica Chile*, *149*, 248–254.

Picazo Montoya, Ó. (2018). *Redes neuronales convolucionales profundas para el reconocimiento de emociones en imágenes* [Máster Universitario en Inteligencia Artificial, Universidad Politécnica de Madrid]. https://oa.upm.es/51441/1/TFM_OSCAR_PICAZO_MONTOYA.pdf

Pressman, R. S. (2013). *Ingeniería del software: Un enfoque práctico* (Séptima). McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V. http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Quintero, C., Merchán, F., Cornejo, A., & Sánchez-Galán, J. (2018). Uso de Redes Neuronales Convolucionales para el Reconocimiento Automático de Imágenes de Macroinvertebrados para el Biomonitoreo Participativo. *KnE Engineering*, 585–596. https://doi.org/10.18502/keg.v3i1.1462

Quiñones Huatangar, L., Ochoa Toledo, L., Kemper Valverde, N., Gamarra Torres, O., Bazán Correa, J., & Delgado Soto, J. (2020). Red neuronal artificial para estimar un índice de calidad de agua. *Enfoque UTE*, *11*(2), 109–120. https://doi.org/10.29019/enfoque.v11n2.633

R Foundation. (2022). *R: The R Project for Statistical Computing*. https://www.r-project.org/

Reynaldo Molina, Y. (2017). *Herramienta para el despliegue de software bajo la arquitectura Xalix del Centro FORTES* [Ttítulo de Ingeniero en Ciencias Informáticas].

https://repositorio.uci.cu/jspui/bitstream/123456789/9365/1/TD_08959_17.pdf

Rivas-Asanza, W., Mazon-Olivo, B., & Mejía-Peñafiel, E. (2018). Capítulo 1: Generalidades de las redes neuronales artificiales. En *Redes neuronales artificiales aplicadas al reconocimiento de patrones* (Primera, pp. 11–35). UTMACH. https://bit.ly/3uzHHH4

Rodríguez, I. (2020). *Efecto de la sustitución de la función de Pooling en Redes Neuronales Convolucionales* [Máster Universitario en Ingeniería Informática, Universidad Pública de Navarra]. https://academica-e.unavarra.es/bitstream/handle/2454/38674/Iosu%20Rodr%C3%ADguez%20Mart%C3%ADnez%20-%20TFM.pdf?sequence=1&isAllowed=y

Rodríguez, J. (2018). *Aplicación de técnicas de Machine Learning a la detección de ataques* [Trabajo de Fin de Máster MISTIC, Universitat Oberta de Catalunya]. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/81126/11/jmrodriguez85TFM0618memoria.pdf

Sáez de la Pascua, A. (2019). *Deep learning para el reconocimiento facial de emociones básicas* [Grado en Ingeniería de Sistemas de Telecomunicaciones, Universitat Politénica de Catalunya]. https://upcommons.upc.edu/bitstream/handle/2117/129220/memoria.pdf?sequence=1&isAllowed=y

Sandoval, L. (2018). Algoritmos de aprendizaje automático para análisis y predicción de datos. *ITCA FEPADE Tecnológicos e Ingenieros*, *11*. http://redicces.org.sv/jspui/bitstream/10972/3626/1/Art6_RT2018.pdf

Santander Universidades. (2020). *Metodologías de desarrollo de software: ¿qué son?* https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html

Sierra-García, J. E., & Santos, M. (2021). Redes neuronales y aprendizaje por refuerzo en el control de turbinas eólicas. *Revista Iberoamericana*

de Automática e Informática Industrial, *18*(4), 327–335. https://doi.org/10.4995/riai.2021.16111

Sommerville, I. (2011). *Ingeniería de software* (Novena). Pearson Educación de México, S.A. de C.V. https://elibro.net/ereader/elibrodemo/37857

Syntonize. (2021). *Tecnología en 2022: Tendencias e innovaciones*. Syntonize. https://www.syntonize.com/tendencias-de-la-tecnologia-en-2022/

Universia. (2020). *Para qué sirve Python: Qué es y usos*. https://www.universia.net/es/actualidad/orientacion-academica/para-que-sirve-phyton-que-es-y-usos-1154393.html

Universitat Carlemany. (2020). *Metodologías de desarrollo de software*. https://www.universitatcarlemany.com/actualidad/metodologias-de-desarrollo-de-software

Vannieuwenhuyze, A. (2020). *Inteligencia artificial fácil. Machine Learning y Deep Learning prácticos*. Ediciones ENI. https://www.ediciones-eni.com/libro/inteligencia-artificial-facil-machine-learning-y-deep-learning-practicos-9782409025327/extracto-del-libro.pdf

Varela Arregocés, E., & Campbells Sánchez, E. (2011). Redes Neuronales Artificiales: Una Revisión del Estado del Arte, Aplicaciones Y Tendencias Futuras. *Investigación y Desarrollo en TIC*, *2*(1). https://core.ac.uk/download/pdf/267928931.pdf

Visus, A. (2020). *¿Para qué sirve Python? Razones para utilizarlo*. Python. https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

Yu, H., Zheng, J., & Lin, Q. (2022). Strength prediction of seawater sea sand concrete based on artificial neural network in python. *Materials Research Express*, *9*(3). https://doi.org/10.1088/2053-1591/ac5957

Yuvaraj, K., Oorappan, G. M., Megavarthini, K. K., Pravin, M. C., Adharsh, R., & Kumaran, M. A. (2020). Design And Development Of An Application For Database Maintenance In Inventory Management

System Using Tkinter And Sqlite Platform. *IOP Conference Series. Materials Science and Engineering*, *995*(1). https://doi.org/10.1088/1757-899X/995/1/012012

Zumba Gamboa, J., & León Arreaga, C. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal*, *3*(10), 20–33.

ANEXOS

Formato de entrevista

Objetivo: Conocer el punto de vista de un experto y validar algunas imágenes que forman parte del conjunto de las imágenes para entrenamiento.

- **¿Cuál es el procedimiento para detectar un tumor cerebral?**
- **¿Considera usted que los factores raza(sexo) puedan afectar la detención de los tumores cerebrales?**
- **¿Considera interesante el desarrollo un modelo predictivo para la detección de tumores?**
- **¿Cree usted que una herramienta tecnología podría agilizar el proceso de detección de tumores?**

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Castellano Sánchez Kevin Lester**, con C.C: # **0931526289** autor/a del trabajo de titulación: "**Modelo Machine Learning para la detección de tumores cerebrales mediante un conjunto de imágenes de resonancias magnéticas"** previo a la obtención del título de **Ingeniero en Sistemas Computacionales** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 15 de septiembre del 2022

_____

Nombre: **Castellano Sánchez Kevin Lester**

C.C: **0931526289**

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA			
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN			
TEMA Y SUBTEMA:	**Modelo Machine Learning para la detección de tumores cerebrales mediante un conjunto de imágenes de resonancias magnéticas**		
AUTOR(ES)	**Kevin Lester, Castellano Sánchez**		
REVISOR(ES)/TUTOR(ES)	**Ing. Mario Colon, Célleri Mujica**		
INSTITUCIÓN:	**Universidad Católica de Santiago de Guayaquil**		
FACULTAD:	**Ingeniería**		
CARRERA:	**Ingeniería en Sistemas Computacionales**		
TITULO OBTENIDO:	**Ingeniero en Sistemas Computacionales**		
FECHA DE PUBLICACIÓN:	**15 de septiembre del 2022**	**No. DE PÁGINAS:**	80
ÁREAS TEMÁTICAS:	**Machine learning**		
PALABRAS CLAVES/ KEYWORDS:	Machine Learning, red neuronal convolucional, aprendizaje por transferencia, Deep Learning.		

RESUMEN/ABSTRACT:

El Machine Learning se utiliza en algunos campos del conocimiento, como la medicina, por su predicción de la eficacia en tratamientos de enfermedades y en la toma de decisiones luego de la identificación de patrones. Esto permite crear herramientas de soporte para diagnóstico, como el Modelo ML para la detección de tumores cerebrales mediante resonancias magnéticas que se propone, por medio del cual se permitirá la automatización de dicho proceso, para agilizar el diagnóstico del médico tratante. Se conoció sobre las redes neuronales, el aprendizaje por transferencia, la arquitectura ResNet 50, Deep Learning, además de las redes neuronales convolucionales. Se utilizó la metodología descriptiva con enfoque cualitativo y entrevista a profesionales médicos, además del prototipado evolutivo como metodología de desarrollo. Se conoció que la tecnología tiene relevancia en la medicina, y que sí es conveniente implementar una herramienta que sirva de soporte para la toma de decisiones en cuanto a diagnóstico de tumores cerebrales. De la evaluación del prototipo se obtuvo una tasa de aciertos aceptable para el desarrollo de la herramienta. Al finalizar, se propusieron algunas conclusiones y recomendaciones a considerar para posibles trabajos futuros.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:**+593-992300295	E-mail: **kevin.castellano@cu.ucsg.edu.ec**
CONTACTO CON LA INSTITUCIÓN (C00RDINADOR DEL PROCESO UTE)::	**Toala Quimí, Edison José**	
	Teléfono: +593-990-976776	
	E-mail: edison.toala@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	