



Universidad de Castilla-La Mancha
Escuela Superior de Ingeniería Informática

Trabajo Fin de Grado
Grado en Ingeniería Informática
Tecnología Específica en Computación

Detección de embolismos pulmonares mediante técnicas de aprendizaje automático

Pablo Palacios López

15 de enero de 2024



Trabajo Fin de Grado

Grado en Ingeniería Informática

Tecnología Específica en Computación

Detección de embolismos pulmonares mediante técnicas de aprendizaje automático

Autor: Pablo Palacios López

Tutor: José Antonio Gámez Martín , Juan Carlos Alfaro Jiménez

15 de enero de 2024

*Si tienes un sueño, sacrifícate por ello.
y nunca, jamás, descanses en mitad del proceso.*

Declaración de autoría

Yo, *Pablo Palacios López*, con DNI *49429183K*, declaro que soy el único autor del trabajo fin de grado titulado “*Detección de embolismos pulmonares mediante técnicas de aprendizaje automático*”, que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual, y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 15 de enero de 2024

Fdo.: Pablo Palacios López

Resumen

Los embolismos pulmonares (PE) son los responsables de entre 100.000 y 200.000 muertes al año aproximadamente solamente en los Estados Unidos debido a que el diagnóstico de este tipo de enfermedad es todo un reto [14].

Esta enfermedad consiste en la formación de un coágulo de sangre que se desarrolla en un vaso sanguíneo (normalmente se produce en la pierna) viajando a través de la arteria pulmonar donde se produce el bloqueo del flujo sanguíneo. Actualmente, esta clase de enfermedad no presenta síntomas y señales específicas, es por ello por lo que es muy complicado su detección y su diagnóstico, siendo la tercera causa de muertes cardiovasculares en todo el mundo [18].

El aprendizaje automático o *Machine Learning* puede ayudar a identificar de manera más precisa estos casos de embolismos pulmonares, permitiendo un mayor control y tratamiento sobre los pacientes. *The Radiological Society of North America* (RSNA) junto con *The Society of Thoracic Radiology* (STR) son organizaciones que se encargan de buscar la mejora en la detección a través del uso de Angiografías Pulmonares por Tomografía Computerizada (CTPA).

Este Trabajo de Fin de Grado pretenderá llevar a cabo la detección y clasificación de embolias pulmonares mediante técnicas de aprendizaje automático gracias a un conjunto de CTPAs proporcionadas por RSNA y STR. También llevaremos a cabo una aplicación en la que el usuario pueda introducir una CTPA y pueda detectar si existe o no un embolismo pulmonar, su localización y, en su caso, la gravedad de la enfermedad.

Agradecimientos

Primero de todo, quiero agradecerle a mis padres, Virgilio y Mercedes, su increíble esfuerzo. Por quererme, cuidarme, motivarme a seguir cuando todo se tornaba en mi contra, sobre todo, una vez más, por haber hecho mis sueños realidad. Sin ellos, no habría conseguido nada de lo que tengo hoy en día.

A mi hermana, Andrea, que te he visto crecer desde el día 1 que llegaste, te has convertido en toda una referente para mí. Yo seré el mayor de los dos, pero tú tienes el corazón más grande.

Quiero continuar agradeciéndoselo a mis tutores, José Antonio Gámez Martín y Juan Carlos Alfaro Jiménez. Siempre pendientes de los problemas que me han surgido durante este trabajo, proporcionándome una gran ayuda cuando más lo necesitaba.

A mis amigos, a todos y cada uno de ellos que han estado conmigo en clase o compartiendo momentos inolvidables. Juan Ramón, Javi, Alex, José Jesús, Álvaro, Campa, Juan Carlos y Lucas, no os olvidaré nunca, sois un pilar fundamental en mi vida.

Pero sobre todo, darle las gracias al mejor amigo y compañero que se puede tener, Jaime Parada. Sin ti, estos 4 años no habrían sido igual, siempre has estado ahí cuando más lo he necesitado. Nunca olvidaré todo lo que hemos pasado juntos, tantas clases, trabajos y muchas cosas más que me llevo al corazón. Me has demostrado ser la mejor persona del mundo y el mejor ingeniero que este precioso campo puede tener, siempre ambicioso a querer conseguir más, motivándome cada día y ser mi mano derecha durante este largo camino. Espero que esto sea un punto y seguido en nuestras vidas, que continuemos riendo y disfrutando de la forma que lo hemos hecho durante todo este tiempo, esto va por ti. *Job not finished.*

Por último, al baloncesto, un amor tan profundo que mi mente, mi cuerpo, mi espíritu y alma son tuyos. Siempre me verás ser aquel niño feliz botando un simple balón.

Muchas gracias a todos.

Índice general

1 Introducción	1
1.1 Motivación	2
1.2 Objetivos.....	2
1.3 Competencias abordadas.....	3
1.4 Estructura del documento.....	3
2 Estado del arte.....	5
2.1 Embolismos pulmonares	5
2.2 Aprendizaje automático	8
2.3 Redes neuronales	10
2.3.1 <i>El perceptrón</i>	11
2.3.2 <i>Perceptrón multicapa</i>	11
2.3.3 <i>Redes convolucionales</i>	12
2.4 Evaluación y validación.....	14
2.4.1 <i>Clasificación binaria</i>	14
2.5 Métricas de evaluación	16
2.6 Cross Industry Standard Process for Data Mining (CRISP-DM)	19
3 Estudio de los datos	21
3.1 Introducción	21
3.2 Metadatos de los pacientes	22
3.2.1 <i>Variable "negative_exam_for_pe"</i>	24
3.2.2 <i>Variable "indeterminate"</i>	25

3.2.3	<i>Variable "qa_motion"</i>	25
3.2.4	<i>Variable "qa_contrast"</i>	26
3.2.5	<i>Variable "pe_present_on_image"</i>	26
3.2.6	<i>Variables "rightsided_pe", "leftsided_pe" y "central_pe"</i>	27
3.2.7	<i>Variables "rv_lv_ratio_gte_1" y "rv_lv_ratio_lt_1"</i>	29
3.2.8	<i>Variables "chronic_pe" y "acute_and_chronic_pe"</i>	30
3.3	Análisis de imágenes	30
4	Preprocesamiento de los datos	33
4.1	Introducción	33
4.2	Flujo de preprocesamiento de las imágenes	33
4.2.1	<i>Filtro de realce de bordes de Sobel</i>	34
5	Creación del modelo	41
5.1	Introducción	41
5.2	Recursos hardware y conjunto de datos	41
5.3	Redes neuronales	42
5.3.1	<i>Diseño propio de una CNN Básica</i>	43
5.3.2	<i>Transfer Learning</i>	48
5.4	Conclusiones y futuras mejoras	58
6	Desarrollo de la aplicación	61
6.1	Introducción	61
6.2	Diseño de la aplicación	61
6.3	Flask	62
6.4	Gradient-weighted Class Activation Mapping	62
6.5	Aplicación web	63
6.6	Conclusiones y futuras mejoras	66
7	Conclusiones	69
7.1	Conclusiones y trabajos futuros	69
7.2	Competencias adquiridas	70
A	Planificación del proyecto	71
B	Recursos Software	73
C	Recursos Hardware	75

Referencia bibliográfica	80
--------------------------------	----

Índice de figuras

2.1	Trombosis Venosa Profunda	6
2.2	CTPA	6
2.3	Información mostrada en una CTPA	7
2.4	CTPA con embolia pulmonar	8
2.5	Aprendizaje Supervisado	9
2.6	Aprendizaje No Supervisado	9
2.7	Aprendizaje Por Refuerzo	10
2.8	Perceptrón	11
2.9	Perceptrón Multicapa	12
2.10	Operación de convolución	12
2.11	Ejemplo de CNN con múltiples capas convolucionales	14
2.12	Conjunto de entrenamiento desbalanceado	15
2.13	Conjunto de entrenamiento balanceado	16
2.14	Área bajo la curva ROC	17
2.15	Matriz de confusión	17
2.16	Fórmula del Recall	18
2.17	Fórmula del accuracy	18
2.18	Fórmula de la precisión	19
2.19	Fases de la metodología CRISP-DM Process Model for Data Mining	19
3.1	Diagrama de relación de variables	24
3.2	Distribución de la variable <i>negative_exam</i>	25
3.3	Distribución de la variable <i>indeterminate</i>	25
3.4	Distribución de la variable <i>qa_motion</i>	26
3.5	Distribución de la variable <i>qa_contrast</i>	26
3.6	Distribución de la variable <i>pe_present_on_image</i>	27
3.7	Localización de la embolia	28
3.8	Ejemplo de embolia pulmonar en la parte derecha, izquierda y central .	28
3.9	Ejemplo gráfico de disposición de los ventrículos	29

3.10	Distribución de la variable radio del ventrículo derecho hasta el ventrículo izquierdo	29
3.11	Gravedad de la embolia	30
3.12	Imagen en formato DICOM	31
3.13	Imagen en formato JPG	31
4.1	3x3 kernels del filtro de Sobel	34
4.2	Imagen resultante de aplicar el Kernel G_x	35
4.3	Imagen resultante de aplicar el Kernel G_y	36
4.4	Matriz de la magnitud del gradiente	37
4.5	Matriz resultante del filtro de Sobel	38
4.6	Ejemplo del filtro de Sobel.	38
4.7	Ejemplo del filtro de Sobel.	39
5.1	Arquitectura CNN propia.	45
5.2	Arquitectura VGG19	49
5.3	Arquitectura Xception.	52
5.4	Arquitectura modelo de Transfer Learning + XGBoost.	56
6.1	Grad-CAM de un perro y un gato	63
6.2	Interfaz de la aplicación	64
6.3	Mensaje de error.	64
6.4	Interfaz de la aplicación con una imagen que presenta embolia.	65
6.5	Interfaz de la aplicación con una imagen que no presenta embolia.	66
A.1	Planificación del proyecto.	72

Índice de tablas

3.1	Contenido ubicado en la competición de Kaggle " <i>RSNA STR Pulmonary Embolism Detection</i> "	22
3.2	Contenido <i>train.csv</i>	23
5.1	Entrenamiento modelos simples	47
5.2	Validación modelos simples	47
5.3	Evaluación modelos simples	47
5.4	Entrenamiento modelos VGG19.	50
5.5	Validación modelos VGG19.	50
5.6	Evaluación modelos VGG19.	50
5.7	Entrenamiento de los modelos Xception y SeXception.	53
5.8	Validación de los modelos Xception y SeXception.	53
5.9	Validación de los modelos Xception y SeXception.	53
5.10	Entrenamiento de los modelos ResNet50.	55
5.11	Validación de los modelos ResNet50.	55
5.12	Validación de los modelos ResNet50.	55
5.13	Entrenamiento de los modelos ResNet50 + XGBoost.	57
5.14	Validación de los modelos ResNet50 + XGBoost.	58
5.15	Evaluación de los modelos ResNet50 + XGBoost.	58
5.16	Resumen mejores modelos tras la fase de evaluación.	59
5.17	Mejor modelo ResNet50 + XGBoost tras la fase de evaluación.	59

Índice de listados de código

5.1 Código de <i>Squeeze-and-Excitation</i> block	52
---	----

Capítulo 1. Introducción

La trombosis representa la formación de un trombo¹ en el sistema circulatorio dentro de un vaso sanguíneo o en el corazón, siendo una de las mayores causas de mortalidad en muchos países. A pesar de la gran cantidad de técnicas de prevención para este tipo de enfermedad, se siguen diagnosticando cada vez más casos, por ello los médicos recomiendan que un estilo de vida saludable es la mejor forma de prevención.

La trombosis ocurre cuando se forman coágulos de sangre potencialmente mortales en las arterias (trombosis arterial) o en las venas (trombosis venosa). La formación de un coágulo puede ralentizar o bloquear la circulación de la sangre normal e incluso desprenderse y trasladarse hacia un órgano, en cuyo caso se denomina "embolia". De todos los tipos presentes la más común y conocida es la Trombosis Venosa Profunda (TVP) [6].

Aproximadamente más de 10 millones de casos de TVP son diagnosticados anualmente, un millón de casos ocurren en los Estados Unidos, y 700.000 casos totales entre Francia, España, Italia, Alemania, Suecia y Reino Unido [31].

La complicación más grave debido a esta Trombosis Venosa Profunda es la que conocemos como tromboembolismo o Embolia Pulmonar (PE). En este caso el coágulo viajará hasta las arterias pulmonares interrumpiendo el transcurso de la sangre. Aquellos pacientes con estados específicos de enfermedad como, trastornos en la coagulación de la sangre, cáncer, parálisis e hipertensión, tienen un mayor riesgo de sufrir un embolismo pulmonar. Siendo una de las categorías más peligrosas de trombosis venosas, con una alta tasa de mortalidad si no se diagnostica bien y no se trata [2].

¹**Trombo:** Del griego thrombos (grumo, coágulo), es un cuerpo sólido integrado por plaquetas y fibrina, que se forma a causa de una lesión endotelial, un retardo de la corriente sanguínea o una alteración de la composición química de la sangre.

1.1. Motivación

La motivación de este trabajo viene dada por la gran cantidad de aplicaciones del mundo de la informática en un campo tan importante como es el de la medicina. Desde los orígenes de la informática hasta el día de hoy, se han producido grandes avances en esta área, como facilitar el análisis de las secuencias genéticas para la producción de vacunas, e incluso en el campo de la Obstetricia, proporcionando un mejor control y observación del feto en desarrollo.

Actualmente, nos encontramos en el punto álgido de la tecnología y todo lo que podemos aprender de ella nos beneficiará como especie. Lo que se pretende realizar en este Trabajo de Fin de Grado es ayudar o facilitar el trabajo de los médicos a detectar tromboembolismos pulmonares debido a la considerable dificultad de esta tarea. Para ello se llevará a cabo el uso del *machine learning*, el análisis de imágenes y la ciencia de datos. Basándonos en el conjunto de datos liberado por *The Radiological Society of North America* (RSNA) junto con *The Society of Thoracic Radiology* (STR) de los Estados Unidos de América y recogido en una competición de la plataforma Kaggle.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado es estudiar y clasificar las distintas imágenes que tenemos en nuestro conjunto de datos, proporcionando una mayor facilidad tanto para la detección de embolias pulmonares como para su tratamiento. Por último, los mejores modelos obtenidos en el estudio de ciencia de datos realizado se incorporarán a una pequeña aplicación de escritorio donde poder introducir nuestras imágenes y que nos proporcione un resultado. Por tanto, nos propondremos alcanzar los siguientes objetivos:

- **Realizar un estudio de los datos disponibles** (imágenes y metadatos de los pacientes).
- **Definir** distintas **estrategias de predicción**, desde técnicas de *Deep Learning* para las imágenes proporcionadas, como métodos clásicos de aprendizaje automático (para los metadatos de los pacientes).
- **Definir** las **estrategias de validación y selección** del mejor modelo.
- **Desarrollo** de una **aplicación para predecir y obtener resultados interpretables** para los médicos.

Todos los datos que se han usado para llevar a cabo este Trabajo de Fin de Grado han sido facilitados por *The Radiological Society of North America* (RSNA) junto con *The Society of Thoracic Radiology* (STR). Dichos datos se pueden encontrar en la página oficial de la competición [23] o en la competición organizada en Kaggle [21].

1.3. Competencias abordadas

Este Trabajo de Fin de Grado se enmarca en la intensificación de *Computación*, por ello se trabajarán en las siguientes competencias:

- Capacidad para **evaluar la complejidad computacional** de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.
- Capacidad para **conocer los fundamentos**, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.
- Capacidad para **adquirir, obtener, formalizar y representar el conocimiento humano** en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes entornos inteligentes.
- Capacidad para **conocer y desarrollar técnicas de aprendizaje computacional** y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

1.4. Estructura del documento

Este trabajo está constituido por las siguientes partes:

1. **Introducción y contexto.** Presentamos el trabajo a desarrollar, se realizará un estudio y análisis sobre las imágenes de las que disponemos (Capítulo 2).
2. **Desarrollo del modelo.** Se realizará un ciclo de vida completo en los procesos actuales de minería de datos siguiendo la metodología CRISP-DM. Llevaremos a cabo desde un estudio de los datos (Capítulo 3) disponibles hasta la experimentación con modelos de clasificación (Capítulo 5).
3. **Desarrollo de una aplicación.** En esta tercera parte mostraremos el proceso que hemos utilizado para realizar una aplicación que actúe de interfaz para uno de los modelos de clasificación que llevemos a cabo (Capítulo 6).
4. **Conclusiones del proyecto.** Finalmente, se extraerán las conclusiones obtenidas del proyecto (Capítulo 7).
5. **Anexos.** Parte complementaria del proyecto que contendrá información de la planificación seguida (Anexo A), los recursos software (Anexo B) y hardware (Anexo C) usados en el proyecto.

Capítulo 2. Estado del arte

En este capítulo introduciremos brevemente el problema desde un punto de vista del dominio médico sobre los embolismos pulmonares, como se producen, y en que pueden derivar. Después revisaremos las técnicas de aprendizaje automático que se usarán en el desarrollo del Trabajo de Fin de Grado.

2.1. Embolismos pulmonares

En primer lugar, vamos a describir como se originan las embolias pulmonares.

Un coágulo de sangre es una masa que se forma cuando las plaquetas, proteínas y células de la sangre se pegan entre sí. Cuando se produce una herida, el cuerpo humano tiende a formar coágulos de sangre para impedir que la sangre salga al exterior. Una vez que se detiene el sangrado, el cuerpo descompone y elimina el coágulo. A veces se producen coágulos anormales dentro de nuestro cuerpo o incluso no es capaz de descomponerlos como debería [20].

Existen dos tipos de coágulos de sangre:

- **Coágulos arteriales:** Formados en las arterias y causan síntomas inmediatamente, dando lugar a ataques cardíacos, ataques cerebrales o parálisis.
- **Coágulos venosos:** Se producen en las venas y se forman lentamente durante un periodo de tiempo.

Los coágulos de sangre se pueden formar en los vasos sanguíneos o viajar desde ellos hasta las extremidades, los pulmones, el cerebro, el corazón y los riñones. Dependiendo de donde se originen dichos coágulos pueden causar distintos tipos de problemas. En este caso nos centraremos en los coágulos venosos, en especial en la Trombosis Venosa Profunda (TVP).

La Trombosis Venosa Profunda (TVP) o *Deep Vein Thrombosis* (DVT) es un tipo de coágulo venoso que se forma en una vena profunda del cuerpo. Suele ocurrir en las piernas o muslos, como vemos en la Figura 2.1. Una Trombosis Venosa Profunda puede desprenderse y viajar por el torrente sanguíneo, causando un problema serio en los pulmones, debido al bloqueo que se puede producir en una de sus arterias. Esto es lo que conocemos como embolia pulmonar.

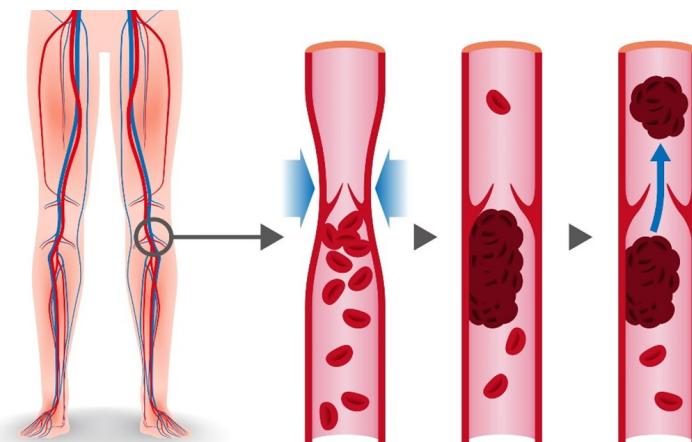


Figura 2.1: Trombosis Venosa Profunda

Una vez conocemos como se producen las embolias pulmonares, las identificaremos a través del uso de Angiografías Pulmonares por Tomografía Computerizada (APTC) o *Computed Tomography Pulmonary Angiogram* (CTPA/CTPE). Como podemos observar en la Figura 2.2, esta metodología utiliza Rayos X para crear imágenes transversales del tórax y la parte superior del abdomen.



Figura 2.2: Ejemplo de CTPA indicando que existe un coágulo

Es importante recordar que los embolismos pulmonares no afectan directamente a los pulmones, sino los vasos sanguíneos que transportan la sangre son los afectados. Como vemos en la Figura 2.3, esto es exactamente lo que nos muestra un CTPA. Observamos ambos pulmones y el corazón. Las arterias pulmonares están mostradas en color azul, desde ambos lados del pecho hasta la parte superior del corazón. Mientras que las venas son las que están mostradas en rojo, encargadas de transportar sangre al corazón.

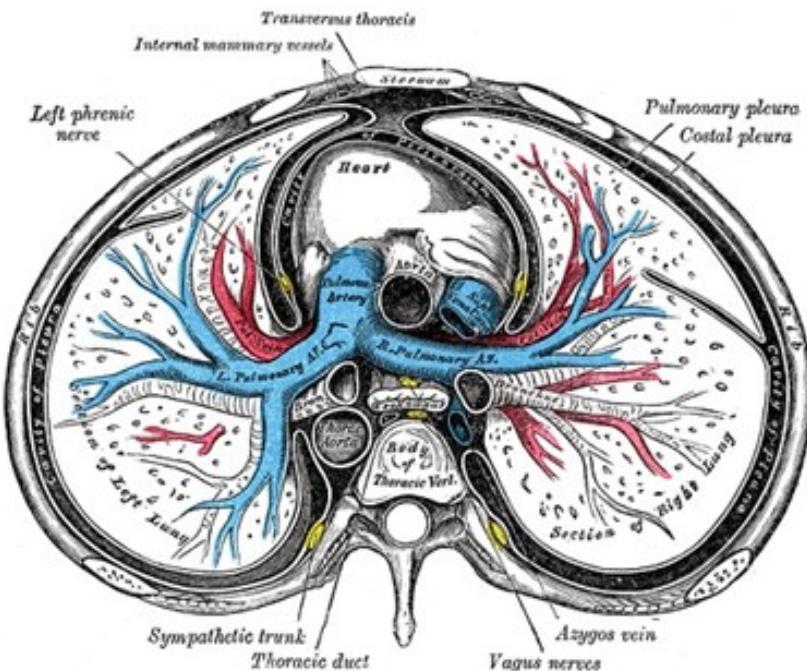


Figura 2.3: Información mostrada en una CTPA

Como vemos en la Figura 2.4, a simple vista no somos capaces de observar donde se encuentra el tipo de coágulo, se necesitan muchas horas observando los distintos fotogramas de las imágenes para identificar si el paciente tiene algún tipo de coágulo en alguna arteria pulmonar de su cuerpo [12], es por ello que la aportación de las técnicas de Aprendizaje Automático a este problema y a esta disciplina en general, se antojan imprescindibles.



Figura 2.4: CTPA con embolia pulmonar

2.2. Aprendizaje automático

El Aprendizaje Automático o *Machine Learning* es un subcampo de las ciencias de la computación y se considera una disciplina dentro del campo de la Inteligencia Artificial con la que, a través de algoritmos matemáticos, podemos dotar a los ordenadores de la capacidad de identificar patrones en datos masivos para realizar posteriormente predicciones. De esta manera, un ordenador adquiere el talento para distinguir entre distintos datos que se le pasen, para etiquetarlos, además de predecir comportamientos y tomar decisiones [9].

Existen numerosas aproximaciones que se pueden utilizar para abordar un problema de *Machine Learning*, pero normalmente se dividen en los siguientes paradigmas del aprendizaje:

- **Aprendizaje Supervisado:** Agrupa aquellos algoritmos en la que los datos de entrada contienen la salida esperada. Estos datos son etiquetados normalmente por el humano y el objetivo es aprender una regla general que, dada una entrada, le asigne una salida [38]. Un ejemplo de tipos de algoritmos que pertenecen a este grupo son los de clasificación, como vemos en la Figura 2.5.

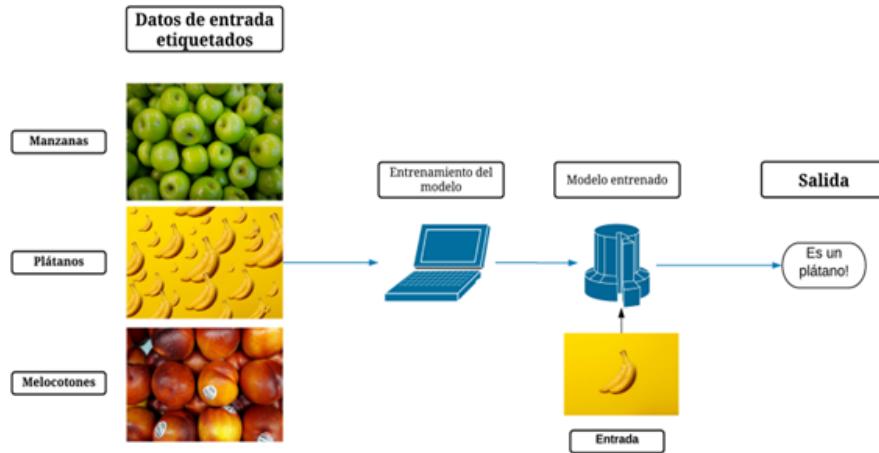


Figura 2.5: Aprendizaje Supervisado

Los tipos de problemas más estudiados por este subcampo son:

- **Clasificación:** Se usa cuando el resultado deseado es una etiqueta discreta. A cada nueva entrada se le asignará a la salida del sistema una de las clases o categorías con las que contamos en el problema. Se puede contar con 2 o más clases o categorías.
- **Regresión:** Se usa para predecir valores que son continuos. Para cada nueva entrada se le proporcionará a la salida del sistema un valor continuo.
- **Aprendizaje No Supervisado:** Agrupa aquellos algoritmos en la que los datos de entrada no contienen la salida. Funcionan buscando patrones en los datos de entrenamiento, haciendo agrupaciones de estos, para posteriormente cambiar en base de la presencia o la ausencia de puntos en común en cada nuevo dato [17]. Un esquema de este tipo se puede ver en la Figura 2.6.

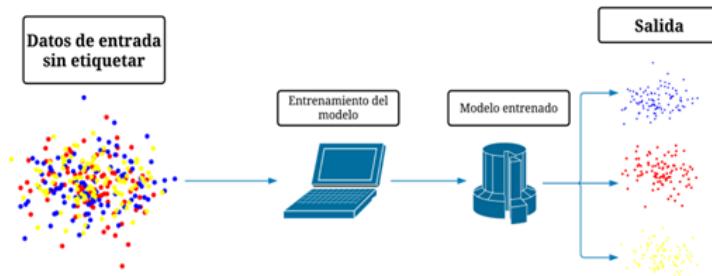


Figura 2.6: Aprendizaje No Supervisado

-
- **Aprendizaje por Refuerzo:** Agrupa aquellos algoritmos en los que, en vez de partir de unos datos de entrada dados, utilizan la información que reciben del entorno que les rodea. Su esquema de funcionamiento se puede ver en la Figura 2.7, donde se observa que el agente selecciona la mejor acción y el entorno le devuelve un premio o recompensa en función de lo buena que sea la acción.

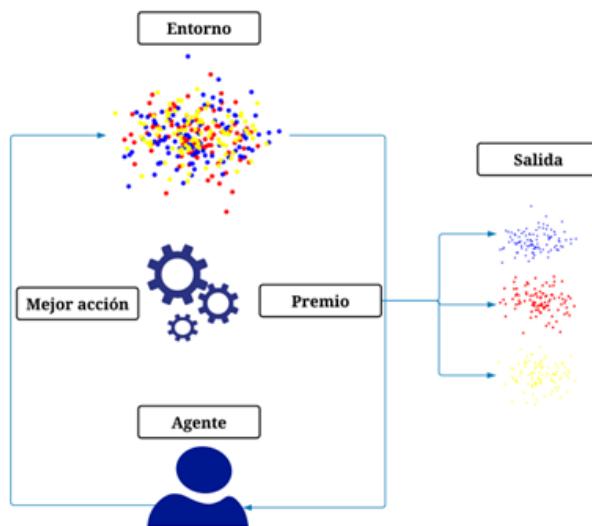


Figura 2.7: Aprendizaje Por Refuerzo

Dado que el objetivo de este trabajo es la detección de embolismos pulmonares, nos centraremos en usar algoritmos de aprendizaje automático para problemas de clasificación. Hoy en día, el tratamiento de imágenes se centra en utilizar modelos basados en redes neuronales [10].

2.3. Redes neuronales

Las redes neuronales tratan de emular el comportamiento del cerebro humano, caracterizado por el aprendizaje a través de la experiencia y la extracción del conocimiento genérico a través de un conjunto de datos.

El primer modelo de red neuronal surge en 1943, de la mano de Warren McCulloch y Walter Harry Pitts siendo un modelo básico y que no se llegó a implementar físicamente debido a las limitaciones técnicas de la época [33].

En 1958, Rosenblatt elabora el modelo de perceptrón, con una salida binaria y permitiendo variar sus pesos. Sin embargo, el *Perceptrón* es incapaz de resolver problemas que no presentan una solución lineal, como sería el ejemplo de la **puerta XOR**.

Pero no es hasta 1986, con la publicación del algoritmo “*Backpropagation*” de David Rumelhart, Geoffrey Hinton, y Ronald Williams, cuando se produce el resurgir de las redes neuronales artificiales. El cual presentaba una forma de aprendizaje para las redes neuronales, mediante la retropropagación de errores y haciendo uso del gradiente descendiente [33].

2.3.1. El perceptrón

El perceptrón fue el primero modelo de una red neuronal artificial. Su estructura es sencilla, tal y como podemos ver en la Figura 2.8, posee varias entradas (x_1, \dots, x_{n-1}, x_n) con sus respectivos pesos (w_1, \dots, w_{n-1}, w_n) y se realizará una suma ponderada de dichos valores. Su resultado pasará por una función de activación y proporcionará la salida del perceptrón.

Pero como hemos comentado anteriormente, los perceptrones solo son capaces de realizar particiones lineales y están limitados por ello [11].

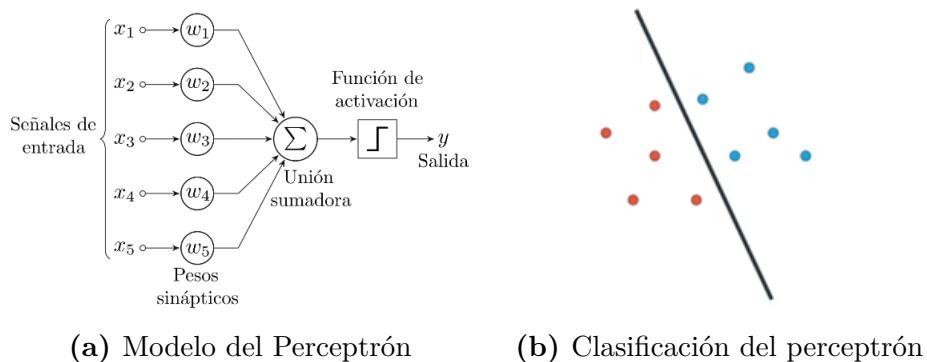


Figura 2.8: Perceptrón

2.3.2. Perceptrón multicapa

Un modelo de perceptrón multicapa tiene una estructura similar a la de un modelo de perceptrón, solamente que esta vez posee al menos 3 capas. En la Figura 2.9 vemos un ejemplo, la capa que recibe los datos recibe el nombre de **capa de entrada**, la que da los resultados de la red se llama **capa de salida** y la capa o capas intermedias entre

ambas se llaman **capas ocultas** pues no tienen contacto directo ni con los datos ni con los resultados, permaneciendo invisibles desde fuera de la red neuronal [33].

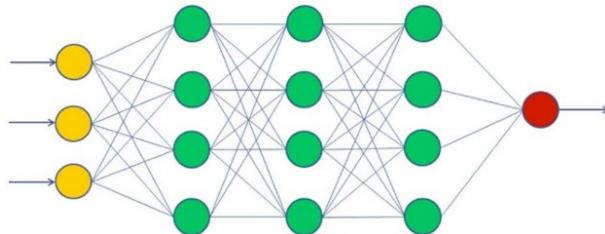


Figura 2.9: Perceptrón Multicapa

2.3.3. Redes convolucionales

Es un tipo de red neuronal artificial que surgió de la mano de Yann Lecun en 1998. Una red neuronal convolucional o CNN tendrá como datos de entrada imágenes en forma de matrices, donde cada valor representa un píxel [33].

Usar imágenes como datos de entrada presenta varias complicaciones, debido a que no solo se deben tener en cuenta los píxeles de las imágenes, sino su contexto espacial, es decir, los píxeles vecinos. Para ello usamos una operación matemática llamada convolución.

Esta operación consiste en tomar “grupos de píxeles” cercanos de la imagen de entrada e ir operando matemáticamente (Producto escalar) contra una pequeña matriz que se llama Kernel o Filtro y finalmente obtener una nueva imagen, como podemos observar en la Figura 2.10.

$$\begin{array}{|c|c|c|c|c|c|} \hline
 3 & 0 & 1 & 2 & 7 & 4 \\ \hline
 1 & 5 & 8 & 9 & 3 & 1 \\ \hline
 2 & 7 & 2 & 5 & 1 & 3 \\ \hline
 0 & 1 & 3 & 1 & 7 & 8 \\ \hline
 4 & 2 & 1 & 6 & 2 & 8 \\ \hline
 2 & 4 & 5 & 2 & 3 & 9 \\ \hline
 \end{array}
 \quad *
 \quad
 \begin{array}{|c|c|c|} \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 \end{array}
 \quad =
 \quad
 \begin{array}{|c|c|c|c|} \hline
 -5 & -4 & 0 & 8 \\ \hline
 -10 & -2 & 2 & 3 \\ \hline
 0 & -2 & -4 & -7 \\ \hline
 -3 & -2 & -3 & -16 \\ \hline
 \end{array}$$

Imagen Filtro Resultado

Figura 2.10: Operación de convolución

Las primeras capas aprenden características simples, como bordes y texturas, mientras que las capas posteriores aprenden características más complejas y abstractas basadas en las características aprendidas previamente.

Algunas de las características más importantes de las redes convolucionales son por ejemplo:

- **Capa de convolución:** Aplica filtros a las imágenes de entrada para detectar características simples en diferentes regiones. Cada filtro se desliza sobre la imagen y realiza una operación de convolución para generar mapas de características [29].
- **Capa de MaxPooling:** Reduce la dimensión espacial de las características extraídas mediante un proceso de submuestreo. Se selecciona el valor máximo en cada región de la imagen, lo que ayuda a mantener las características más relevantes y reduce la complejidad computacional [8].
- **Capa de aplanamiento (Flatten):** Transforma las características de las capas anteriores en un vector unidimensional. Esta capa se utiliza para preparar las características para ser procesadas por las capas totalmente conectadas [16].
- **Capas totalmente conectadas:** Consisten en neuronas conectadas a todas las neuronas de la capa anterior. Ayudan a aprender representaciones más complejas de las características extraídas. La última capa puede tener una sola unidad con una función de activación adecuada para la tarea de clasificación, como la función sigmoide para problemas binarios.
- **Dropout:** Apaga aleatoriamente un porcentaje de las neuronas durante el entrenamiento. Esto evita que la red dependa demasiado de neuronas específicas y fomenta la independencia entre las neuronas. Ayuda a prevenir el sobreajuste y mejora la generalización del modelo [3].
- **Batch Normalization:** Normaliza las activaciones intermedias de la red calculando la media y la varianza de cada capa en un lote de ejemplos de entrenamiento. Esto acelera la convergencia y mejora la estabilidad del entrenamiento al reducir la covariación de características[28].

En la Figura 2.11, vemos un ejemplo de una red con múltiples capas convolucionales. Se aplican los filtros a las imágenes de entrenamiento con distintas resoluciones, y la salida resultante de convolucionar cada imagen se emplea como entrada para la siguiente capa [15]. Al final de la red se incluyen una serie de capas totalmente conectadas, al estilo de un perceptrón multicapa estándar.

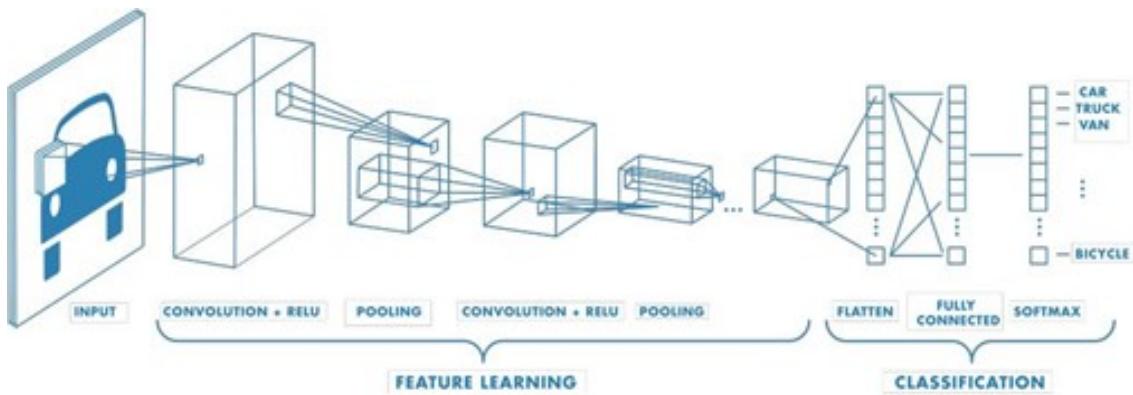


Figura 2.11: Ejemplo de CNN con múltiples capas convolucionales

2.4. Evaluación y validación

2.4.1. Clasificación binaria

Comenzaremos el estudio realizando una clasificación binaria sobre la base de datos correspondiente. En primer lugar, la base de datos de entrenamiento está formada por 1.790.594 imágenes. Sobre la cual se ha realizado una división con un 60% de las imágenes que pertenezcan al conjunto de entrenamiento, un 20% para validar y el restante 20% para testear los modelos.

Es importante tener en cuenta, que el conjunto de entrenamiento podemos retocarlo o transformarlo como queramos, mientras que para los conjuntos de validación y test, no podemos retocar nada, ya que estos son los pasos para realizar una evaluación metodológicamente correcta sobre nuestros modelos.

Analizando el conjunto de entrenamiento, detectamos que existían varias instancias en las que algunas variables como *pe_present_on_image*, *negative_exam_for_pe* e *indeterminate* eran 0, por tanto, resultaba un poco contradictorio con el gráfico de correlación de variables de la Figura 3.1. Nos estaba indicando que no es un examen negativo, tampoco es indeterminado y tampoco tiene embolia pulmonar presente en la imagen, así que decidimos eliminar dichos registros del conjunto de entrenamiento eliminando en total 308.500 instancias del conjunto de entrenamiento.

A continuación, en la Figura 2.12 vemos una distribución de la variable clase *pe_present_on_image* en el conjunto de entrenamiento, y como vemos está totalmente desbalanceada. Un 92,60% de las imágenes no presentan ningún tipo de embolia aparente, mientras que solamente el 7,40% de las imágenes presentan embolia.

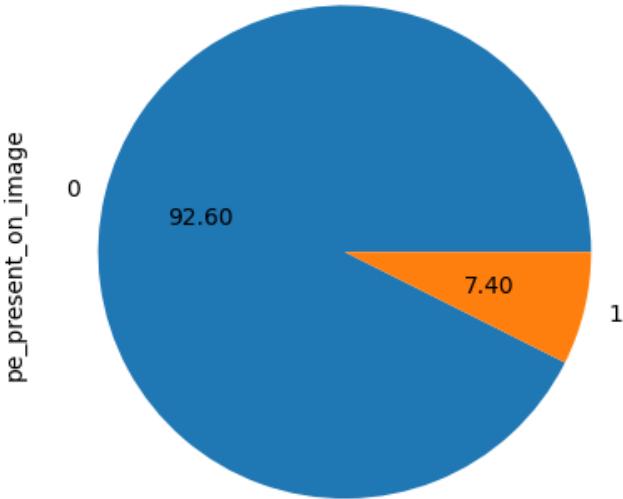


Figura 2.12: Conjunto de entrenamiento desbalanceado

Para tratar este problema, hemos optado por utilizar *random under-sampling*, una técnica de balanceo de datos que consiste en escoger la clase mayoritaria (775.458 imágenes sin embolia) y eliminar sus instancias hasta igualarse con la clase minoritaria (62.003 imágenes con embolia), en este caso eliminaremos imágenes que no posean embolia pulmonar hasta igualarla con las imágenes que si tienen embolia.

Una vez hemos realizado dicho proceso, el conjunto de entrenamiento se nos quedará totalmente balanceado, como podemos ver en la Figura 2.13. Ahora tenemos 62.003 imágenes con embolia y 62.003 imágenes sin embolia.

Para el conjunto de prueba tendremos 19.320 imágenes (5,39%) que contienen embolia y 338.799 (94,60%) imágenes que no contienen ningún tipo de embolia. Como vemos, el conjunto de prueba está totalmente desbalanceado, pero en este caso no podemos realizar ningún tipo de proceso de balanceo, ya que sobre este conjunto no se puede llevar a cabo ningún cambio.

Lo mismo ocurre con el conjunto de validación, el cual presenta 15.217 imágenes con embolia (5,31%) y 271.278 sin embolia (94,68%). Al igual que sobre el conjunto de test, el único proceso que haremos será introducir las imágenes en sus respectivos directorios. Esta distribución se puede encontrar en la libreta de Kaggle [24].

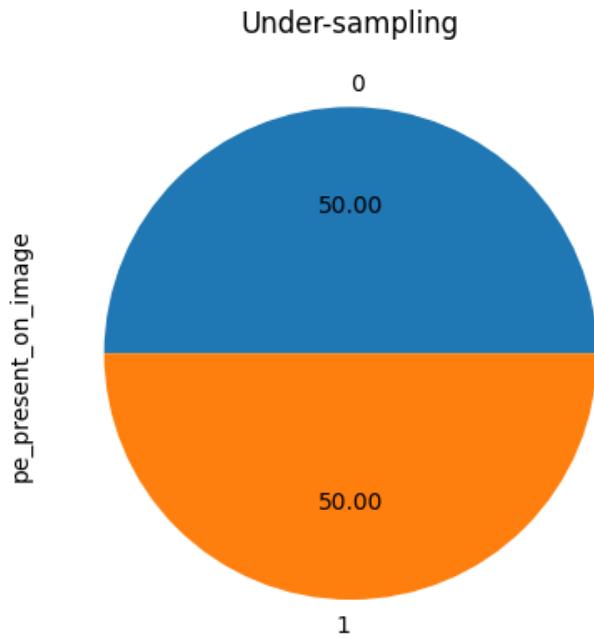


Figura 2.13: Conjunto de entrenamiento balanceado

2.5. Métricas de evaluación

Las métricas de evaluación son medidas utilizadas para evaluar la calidad y el rendimiento de los modelos utilizados en un problema de minería de datos. Estas métricas proporcionan una evaluación objetiva de como los modelos experimentados se ajustan sobre los datos, y como de precisos son a la hora de realizar predicciones o clasificaciones de nuevos casos.

Las métricas de evaluación, como el Área bajo la Curva (AUC), el recall, la precisión y el F1-score, son herramientas fundamentales en la evaluación de modelos de aprendizaje automático, especialmente en problemas de clasificación binaria [19].

A continuación veremos en que consisten cada una de estas métricas:

- **Área bajo la Curva (AUC):** Métrica comúnmente utilizada para evaluar la calidad de un clasificador binario. Representa la capacidad del modelo para distinguir entre clases positivas y negativas. Un valor de AUC cercano a 1 indica un clasificador con un rendimiento excelente, mientras que un valor cercano a 0.5 indica un rendimiento similar al azar, como vemos en la Figura 2.14a y sus respectivas fórmulas en la Figura 2.14b.

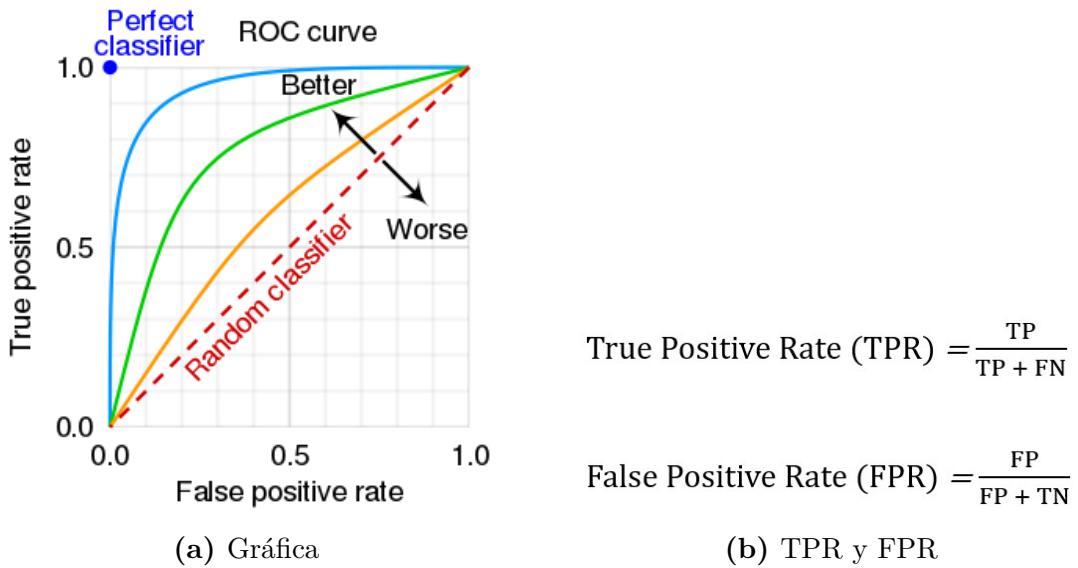


Figura 2.14: Área bajo la curva ROC

- **Matriz de confusión:** Es una representación tabular que muestra el rendimiento de un modelo de clasificación. La matriz de confusión muestra el número de casos clasificados correctamente e incorrectamente en las diferentes clases que podemos observar en la Figura 2.15. A partir de la matriz de confusión, se pueden calcular otras métricas como el recall, la precisión y el F1-score.

		PREDICCIÓN	
		POSITIVO	NEGATIVO
VALOR ACTUAL	POSITIVO	TP	FN
	NEGATIVO	FP	TN

Figura 2.15: Matriz de confusión

-
- **Recall:** Esta métrica mide la proporción de instancias positivas que se identifican correctamente. Es una métrica importante cuando se desea minimizar los falsos negativos, es decir, evitar clasificar incorrectamente instancias positivas como negativas, en la Figura 2.16 vemos su fórmula.

$$Recall = \frac{TP}{TP + FN}$$

Figura 2.16: Fórmula del Recall

- **Tasa de acierto(Accuracy):** Mide la proporción de casos clasificados correctamente por el modelo, es decir, el número de casos clasificados correctamente dividido por el número total de casos, su fórmula se encuentra en la Figura 2.17. Es una métrica básica y generalmente utilizada para problemas de clasificación binaria y multiclas.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figura 2.17: Fórmula del accuracy

- **Precisión:** Métrica que representa el número de verdaderos positivos que son realmente positivos en comparación con el número total de valores positivos predichos, en la Figura 2.18 vemos su fórmula.

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Figura 2.18: Fórmula de la precisión

2.6. Cross Industry Standard Process for Data Mining (CRISP-DM)

Cross Industry Standard Process for Data Mining (CRISP-DM) es una metodología capaz de proporcionar una descripción normalizada del ciclo de vida de un proyecto estándar de análisis de datos. Este se divide en 6 fases como podemos observar en la Figura 2.19. Las fases que aparecen no es obligatorio realizarlas en orden, depende de la salida de cada una de las fases, o las diferentes tareas que componen cada fase [40].

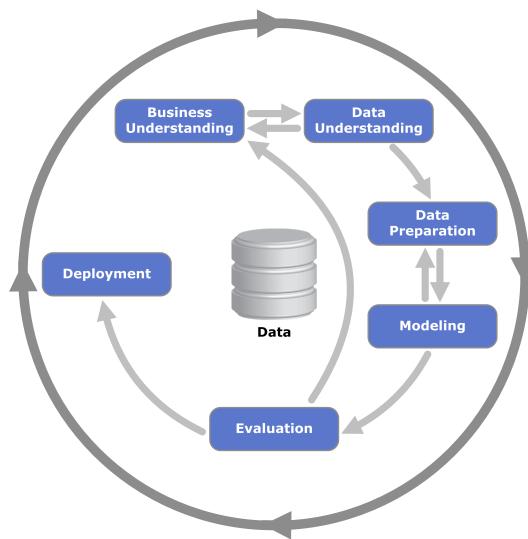


Figura 2.19: Fases de la metodología CRISP-DM Process Model for Data Mining

A continuación, haremos un breve análisis de las fases que componen esta metodología [40]:

- ***Business Understanding:*** Esta fase inicial se basa en entender los objetivos y requerimientos del proyecto, para posteriormente convertir dicho conocimiento en un problema de minería de datos.
- ***Data Understanding:*** Comenzaremos esta fase con una conjunto de datos y realizaremos las actividades necesarias para familiarizarnos con los datos o identificar sus problemas. La formulación del problema y el plan de nuestro proyecto requiere la comprensión de los datos.
- ***Data Preparation:*** La preparación de datos consiste en realizar todas las actividades necesarias para construir el conjunto de datos final, es decir, aquel conjunto de datos que introduciremos en los modelos. En este caso, la preparación de los datos no tiene un orden prescrito y es posible que haya que realizar dichos procesos varias veces, como por ejemplo: Limpieza o transformación de los datos.
- ***Modeling:*** En esta fase, seremos capaces de seleccionar y aplicar varias técnicas de modelado. Normalmente, existen unas cuantas técnicas para el mismo tipo de problemas de minería de datos, otras, en cambio, requieren los datos en un formato específico.
- ***Evaluation:*** Una vez hayamos construido uno o más modelos con un alto grado de calidad, desde el punto de vista del análisis de datos. Antes de desplegar el modelo, es importante realizar una evaluación minuciosa del mismo, para asegurarnos que cumple con las especificaciones u objetivos que hemos definido en la fase de *business understanding*.
- ***Deployment:*** La creación de los modelos no será el final del proyecto. Necesitamos organizar el conocimiento sobre el problema y presentarlo de manera que los clientes sean capaces de usarlo. Dependiendo del proyecto, el despliegue puede ser tan simple como solamente generar un documento o tan complejo como implementar un proceso de minería de datos repetible.

Capítulo 3. Estudio de los datos

3.1. Introducción

En este capítulo se analizará todo lo relativo a la ciencia de datos, desde el estudio de los datos hasta el análisis de los mismos. Se utilizará la metodología *CRISP-DM* (*Cross Industry Standard Process for Data Mining*), cuyo primer paso (*Business understanding*) se llevó a cabo en el Capítulo 2.

Nos centraremos en el segundo paso de la metodología *CRISP-DM* (*Data Understanding*) el cual consiste en familiarizarnos y analizar los datos disponibles [40]. Tal y como se especifica en el Capítulo 1, los datos se pueden encontrar en la página oficial de la competición [23] o en la competición organizada en Kaggle [21].

El conjunto de datos original contiene 1.790.594 CTPA (*Computed Tomography Pulmonary Angiogram*). Cada imagen contiene una representación del tórax de los pacientes, tal y como se muestra en la Figura 2.4. Este conjunto de imágenes corresponden a un total de 7.279 pacientes. Adicionalmente, cada imagen se complementa con un registro de datos de dicha imagen, como pueden ser si existe embolia o la localización de la misma (posteriormente se detallará en profundidad en la Tabla 3.1).

Las imágenes han sido proporcionadas por la Universidad de Stanford (Palo Alto, California), Unity Health Toronto (Toronto, Canadá), Universidad Federal de São Paulo (São Paulo, Brasil), Alfred Health (Melbourne, Australia), y la Universidad Koç (Estambul, Turquía)[4].

Los datos usados en este trabajo son aquellos localizados en la competición de Kaggle [21]. Los ficheros obtenidos por medio de esta competición se encuentran listados y descritos en la Tabla 3.1.

La competición busca obtener un modelo que devuelva una probabilidad (entre 0 y 1) dada una imagen y sus correspondientes metadatos, de tal forma que seamos capaces de identificar como y donde se encuentra la embolia pulmonar.

Ficheros/Carpetas	Descripción
<i>train</i>	Carpeta contenedora de todos los registros de entrenamiento en formato DICOM.
<i>test</i>	Carpeta contenedora de todos los registros de prueba en formato DICOM.
<i>sample_submission.csv</i>	Fichero de ejemplo para conocer el formato que ha de tener el fichero a subir a la competición.
<i>train.csv</i>	Fichero que contiene el conjunto de datos de entrenamiento, donde el diagnóstico es conocido. Se estudiará en profundidad en la Sección 3.2.
<i>test.csv</i>	Fichero que contiene el conjunto de datos de test, donde el diagnóstico es desconocido. Con este fichero se han de generar las predicciones para la competición.

Tabla 3.1: Contenido ubicado en la competición de Kaggle "RSNA STR Pulmonary Embolism Detection" [22].

3.2. Metadatos de los pacientes

A continuación, se analizará de forma detallada los datos que podemos encontrar en el fichero *train.csv* (Tabla 3.1). Este fichero contiene un total de 1.790.594 entradas únicas de CTPA divididas entre 7.279 pacientes diferentes. En la Tabla 3.2 se recogen todas las variables recopiladas en el fichero.

Estas variables se encuentran correlacionadas entre sí, como veremos en la Figura 3.1. De las cuales, cuatro de ellas son solamente informativas y no requieren realizar ningún tipo de predicción. Estas son: QA Contrast, QA Motion, True filling defect not PE, y Flow artifact.

Variable	Descripción
<i>StudyInstanceUID</i>	ID únicos de cada estudio en los datos. (Paciente).
<i>SeriesInstanceUID</i>	ID únicos de cada serie en el estudio. (N. ^o de serie del Paciente).
<i>SOPInstanceUID</i>	ID único de cada imagen.
<i>pe_present_on_image</i>	Indica si existe o no embolia pulmonar en la imagen.
<i>negative_exam_for_pe</i>	Indica si no existe embolia pulmonar en la imagen.
<i>qa_motion</i>	Indica si existe algún tipo de problema con respecto al movimiento de la imagen.
<i>qa_contrast</i>	Indica si existe algún tipo de problema con respecto al contraste de la imagen.
<i>flow_artifact</i>	Información adicional.
<i>rv_lv_ratio_gte_1</i>	Indica si el radio del VD(Ventriculo derecho)/VI(Ventriculo izquierdo) es mayor o igual que 1.
<i>rv_lv_ratio_lt_1</i>	Indica si el radio del VD(Ventriculo derecho)/VI(Ventriculo izquierdo) es menor que 1.
<i>leftsided_pe</i>	Indica que existe una embolia pulmonar en la parte izquierda de la imagen a estudiar.
<i>chronic_pe</i>	Indica que la embolia pulmonar existente en el estudio es crónica.
<i>true_filling_defect_not_pe</i>	Indica un defecto en el pulmón que no es una embolia pulmonar.
<i>rightsided_pe</i>	Indica que existe una embolia pulmonar en la parte derecha de la imagen a estudiar.
<i>acute_and_chronic_pe</i>	Indica que la embolia pulmonar existente en la imagen es aguda y crónica.
<i>central_pe</i>	Indica que existe una embolia pulmonar en la parte central de la imagen a estudiar.
<i>indeterminate</i>	Mientras el estudio de la imagen no es negativo sobre la presencia de embolia pulmonar en la imagen, indica que no es posible realizar un estudio exhaustivo sobre la imagen debido a <i>qa_motion</i> y <i>qa_contrast</i> .

Tabla 3.2: Contenido *train.csv* [22].

En el primer caso, tenemos que detectar si la imagen que estamos observando es o no negativo con la supuesta embolia pulmonar. En caso de que si sea negativo (es decir, que no exista ningún tipo de embolia pulmonar) habremos terminado y con esa imagen no hay nada que hacer, en cambio, si el examen no es negativo (es decir, hay embolia pulmonar aparente en la imagen) pasaremos a Indeterminado.

En segundo lugar, tenemos que ver si la imagen que estamos observando es o no Indeterminado. Será Indeterminado cuando la imagen no se pueda apreciar bien, debido al contraste (*Q Contrast*) o el movimiento (*Q Motion*) de la imagen. En cambio, si no es Indeterminado, significa que el paciente presenta una embolia perfectamente observable, ahora tenemos que ver de que se trata el embolismo.

Una vez que nos encontramos ya ante una embolia pulmonar, tenemos que evaluar los distintos parámetros que nos permitirán detectar la gravedad del embolismo (solamente podemos seleccionar un parámetro), la localización del mismo (tenemos que seleccionar al menos un parámetro) y el radio del ventrículo derecho (*RV*) hasta el ventrículo izquierdo(*LV*) (solamente podremos seleccionar un parámetro) [22].

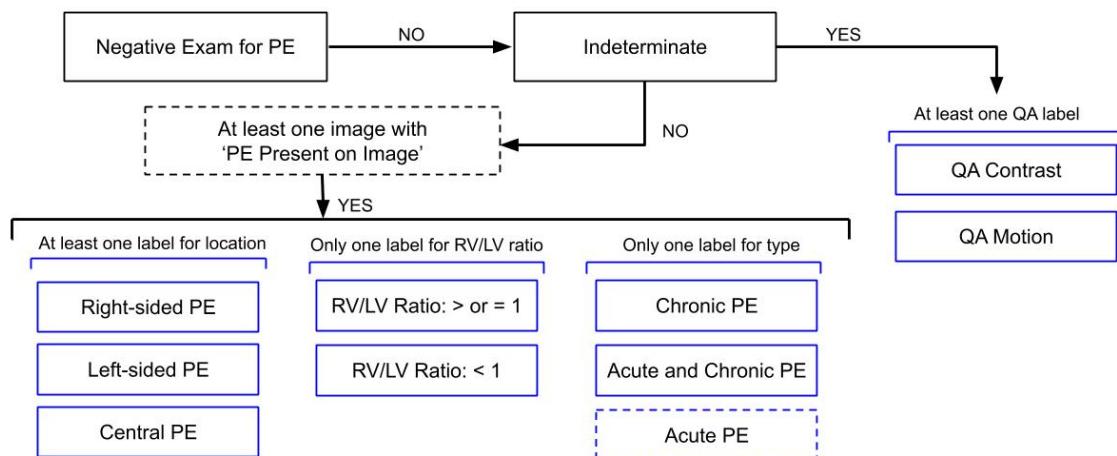


Figura 3.1: Diagrama de relación de variables

3.2.1. Variable *"negative_exam_for_pe"*

La variable *"negative_exam_for_pe"* indica si no existe embolia pulmonar en la imagen (0 - No es negativo: Por tanto, es probable que haya embolia pulmonar; 1 - Es negativo: No hay embolia pulmonar).

En la siguiente Figura 3.2 vemos que en la distribución las imágenes negativas abundan en la base de datos con un 67,6%, por tanto, podemos concluir que esas imágenes no tienen ningún tipo de embolismo, en cambio, aquellas imágenes que pasaremos a comprobar si son indeterminadas constituyen el 32,4%.

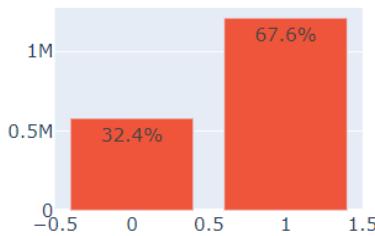


Figura 3.2: Distribución de la variable *negative_exam*

3.2.2. Variable *"indeterminate"*

La siguiente variable que tenemos será *"indeterminate"*, sobre ese 32,4% que hemos visto anteriormente, en ella indicaremos la existencia de algún tipo de problema que no nos permita realizar un estudio exhaustivo sobre la variable (0 - No hay ningún problema, 1 - Hay problemas con QA_motion o QA_contrast).

Como vemos en la Figura 3.3, existe un importante desbalanceo de los datos, ya que solamente el 2% del total de las imágenes presentan problemas con el movimiento o el contraste, mientras que el resto de imágenes pasarán al siguiente filtro que será si tienen o no embolismo pulmonar (*"pe_present_on_image"*).

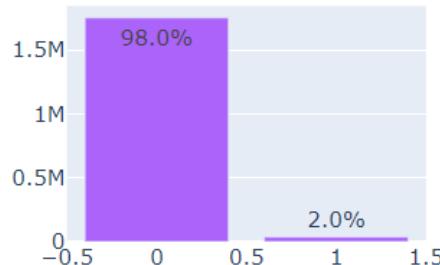


Figura 3.3: Distribución de la variable *indeterminate*

3.2.3. Variable *"qa_motion"*

Con el 2% que hemos visto anteriormente, la variable *"qa_motion"* indica si los médicos detectaron problemas en el movimiento de la imagen que estamos estudiando (0 - No existe Movimiento y la imagen se puede estudiar perfectamente; 1 - Tenemos problemas con el movimiento). Esta variable es solamente de información, y como vemos en la Figura 3.4 el 99,2% del total de las imágenes no tienen ningún tipo de problema.



Figura 3.4: Distribución de la variable *qa_motion*

3.2.4. Variable ”*qa_contrast*”

Al igual que la Sección 3.2.3, la variable ”*qa_contrast*” indica si los médicos detectaron problemas en el contraste de la imagen que estamos estudiando (0 - No existe ningún problema de contraste y la imagen se puede estudiar perfectamente; 1 - Tenemos problemas con el contraste). Esta variable es solamente de información, como podemos observar en la Figura 3.5 el 98,4% del total de las imágenes no tienen ningún problema de este tipo.

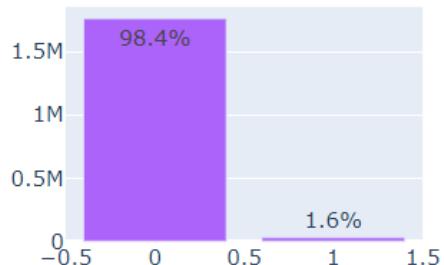


Figura 3.5: Distribución de la variable *qa_contrast*

3.2.5. Variable ”*pe_present_on_image*”

La variable ”*pe_present_on_image*” identifica si existe o no una embolia pulmonar en la imagen que estamos estudiando. (0 - No ”PE” , 1 - ”PE”).

En la Figura 3.6, podemos observar una distribución de esta variable entre las 1.790.594 imágenes que tenemos en la base de datos. Vemos que la variable se encuentra desbalanceada, un 94,6% del total de imágenes no contienen embolia pulmonar, mientras que un 5,4% de imágenes restante si contiene embolia.

Por tanto, podemos concluir que la distribución de esta variable se encuentra totalmente desbalanceada, ya que existen muchos más casos que no tienen embolia con respecto a los que tienen. Más adelante veremos como poder tratar este desbalanceo para realizar una clasificación binaria.



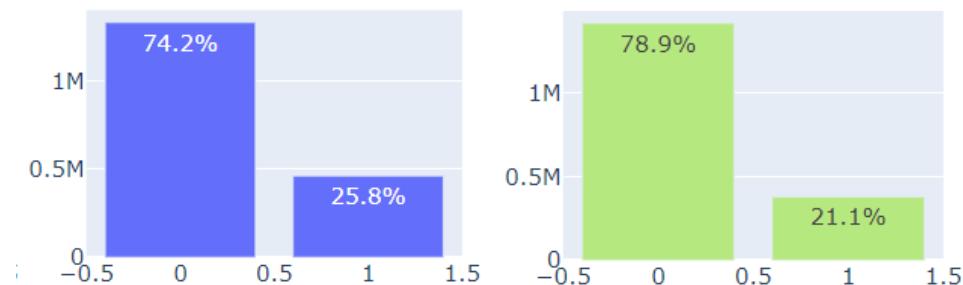
Figura 3.6: Distribución de la variable *pe_present_on_image*

3.2.6. Variables "*rightsided_pe*", "*leftsided_pe*" y "*central_pe*"

Estas tres variables representan la posición en la que se encuentra la embolia pulmonar en la imagen que estamos estudiando (parte derecha, parte izquierda y parte central de la imagen, respectivamente). Estas 3 variables son a nivel de examen, y debemos escoger al menos una de ellas, lo que quiere decir, que se pueden dar los 3 casos en una misma imagen.

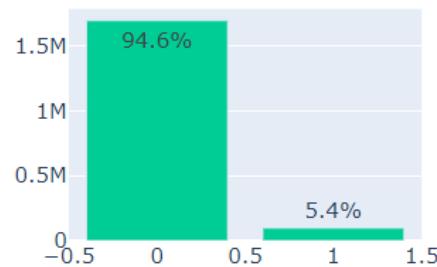
Como vemos en la Figura 3.7, existe un 25,8% de las imágenes que presentan embolia en la parte derecha de la imagen, al igual que en el caso de que se encuentre en la parte izquierda de la imagen con un 21,1%, en cambio, tenemos que la gran mayoría de las imágenes no contiene embolias en la parte central con un 5,4%.

En la Figura 3.8 vemos lo que hemos comentado anteriormente, se puede dar el caso en el que las 3 variables sean 1 al mismo tiempo, ya que la embolia puede ocupar gran parte de la imagen, o haber más de una embolia en el torax del paciente. En este ejemplo deberíamos ser capaces de detectar embolias pulmonares en la parte izquierda, central y derecha.



(a) Distribución de la variable rightside_pe

(b) Distribución de la variable leftside_pe



(c) Distribución de la variable central_pe

Figura 3.7: Localización de la embolia



Figura 3.8: Ejemplo de embolia pulmonar en la parte derecha, izquierda y central

3.2.7. Variables "rv_lv_ratio_gte_1" y "rv_lv_ratio_lt_1"

Estas dos variables indican el radio del corazón, es decir, la distancia del Ventrículo Derecho (*RV*) del corazón hasta el Ventrículo Izquierdo (*LV*), puede ser mayor o igual que 1 (*gte = greater than or equal*, $rv_lv_ratio_gte_1 = 1$) o menor que 1 (*lt = lower than*, $rv_lv_ratio_lt_1 = 1$) Para un ejemplo más visual, podemos ver la Figura 3.9. En esta se observa que el radio del ventrículo derecho es menor que el radio del ventrículo izquierdo, esto se debe a que la parte izquierda del corazón es la encargada de proporcionar sangre a todo el organismo y, por tanto, el empuje de sangre debe ser mayor, mientras que el ventrículo derecho solamente proporciona sangre a los pulmones.

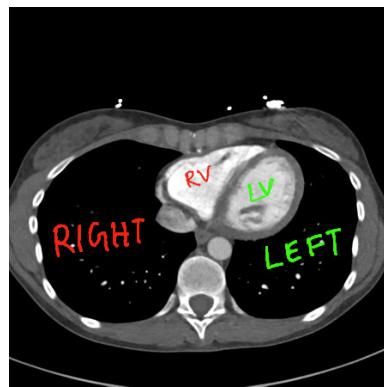


Figura 3.9: Ejemplo gráfico de disposición de los ventrículos

En la Figura 3.10, vemos la distribución de ambas variables en la base de datos, es importante comentar que solamente podemos seleccionar una o ninguna de estas variables en cada imagen (este último caso solamente se da cuando nos encontramos en la variable "*negative_exam_for_pe*" y, por tanto, como podemos ver en la Figura 3.1 no somos capaces de alcanzar estas variables).



Figura 3.10: Distribución de la variable radio del ventrículo derecho hasta el ventrículo izquierdo [22].

3.2.8. Variables "*chronic_pe*" y "*acute_and_chronic_pe*"

Para finalizar, las variables *chronic_pe* y *acute_and_chronic_pe* indican la gravedad de la embolia en la imagen que estemos estudiando. La embolia puede ser crónica o aguda y crónica.

En la Figura 3.11 podemos ver la distribución de las variables, en la que tenemos un 1,9% de imágenes que presentan embolias agudas y crónicas para los pacientes.

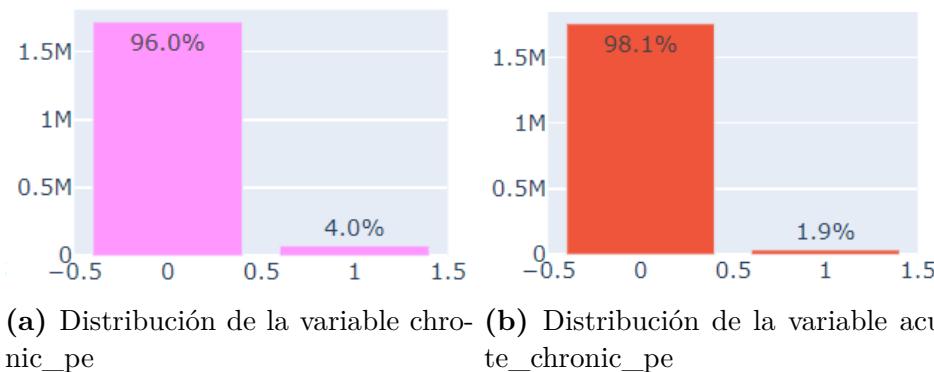


Figura 3.11: Gravedad de la embolia

3.3. Análisis de imágenes

Una vez realizado el análisis de los metadatos de los pacientes, continuaremos realizando el análisis de las imágenes, las cuales serán la principal fuente de información entre si existen o no embolias pulmonares.

Hay que recordar que el conjunto de entrenamiento original de la competición está compuesto por 1.790.594 imágenes, es por ello por lo que hemos decidido que para realizar un estudio binario, dividiremos este directorio en un 60% para el conjunto de entrenamiento, 20% para el conjunto de prueba y 20% para el conjunto de validación. En el Capítulo 2.4 llevamos a cabo un estudio más exhaustivo de este problema.

En primer lugar, todas las imágenes se encuentran en formato *DICOM* (*Digital Imaging and Communications in Medicine*). Las imágenes médicas por sí solas no aportan suficiente información, por ello para que sea correctamente interpretada debe ir acompañada de datos del paciente. El formato *DICOM* cuenta con objetos *IOD* (*Information Object Definition*), formados por la imagen y su información asociada [35].

Un *IOD* se compone de IEs (Entidades de información) (Hay IE de paciente, de estudio, de serie, de equipo, de imagen...).

En la Figura 3.12 vemos un ejemplo de un escáner en formato DICOM.

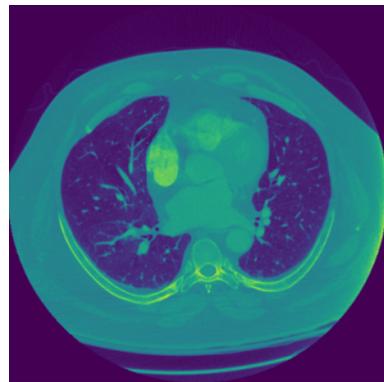


Figura 3.12: Imagen en formato DICOM

A pesar de toda la información que nos puede llegar a proporcionar el formato DICOM, este trabajo está dirigido con un propósito académico, y por temas de espacio y tiempo de computación hemos decidido convertir las imágenes en formato JPG 256x256. En este caso, no hemos realizado la conversión de las imágenes a mano, ya que encontramos una base de datos procedente de Kaggle [26] que ya nos proporcionaba todas las imágenes convertidas al formato que nosotros deseábamos.

En la Figura 3.13 podemos ver la Figura anterior 3.12 en formato JPG.

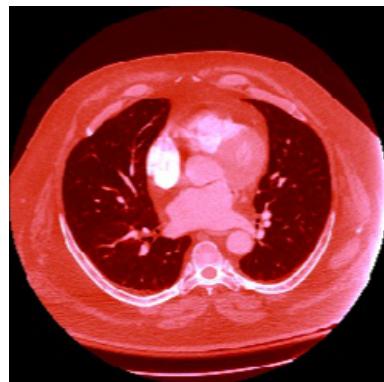


Figura 3.13: Imagen en formato JPG

Capítulo 4. Preprocesamiento de los datos

4.1. Introducción

El preprocesamiento de los datos es una de las fases más importantes en el ciclo de vida de un proyecto de Minería de datos. Ya que mejorar el *dataset* ayudará al modelo en el proceso de entrenamiento a encontrar patrones más significativos y evidentes.

Une vez hemos realizado un estudio exhaustivo de los datos, pasaremos a la siguiente fase de la metodología *CRISP-DM*, *Data preparation* o Preparación de los datos. En este capítulo llevaremos a cabo las modificaciones sobre el conjunto de datos original, y así obtener un conjunto de datos adecuado para entrenar los modelos.

En el Capítulo 5 explicaremos los diferentes modelos que hemos llegado a usar, en el caso de este trabajo utilizaremos Redes Neuronales para realizar las predicciones.

4.2. Flujo de preprocesamiento de las imágenes

En la Sección 3.3 hemos analizado las imágenes que se encuentran dentro del conjunto de datos. A continuación presentaremos algunos algoritmos utilizados donde es posible mejorar la precisión de los modelos que serán posteriormente explicados en el Capítulo 5.

El primer paso será convertir las imágenes a escala de grises, ya que estas tienen varias ventajas sobre las imágenes JPG mostradas en color, como la que vemos en la Figura 3.13.

Para complementar la conversión de las imágenes a escala de grises, utilizaremos varios algoritmos para filtrar sus características y resaltar sus componentes importantes.

4.2.1. Filtro de realce de bordes de Sobel

El operador de sobel es usado en procesamiento de imágenes y visión por computador. Basicamente, este filtro resalta los bordes en una imagen, basado en la detección de cambios abruptos en la intensidad de los píxeles para resaltar transiciones entre diferentes regiones en una imagen.

El filtro de Sobel se aplica la convolución de la imagen original con dos máscaras (kernels) 3x3 separadas, una para la detección de bordes horizontales y otra para la detección de bordes verticales. Estas máscaras están diseñadas para calcular las derivadas de primer orden de la imagen en las direcciones horizontal y vertical [39].

Por ejemplo, en la Figura 4.1: Definimos G_x y G_y como los dos kernels que serán capaces de detectar los bordes de la imagen de manera horizontal y verticalmente [7].

-1	0	+1
-2	0	+2
-1	0	+1

G_x

+1	+2	+1
0	0	0
-1	-2	-1

G_y

Figura 4.1: 3x3 kernels del filtro de Sobel

Para ello debemos obtener la imagen de entrada, en la Figura 4.2 vemos como se aplican las operaciones a la imagen de entrada con el kernel G_x .

10	9	9	4	0
0	6	6	2	2
5	9	8	4	3
7	5	5	4	3
8	10	8	5	0

Imagen de entrada

-1	0	1
-2	0	2
-1	0	1

G_x

$$(10 * -1) + (9 * 0) + (9 * 1) + (0 * -2) + (6 * 0) + (6 * 2) + (5 * -1) \\ + (9 * 0) + (8 * 1) = 14$$

...

14		

14	-18	-22
10	-15	-16
-1	-12	-17

I_x

Figura 4.2: Imagen resultante de aplicar el Kernel G_x

En la Figura 4.3 vemos el resultado pero aplicando el kernel G_y .

10	9	9	4	0
0	6	6	2	2
5	9	8	4	3
7	5	5	4	3
8	10	8	5	0

Imagen de entrada

1	2	1
0	0	0
-1	-2	-1

G_y

$$(10*1) + (9*2) + (9*1) + (0*0) + (6*0) + (6*0) + (5*-1) + \\ (9*-2) + (8*-1) = 6$$

...

6		

6	2	-2
-4	-1	-4
-5	-2	1

I_y

Figura 4.3: Imagen resultante de aplicar el Kernel G_y

A continuación, en la Figura 4.4 realizamos las siguientes operaciones para medir la magnitud del gradiente.

14	-18	-22
10	-15	-16
-1	-12	-17

 I_x

6	2	-2
-4	-1	-4
-5	-2	1

 I_y

$$A = \sqrt{I_x^2 + I_y^2}$$

$$A = \sqrt{14^2 + 6^2} = 15.23 \approx 15$$

$$A = \sqrt{(-18)^2 + 2^2} = 18.11 \approx 18$$

$$A = \sqrt{-22^2 + (-2)^2} = 22.09 \approx 22$$

$$A = \sqrt{10^2 + (-4)^2} = 10.77 \approx 11$$

$$A = \sqrt{(-15)^2 + 1^2} = 15.03 \approx 15$$

$$A = \sqrt{(-16)^2 + (-4)^2} = 16.49 \approx 16$$

$$A = \sqrt{(-1)^2 + 5^2} = 5.09 \approx 5$$

$$A = \sqrt{(-12)^2 + (-2)^2} = 12.16 \approx 12$$

$$A = \sqrt{(-17)^2 + 1^2} = 17.02 \approx 17$$

15	18	22
11	15	16
5	12	17

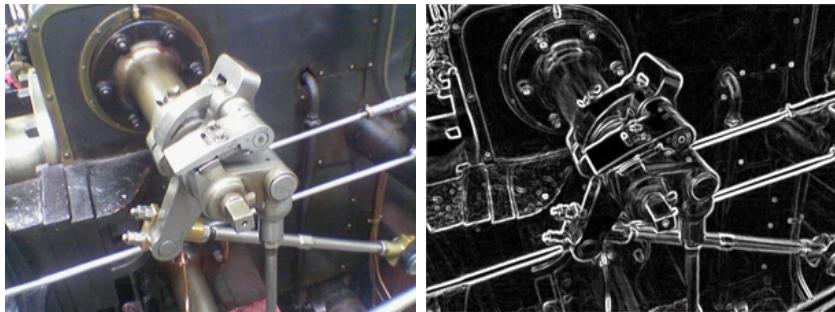
Figura 4.4: Matriz de la magnitud del gradiente

Sumaremos todos los valores de la matriz resultante, y posteriormente dividimos dicha suma por el número de elementos que tenga la matriz. $15 + 18 + 22 + 11 + 15 + 16 + 5 + 12 + 17 = 131/9 = 14.55 \approx 15$. Ahora, aquellos valores mayores que 15, pasarán a valer 1, y aquellos menores o iguales que 15, pasarán a valer 0, como podemos ver en la Figura 4.5.

0	1	1
0	0	1
0	0	1

Figura 4.5: Matriz resultante del filtro de Sobel

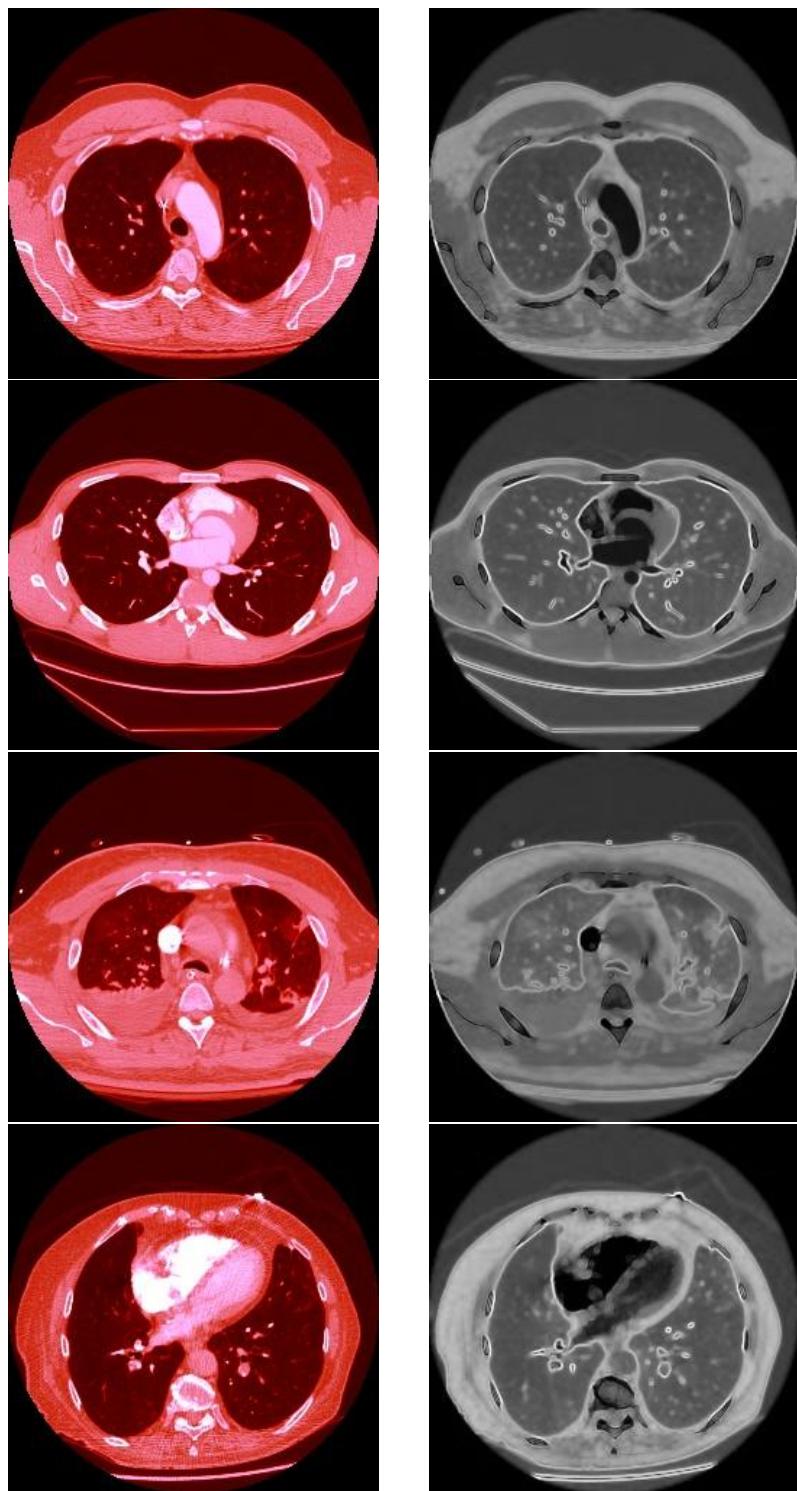
El resultado de aplicar el filtro de Sobel es una imagen que resalta los bordes, donde los píxeles correspondientes a los bordes tienen valores de intensidad más altos, como podemos observar en la Figura 4.6. Esto puede facilitar la detección y el análisis de estructuras y características relevantes en la imagen.



(a) Figura de un motor en color (b) Aplicando el filtro de Sobel

Figura 4.6: Ejemplo del filtro de Sobel.

El resultado que obtenemos al convertir todas las CTPA de los pacientes, es el que podemos ver en la Figura 4.7, estas imágenes serán introducidas en sus respectivos directorios, las cuales comentamos en el Capítulo 2.4 dependiendo de sus características.



(a) CTPA con embolia en color (b) CTPA con el filtro de Sobel

Figura 4.7: Ejemplo del filtro de Sobel.

Capítulo 5. Creación del modelo

5.1. Introducción

Tras el procesamiento realizado en el conjunto de datos original en el Capítulo 4, pasaremos a la siguiente fase de la metodología CRISP-DM, el modelado o *modeling*, descrito en el Capítulo 2.6.

En este capítulo se llevarán a cabo varios modelos de Redes Neuronales Convolucionales (CNN), y alguna implementación donde introduciremos un modelo de *deep learning* y un ensemble XGBoost.

5.2. Recursos hardware y conjunto de datos

Todos los experimentos y modelos han sido creados en la plataforma online Kaggle [21]. En las ejecuciones se ha hecho uso de la GPU que nos proporciona el sistema para acelerar el proceso. Kaggle cuenta con un servicio gratuito de GPU bajo una cuota de 30 horas semanales por usuario. La tarjeta gráfica incorporada por la aplicación es una NVIDIA TESLA P100. Kaggle también presenta un límite de ejecución de 9 horas por libreta, por lo que los experimentos y las ejecuciones no podrán sobrepasar ese límite.

A raíz del preprocesamiento que hemos realizado sobre la base de datos, en el Capítulo 2.4.1, y en el Capítulo 4.2.1 sobre las imágenes para realizar experimentos, se crearán nuevos conjuntos de datos con las modificaciones pertinentes, pero siempre manteniendo la misma distribución existente en la partición original. De esta forma, no se tendrá que aplicar continuamente el flujo tantas veces como modelos queramos realizar.

El primer conjunto de datos creado está constituido por imágenes JPG. Para el conjunto de entrenamiento tenemos 124.006 imágenes, de las cuales, el 50% presenta embolia y el otro 50% de las imágenes no presenta ningún rastro de embolia [24].

Para el conjunto de validación tenemos 286.495 imágenes, de las cuales el 5,31% presenta embolia y el 94,68% no presenta ningún indicio [24].

El conjunto de prueba tenemos 358.119 imágenes, de las cuales el 5,39% de las imágenes presentan embolia y el 94,60% de ellas no [24].

En cambio, el segundo conjunto de datos está constituido por la misma cantidad de imágenes en cada uno de los conjuntos de entrenamiento, validación y test, pero esta vez con las imágenes convertidas con el Filtro de Sobel [25].

5.3. Redes neuronales

En esta sección se explorarán los modelos basados en redes neuronales. Se analizarán los modelos creados, cómo se han validado y qué modificaciones se han realizado para observar como afectaban al modelo final.

Las redes con las que se han experimentado son principalmente redes convolucionales, ya que presentan la capacidad de procesar imágenes y extraer características o patrones visuales de las imágenes de manera eficiente.

En este caso hemos propuesto alguna arquitectura propia, aunque también usaremos redes ya existentes y preentrenadas con los pesos de *Imagenet*. Una red inicializada con los pesos de *Imagenet* quiere decir que ha sido entrenada previamente con imágenes ubicadas en una base de datos que consta 14 millones de imágenes repartidas en 20.000 categorías diferentes.

Las redes o arquitecturas que con las que se van a experimentar serán:

- Diseño propio de una CNN básica.
- VGG19.
- ResNet50.
- Xception.
- TransferLearning model + XGBoost.

En cada uno de estos modelos la validación se realizará al terminar cada epoch. Un epoch es cuando el conjunto de datos de entrenamiento completo ha pasado a través de la red neuronal una vez. Tras finalizar esta fase se usará el conjunto de validación para obtener el área bajo la curva ROC para conocer el estado del entrenamiento del modelo.

El entrenamiento de las redes neuronales conlleva realizar varios epochs. Una vez alcanzado el límite de epochs, habrá finalizado la fase de entrenamiento. Pero debemos tener en cuenta un factor importante a la hora de realizar el entrenamiento de los modelos, saber si está sobreajustando. Debemos observar lo que varían las métricas de evaluación del modelo, si la validación no mejora o empeora, mientras que el entrenamiento no hace más que mejorar es un claro indicio de que la red está sobreajustando, por tanto, no estaríamos entrenando correctamente.

Las redes neuronales han sido construidas y entrenadas con la ayuda de la librería *TensorFlow*, la cual proporciona una plataforma flexible y eficiente para el cálculo numérico y la creación de modelos de aprendizaje profundo. También usaremos el paquete *Keras* de Python, que permite construir modelos de manera modular, proporcionando una amplia gama de capas predefinidas para construir redes convolucionales.

5.3.1. Diseño propio de una CNN Básica

Comenzamos realizando modelos de redes convolucionales y demostrar así su eficacia en el procesamiento de imágenes. En esta sección se mostrarán los experimentos realizados y sus resultados, sobre una red convolucional creada por nosotros mismos.

En primer lugar, llevamos a cabo una red neuronal convolucional simple, a modo de baseline, para así comprobar que todo funciona y entrena correctamente respecto a las imágenes sin preprocessar que tenemos.

- La red comienza con una capa de convolución inicial que aplica 32 filtros de tamaño 3×3 a las imágenes de entrada, estos elementos han sido explicados en el Capítulo 2.3.3. Esto ayuda a detectar características simples en las imágenes.
- Después de cada capa de convolución, se utiliza una capa de *MaxPooling* para reducir la dimensión espacial de las características extraídas y mantener las características más relevantes. Esto ayuda a reducir la cantidad de hiperparámetros y la complejidad computacional de la red [8].
- Después se utiliza una capa de *MaxPooling* para reducir la dimensión espacial de las características extraídas y mantener las características más relevantes. Esto ayuda a reducir la cantidad de hiperparámetros y la complejidad computacional de la red [8].

-
- La red continúa con más capas de convolución y *MaxPooling*, aumentando gradualmente la complejidad y la abstracción de las características extraídas. Esto permite a la red capturar características más sofisticadas y abstractas a medida que profundiza en la arquitectura.
 - Después de las capas de convolución y *MaxPooling*, las características se aplanan en un vector unidimensional utilizando la capa de aplanamiento (*Flatten*). Esto prepara las características para ser procesadas por las capas totalmente conectadas [16].
 - Las capas totalmente conectadas consisten en una capa con 128 unidades, la cual, ayuda a aprender representaciones más complejas de las características, también presenta una capa final con una unidad, que utiliza la función de activación sigmoide para generar una salida binaria que representa la probabilidad de pertenencia a la clase positiva.

La diferencia de este primer modelo con el resto utilizados en esta CNN básica, es que no hemos introducido técnicas de *Dropout* y *Batch Normalization* explicadas en el Capítulo 2.3.3, y hay que tener en cuenta que estas pueden tener varios efectos beneficiosos en el rendimiento y la generalización del modelo.

La motivación por la que se ha decidido usar esta red convolucional es simplemente para observar si los modelos son capaces de generalizar las características y patrones que presentan las imágenes, de tal forma que seamos capaces de trabajar a partir de él, empezando con un modelo inicial y seguidamente ir introduciéndole características adicionales.

En la Figura 5.1 podemos ver una representación gráfica del modelo inicial.

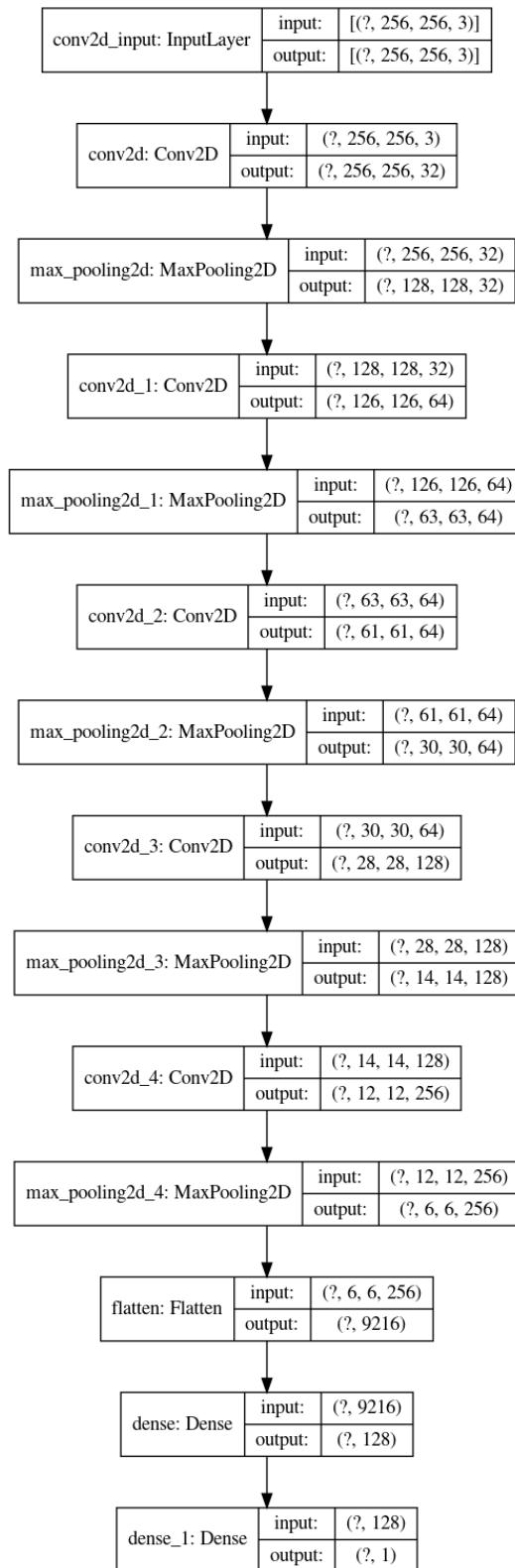


Figura 5.1: Arquitectura CNN propia.

Resultados

En las Tablas 5.1, 5.2 y 5.3 se muestran los resultados obtenidos de los modelos iniciales que han sido entrenados con las imágenes en formato JPG y sin preprocessamiento previo (es decir, el "Modelo inicial" y "2 Drop-batch"), se pueden observar claras diferencias cuando aplicamos *Data Augmentation* o aumento de datos en el tercer modelo, "3 Drop-Batch". Al aplicar esta estrategia, vemos en la Tabla 5.2 de los resultados de validación que el máximo *AUC* que hemos obtenido es de 85,98% con un 34,56% de *loss*.

En el primer modelo sobre el que hemos realizado el estudio, el "Modelo inicial", podemos observar que sobreajusta, ya que a la hora de entrenar, en la tabla 5.1, el *AUC* resultante es del 99%, y su *loss* es de 0,11 mientras que tanto la validación y evaluación del modelo se mantienen en 95,9% el *AUC* y 0,41 de *loss* en ambos casos. Esto se debe a que no hemos aplicado ningún tipo de capas de *Dropout* y *Batch Normalization* como hemos comentado anteriormente.

En cambio, observamos que el modelo "2 Drop-Batch" solamente introduciendo técnicas de *Dropout* y *Batch Normalization* y sin aplicar ninguna técnica de aumento de datos, es el modelo que mejor resultados nos proporciona. Vemos que a la hora de entrenar el *AUC* llega hasta un 98,54% y el *loss* se queda en 0,1515, en los conjuntos de validación y evaluación, los resultados son muy similares, el *AUC* es de un 97,15% y el *loss* es de un 0,3392 para el conjunto de validación.

En el caso del entrenamiento, el conjunto de datos está completamente balanceado, tal y como podemos observar en la Figura 2.13, es por eso por lo que la *precisión* nos da un valor de 95,81%, 95,47% y 87,35% para los modelos 1, 2 y 3, respectivamente.

Sin embargo, una vez pasamos a validar y evaluar los conjuntos, debemos tener presente el desbalanceo que existe sobre estos conjuntos, desarrollado en el Capítulo 2.4.1, dando lugar a una baja precisión.

Las conclusiones que podemos obtener de este tipo de modelos, es que a pesar de introducir aumento de datos sobre las imágenes, no les favorece, esto se puede dar debido a que las versiones ligeramente modificadas de las imágenes que produce el uso de esta técnica, pueden dificultar la generalización del modelo para capturar patrones consistentes.

Por tanto, el mejor modelo obtenido será el segundo que hemos realizado, cuyos resultados en la evaluación del modelo han sido: $AUC = 97,1\%$, $loss = 0,342$, $precision = 26,2\%$, $recall = 97,9\%$.

A continuación, veremos los resultados de este experimento, indicando que él no uso de *Dropout* y *Batch Normalization* generan un sobreajuste en la red, por lo que a pesar de que los resultados sean buenos a simple vista cuando entrenamos, a la hora de validar y evaluar vemos que no es así.

Id modelo	lr inicial	Aumento de datos	AUC_train	Loss_Train	precision	recall
Modelo inicial	1^{-3}	No	0,9918	0,1122	0,9569	0,9581
2 Drop-Batch	1^{-3}	No	0,9854	0,1515	0,9245	0,9547
3 Drop-Batch	1^{-3}	Si	0,8904	0,4042	0,7603	0,8735

Tabla 5.1: Entrenamiento modelos simples.

Id modelo	AUC_val	Loss_val	val_precision	val_recall
Modelo inicial	0,9588	0,4109	0,3434	0,9131
2 Drop-Batch	0,9715	0,3392	0,2600	0,9808
3 Drop-Batch	0,8598	0,3456	0,1947	0,6001

Tabla 5.2: Validación modelos simples.

Id modelo	AUC_test	Loss_test	test_precision	test_recall
Modelo inicial	0,959	0,413	0,347	0,914
2 Drop-Batch	0,971	0,342	0,262	0,979
3 Drop-Batch	0,8606	0,3471	0,1948	0,5944

Tabla 5.3: Evaluación modelos simples.

5.3.2. Transfer Learning

Una vez hemos experimentado con modelos de Redes Convolucionales cuyas capas han sido implementadas desde cero, pasaremos a realizar un estudio sobre algunos modelos de *Transfer Learning* que hemos llevado a cabo.

Es una técnica que aprovecha los conocimientos adquiridos por un modelo previamente entrenado en una tarea relacionada para mejorar el rendimiento en una tarea nueva y distinta. En lugar de entrenar un modelo desde cero en la tarea de interés, se utiliza un modelo preentrenado como punto de partida y se adapta a los nuevos datos.

La principal ventaja del *Transfer Learning* es que permite aprovechar los conocimientos y representaciones aprendidos en tareas previas, lo que puede resultar en un entrenamiento más rápido y un mejor rendimiento con menos datos en la tarea nueva.

Hemos experimentado con distintos modelos para cada caso, por tanto, al final de esta sección, seleccionaremos cuál será el mejor modelo de *Transfer Learning* que hemos realizado.

VGG19

VGG19 es una red convolucional creada por Karen Simonyan y Andrew Zisserman en 2014 [34]. Esta red presenta características mejoradas con respecto a su anterior modelo, VGG16. Los números 16 y 19 se refieren al número de capas con pesos que presenta cada red.

Es una arquitectura que ha sido entrenada con más de un millón de imágenes de la base de datos *Imagenet* [36]. En la Figura 5.2 podemos observar como está compuesta su arquitectura [34].

La VGG19 ha sido ampliamente utilizada en la comunidad de visión por computadora y ha demostrado un rendimiento sobresaliente en una variedad de tareas, como clasificación de imágenes, detección de objetos y segmentación semántica. Debido a su simplicidad y efectividad, la VGG19 se ha convertido en un punto de referencia común en la investigación y se ha utilizado como base para el desarrollo de otras arquitecturas más avanzadas.

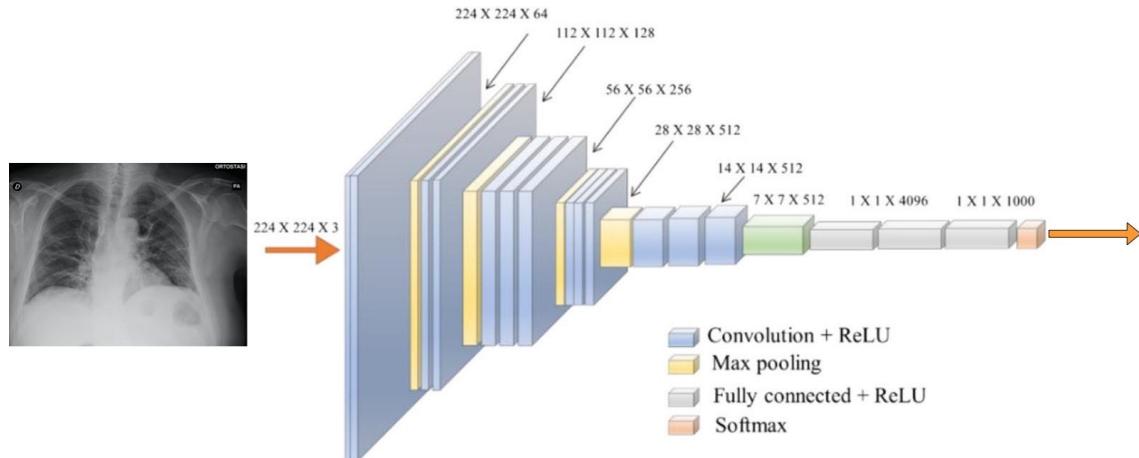


Figura 5.2: Arquitectura VGG19

Resultados

En las Tablas 5.4, 5.5 y 5.6 vemos que las imágenes transformadas con el Filtro de Sobel nos proporcionan resultados muy poco óptimos. El filtro de Sobel se utiliza comúnmente en el procesamiento de imágenes para resaltar los bordes y gradientes de una imagen. Al aplicar este filtro a las imágenes de entrada antes de pasarlas a la red VGG19, se introducen cambios significativos en la estructura y contenido de las imágenes. Esto puede llevar a una degradación en el rendimiento de la red, ya que los filtros de convolución de la VGG19 están optimizados para extraer características específicas de las imágenes originales. Por tanto, al modificar la apariencia de las imágenes con el filtro de Sobel, se pueden introducir ruido o información no relevante que afecta la capacidad de la red para reconocer y clasificar correctamente los objetos.

En el caso de la red VGG19, se ha demostrado que funciona mejor con imágenes en su forma original, sin aplicar transformaciones o filtros adicionales.

Vemos que en la parte de entrenamiento el *AUC* es de 95,18% y el *loss* es de un 0,2431, en cambio, al aplicar las imágenes con el filtro de Sobel, el *AUC* es de 49,92% y el *loss* de 0,6932. Lo que quiere decir que este modelo no está aprendiendo nada al aplicarle el filtro de preprocesamiento que hemos elegido.

Al igual que en la Sección 5.3.1, vemos que la *precisión* y el *recall* en la fase de entrenamiento es de un 88,06% y 94,12%, respectivamente en el modelo sin Filtro de Sobel. En cambio, a la hora de validar y evaluar, vemos que los valores de la *precisión* se reducen hasta un 23,61% en el conjunto de validación y 24,08% en el conjunto de prueba.

Las conclusiones que podemos obtener de este tipo de modelos, es que los resultados inferiores obtenidos al aplicar el Filtro de Sobel en las imágenes con la arquitectura VGG19 se deben a la incompatibilidad entre las características extraídas por el filtro y las características aprendidas por la red neuronal. Por tanto, es recomendable utilizar las imágenes en su forma original para obtener mejores resultados con la VGG19 en tareas de clasificación de imágenes, y sobre todo al ser imágenes que presentan una alta complejidad para identificar donde se encuentra la embolia.

A continuación, veremos los resultados de los dos modelos que hemos realizado. Observaremos la gran diferencia entre las imágenes JPG normales y en el segundo modelo que ha sido entrenado con las imágenes con filtro de Sobel que hemos aplicado.

Id modelo	lr inicial	Filtro de Sobel	AUC_train	Loss_Train	precision	recall
VGG19 - 1º	1 ⁻³	No	0.9518	0.2431	0.8806	0.9412
VGG19 - 2º	1 ⁻³	Si	0.4992	0.6932	0.4989	0.4758

Tabla 5.4: Entrenamiento modelos VGG19.

Id modelo	AUC_val	Loss_val	val_precision	val_recall
VGG19 - 1º	0.9290	0.3516	0.2361	0.9055
VGG19 - 2º	0.5000	0.6870	0.0000e+00	0.0000e+00

Tabla 5.5: Validación modelos VGG19.

Id modelo	AUC_test	Loss_test	test_precision	test_recall
VGG19 - 1º	0.9298	0.3500	0.2408	0.9093
VGG19 - 2º	0.5000	0.6871	0.0000e+00	0.0000e+00

Tabla 5.6: Evaluación modelos VGG19.

Xception

En esta sección haremos uso de un nuevo modelo de *Transfer Learning*, llamado Xception.

Xception es una arquitectura de red neuronal convolucional que destaca por su eficiencia y capacidad de aprendizaje profundo. Fue propuesto por François Chollet, el creador de la biblioteca Keras, en 2016 [30].

El nombre "Xception" es una abreviatura de "Extreme Inception", ya que el modelo se basa en el concepto de los módulos Inception utilizados en la arquitectura Inception-v3 [1]. Sin embargo, Xception lleva este concepto un paso más allá al aplicar convoluciones separables en lugar de convoluciones estándar.

Las convoluciones separables se utilizan para dividir el proceso de convolución en dos etapas: una convolución espacial (también llamada convolución en 2D) y una convolución de punto (también llamada convolución en 1x1). Esta separación permite reducir significativamente el número de hiperparámetros entrenables y la carga computacional, lo que a su vez mejora la eficiencia del modelo y reduce el riesgo de sobreajuste.

En la Figura 5.3 podemos ver de que se compone la arquitectura de Xception.

El modelo Xception ha demostrado un rendimiento sobresaliente en diversas tareas de visión por computadora, como la clasificación de imágenes, la detección de objetos y la segmentación semántica. Además de su eficiencia, se destaca por su capacidad para capturar características de diferentes escalas y niveles de abstracción, lo que lo convierte en una elección popular para aplicaciones de procesamiento de imágenes complejas.

El segundo modelo, estará compuesto por una variante del modelo Xception, llamado SeXception. Este modelo incorpora un bloque de atención llamado "*Squeeze-and-Excitation*" (*Se*).

El bloque SE se compone de dos etapas principales. En la etapa de "*squeeze*", se realiza una reducción global de la dimensionalidad de las características de entrada a través de una operación de max pooling. Esto permite obtener una representación global de la información en cada canal de características. En la etapa de "*excitation*", se utilizan capas totalmente conectadas para aprender la importancia relativa de cada canal de características. Esta información de importancia se utiliza para reescalar las características originales, otorgando más peso a los canales más relevantes [27].

Al incorporar el bloque SE en el modelo Xception, el modelo SeXception es capaz de adaptar y enfocar su atención en los canales de características más importantes, lo que puede mejorar el rendimiento y la capacidad de representación del modelo en tareas de visión por computadora. A continuación, en el Código 5.1 podemos observar el bloque implementado.

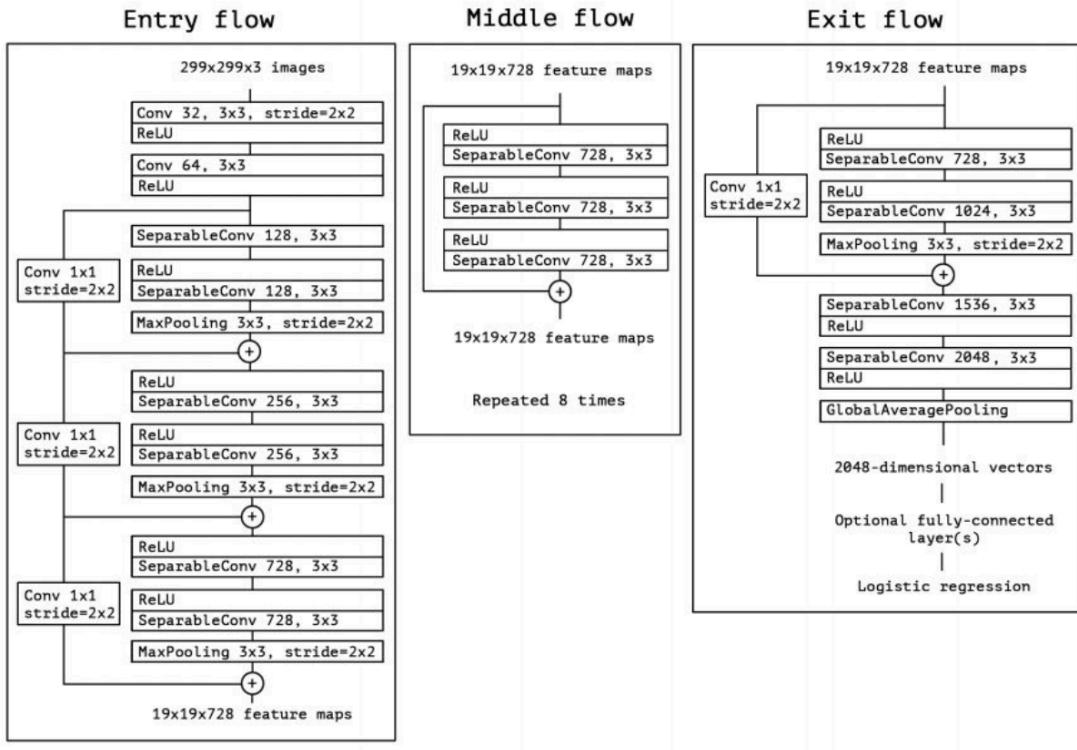


Figura 5.3: Arquitectura Xception

```
def squeeze_excite_block(input_tensor, ratio=16):
    channels = input_tensor.shape[-1] # Obtenemos el número de canales
    # Squeeze: Reducción de dimensionalidad
    x = GlobalAveragePooling2D()(input_tensor)
    x = Reshape((1, 1, channels))(x)
    x = Dense(channels // ratio, activation='relu')(x)
    x = Dense(channels, activation='sigmoid')(x)
    # Excite: Actualización de características
    x = multiply([input_tensor, x])
    return x
```

Código 5.1: Código de *Squeeze-and-Excitation* block

Resultados

Como podemos observar, en las Tablas 5.7, 5.8 y 5.9, los resultados no varían mucho. Vemos que en la fase de entrenamiento el modelo Xception es ligeramente mejor que el SeXception, ya que nos proporciona un *AUC* del 92,01% con respecto al 91,64% de el segundo modelo.

Pasando a las fases de validación y evaluación de los modelos, vemos que el modelo SeXception en la última fase, presenta un mayor *loss*, con respecto al primer modelo Xception, por tanto, el mejor modelo de los dos, será el Xception.

Las conclusiones que sacamos al realizar este tipo de modelos, es que no presentan una diferencia significativa en términos de rendimiento. El problema no se beneficia de la fina adaptación en las características de los canales proporcionada por el bloque "*Squeeze-and-Excitation*". A continuación, veremos los resultados de los modelos Xception y SeXception que hemos implementado.

Id modelo	lr inicial	Aumento de datos	AUC_train	Loss_Train	precision	recall
Xception	1^{-3}	No	0.9201	0.3429	0.7726	0.9398
SeXception	1^{-3}	No	0.9164	0.3474	0.7567	0.9522

Tabla 5.7: Entrenamiento de los modelos Xception y SeXception.

Id modelo	AUC_val	Loss_val	val_precision	val_recall
Xception	0.9062	0.3433	0.1712	0.8850
SeXception	0.8927	0.3904	0.1499	0.9251

Tabla 5.8: Validación de los modelos Xception y SeXception.

Id modelo	AUC_test	Loss_test	test_precision	test_recall
Xception	0.9074	0.3446	0.1747	0.8913
SeXception	0.8927	0.3917	0.1515	0.9238

Tabla 5.9: Validación de los modelos Xception y SeXception.

ResNet50

ResNet50 es un modelo de red neuronal convolucional profunda que se basa en la arquitectura ResNet (Residual Network). ResNet fue desarrollada por Kaiming He y su equipo en Microsoft Research en 2015 [13].

Es una red convolucional formada por 50 capas (48 capas de convolución, una capa de MaxPool y una capa de agrupación promedio o *average pooling*).

La característica principal de ResNet es el uso de conexiones residuales, también conocidas como conexiones de salto o conexiones de atajo. Estas conexiones permiten que los datos fluyan directamente desde las capas de entrada hasta las capas posteriores, saltando algunas capas intermedias.

Esto ayuda a mitigar el problema de desvanecimiento del gradiente que puede ocurrir en redes neuronales muy profundas, permitiendo el entrenamiento efectivo de redes más profundas y mejorando su capacidad para capturar características más complejas [13].

Resultados

En este caso, hemos llevado a cabo 3 experimentos, cuyos resultados se encuentran en las Tablas 5.10, 5.11 y 5.12. En primer lugar, tenemos el primer modelo que ha sido entrenado con un *learning rate* inicial de 1^{-3} con el *optimizer Adam* y con las imágenes originales que presentamos.

En segundo lugar, esta vez hemos utilizado las imágenes con el Filtro de Sobel, y vemos que aumentamos el *AUC* con respecto al primer modelo.

En tercer lugar, tenemos el último experimento de esta sección, el cual ha sido entrenado con un *learning rate* inicial de 1^{-4} con el *optimizer Adam* y con las imágenes JPG originales que presentamos. Vemos que en comparación con el entrenamiento de los otros dos modelos, este tercero es el que mejor resultados nos proporciona, con un $AUC = 99,85\%$ y un $loss = 0,497$.

Vemos que el modelo que mejor valores nos proporciona en los experimentos sobre los conjuntos de validación y prueba, es el modelo 1. Su *AUC* es aproximadamente un 96% y su *loss* es de un 0,38. Es importante comentar que el tercer modelo que hemos realizado, nos proporciona un mayor *AUC*, aproximadamente un 97%, pero en cambio, nos aparece más *loss* que en el primero, con un 0,44, por tanto, no será tan eficiente como el primero de todos.

A continuación, veremos los resultados obtenidos de la ejecución de estos experimentos.

Id modelo	lr inicial	Filtro de Sobel	AUC_train	Loss_Train	precision	recall
ResNet50 - 1º	1^{-3}	No	0.9949	0.0878	0.9643	0.9658
ResNet50 - 2º	1^{-3}	Si	0.9956	0.0815	0.9640	0.9720
ResNet50 - 3º	1^{-4}	No	0.9985	0.0497	0.9793	0.9853

Tabla 5.10: Entrenamiento de los modelos ResNet50.

Id modelo	AUC_val	Loss_val	val_precision	val_recall
ResNet50 - 1º	0.9586	0.3837	0.3178	0.9168
ResNet50 - 2º	0.9536	0.2392	0.4343	0.7961
ResNet50 - 3º	0.9649	0.4382	0.3096	0.9574

Tabla 5.11: Validación de los modelos ResNet50.

Id modelo	AUC_test	Loss_test	test_precision	test_recall
ResNet50 - 1º	0.9597	0.3837	0.3214	0.9166
ResNet50 - 2º	0.9525	0.2415	0.4370	0.7991
ResNet50 - 3º	0.9656	0.4385	0.3137	0.9592

Tabla 5.12: Validación de los modelos ResNet50.

ResNet50 + XGBoost

Una vez conocemos los resultados de todos nuestros modelos *Transfer Learning* (en la Tabla 5.16 se realiza una comparativa de los modelos), podemos concluir que el que mejor resultados nos ha proporcionado es el ResNet50, por ello, llevaremos a cabo un estudio de este modelo de *deep learning* y un *ensemble* XGBoost.

XGBoost es un método de aprendizaje automático supervisado para clasificación y regresión. XGBoost es la abreviatura de las palabras inglesas "*extreme gradient boosting*" (refuerzo de gradientes extremo).

La idea detrás del *boosting* es generar múltiples modelos de predicción “débiles” secuencialmente, y que cada uno de estos tome los resultados del modelo anterior, para generar un modelo más “fuerte”, con mejor poder predictivo y mayor estabilidad en sus resultados [37].

Lo que significa que la combinación de estas dos estrategias puede ser prometedora para mejorar el rendimiento y la generalización en un problema de visión artificial, en la Figura 5.4 podemos ver en que consistirá este modelo.

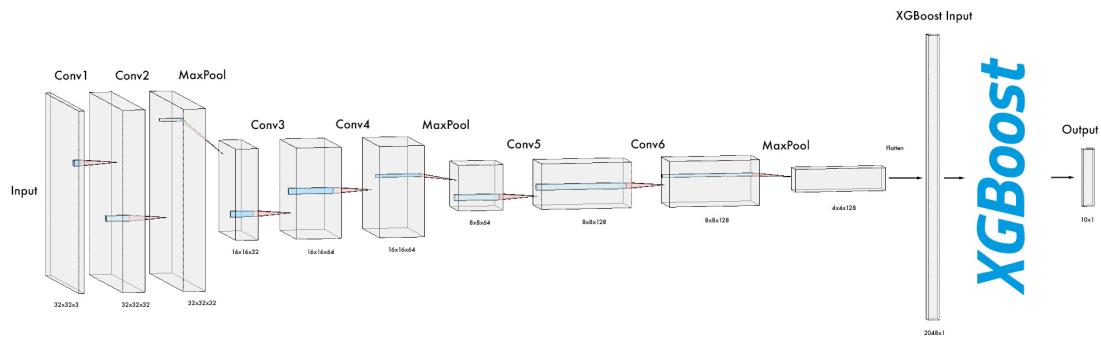


Figura 5.4: Arquitectura modelo de Transfer Learning + XGBoost.

El proceso que hemos seguido para implementar este experimento ha sido el siguiente:

- Cargamos el modelo ResNet50 ya entrenado, en este caso, hemos realizado este estudio con el modelo número 1 de la Sección 5.3.2.
- Extraemos las características de las imágenes del modelo ResNet50, donde ahora el modelo tendrá las capas de salida eliminadas, y nos quedaremos con la salida de las capas anteriores.
- Obtenemos las características y las variables a predecir de las imágenes procedentes de los conjuntos de entrenamiento, validación y prueba. Para extraer las características de las imágenes deberemos utilizar el modelo con las capas modificadas.
- Una vez hemos realizado estos pasos, ya podemos definir los hiperparámetros del ensemble XGBoost, y comenzar a entrenar el modelo.

Resultados

En este experimento hemos realizado 6 experimentos distintos, cuyos resultados se pueden visualizar en las tablas 5.13, 5.14 y 5.15. Hemos variado los hiperparámetros, del modelo XGBoost, entre los que nos podemos encontrar la profundidad máxima de cada árbol (*max depth*), tasa de aprendizaje (*eta*) y el número de árboles, los iremos cambiando en cada modelo, y así observar cuál es la mejor configuración.

Los resultados del entrenamiento son muy similares en los 6 modelos, los que más destacan son los dos primeros modelos que con un *AUC* del 98,07% y 98,1%, respectivamente. Es de destacar el bajo *loss* que obtenemos en todos y cada uno de los experimentos que hemos realizado, pero sobre todo en los dos primeros modelos, con 0,0192 y 0,0189.

Al igual que en el entrenamiento, los resultados proporcionados en las tablas de validación y evaluación son también muy similares, destacando sobre todo los dos primeros modelos, al igual que en la fase de entrenamiento.

En conclusión, podemos destacar la baja cantidad de *loss* que presenta cada modelo en su fase de evaluación, siendo el máximo valor un 0,1396. Lo que significa que la combinación de modelos de *transfer learning* con *ensembles* son todo un éxito y mejoran en gran cantidad el rendimiento en problemas de visión artificial.

A continuación, veremos los resultados y los hiperparámetros de cada uno de los modelos con los que hemos experimentado.

Max depth	eta	Nº arboles	AUC_train	Loss_Train	precision	recall
3	0.1	100	0.9807	0.0192	0.9807	0.9808
5	0.1	100	0.981	0.0189	0.9812	0.9807
3	0.01	100	0.9743	0.0256	0.9698	0.9792
5	0.01	100	0.9802	0.0197	0.9787	0.9819
3	0.01	400	0.9743	0.0256	0.9698	0.9792
5	0.01	400	0.9802	0.0197	0.9787	0.9819

Tabla 5.13: Entrenamiento de los modelos ResNet50 + XGBoost.

AUC_val	Loss_val	val_precision	val_recall
0.9039	0.1228	0.2935	0.9340
0.9038	0.1226	0.2938	0.9336
0.8961	0.1403	0.2664	0.9369
0.9027	0.1275	0.2859	0.9367
0.8961	0.1403	0.2664	0.9369
0.9027	0.1275	0.2859	0.9367

Tabla 5.14: Validación de los modelos ResNet50 + XGBoost.

AUC_test	Loss_test	test_precision	test_recall
0.9045	0.1229	0.2968	0.9354
0.9044	0.1229	0.2969	0.9350
0.8983	0.1396	0.2711	0.9408
0.9034	0.1276	0.2892	0.9382
0.8983	0.1396	0.2711	0.9408
0.9034	0.1276	0.2892	0.9382

Tabla 5.15: Evaluación de los modelos ResNet50 + XGBoost.

5.4. Conclusiones y futuras mejoras

Después de haber experimentado con numerosos modelos, se han llegado a varias conclusiones.

La principal conclusión es que la combinación de un modelo *transfer learning* junto con un *ensemble*, proporcionan unos resultados muy óptimos en esta clase de problemas tan complejos para detectar embolismos pulmonares.

Como propuesta de futuro, se pueden llevar a cabo mejoras en varios aspectos, a pesar de no haber obtenido malos resultados, se pueden apurar mucho más. Se puede realizar una extracción de características más detalladas para probar a usar modelos que no requieren el coste computacional de una red neuronal, ya que la gran mayoría de estos modelos han tardado hasta 8 horas en ejecutarse. Por el contrario, también se puede seguir experimentando con redes neuronales hasta afinar lo suficiente como para obtener mejores resultados.

Para finalizar esta sección se obtendrá una comparativa de los mejores modelos para cada técnica escogida y su conjunto de datos correspondiente. Los resultados de dichos modelos se pueden observar en las Tablas 5.16 y 5.17.

Tal y como se puede comprobar, el modelo que mejor *AUC* ofrece es en el que hemos utilizado la CNN diseñada desde cero, sobre las imágenes sin preprocesamiento y con capas de *Drop-Out* y *BatchNormalization*, con unos resultados de 97,1% de *AUC* y un 0,342 de *loss*.

También hay que destacar que el mejor modelo de *Transfer Learning* es el ResNet50 con un 95,97% de *AUC* y un 0,3837 de *loss*.

Modelo	lr inicial	Filtro de Sobel	Aumento de datos	AUC_test	Loss_test
CNN-Drop-Batch	1^{-3}	No	No	0,971	0,342
VGG19-1°	1^{-3}	No	No	0.9298	0.3500
Xception	1^{-3}	No	No	0.9074	0.3446
ResNet50-1°	1^{-3}	No	No	0.9597	0.3837

Tabla 5.16: Resumen mejores modelos tras la fase de evaluación.

Modelo	Max depth	eta	Nº arboles	AUC_test	Loss_test
ResNet50 + XGBoost	3	0.1	100	0.9045	0.1229

Tabla 5.17: Mejor modelo ResNet50 + XGBoost tras la fase de evaluación.

Capítulo 6. Desarrollo de la aplicación

6.1. Introducción

Este capítulo constará del desarrollo de la aplicación que hemos llevado a cabo, donde se expondrán las tecnologías y las herramientas utilizadas en el proceso. También definiremos a qué público va dirigido esta aplicación y con qué funcionalidades se ha dotado a la misma.

Esta parte del documento corresponde a la última fase de la metodología CRISP-DM descrita en el Capítulo 2.6. La aplicación se construirá a partir de los mejores modelos seleccionados en el Capítulo 5.

Antes de comenzar, es importante destacar que para este proyecto no se ha llevado a cabo un proceso software exhaustivo como recogida de requisitos, seguimiento de una metodología específica, evaluarlo con usuarios reales, etc. La aplicación creada actuará solamente de interfaz sobre el modelo que escojamos en el Capítulo 5.

6.2. Diseño de la aplicación

El objetivo principal de esta aplicación es proporcionar a los usuarios la habilidad de adjuntar un escaneo CTPA del tórax del paciente, de tal forma que nos devuelva la probabilidad de la presencia de embolias pulmonares. El escaneo CTPA deberá subirse en formato .JPG, ya que las imágenes en DICOM es un formato poco accesible para los usuarios y complicado de trabajar, así simplificaremos el uso de la aplicación para los usuarios.

6.3. Flask

Flask es un framework web ligero y flexible escrito en Python que se utiliza para construir aplicaciones web. Proporciona herramientas y bibliotecas que permiten crear rápidamente aplicaciones web y API (interfaces de programación de aplicaciones) utilizando el lenguaje Python, mientras que la estructura de la página web se realizará mediante elementos jerárquicos, deberá ser creada con HTML y el diseño del estilo de la página será con CSS [5].

El preprocesamiento ha sido realizado en Python con la librería TensorFlow, por ello podremos reutilizar gran parte del código a la hora de hacer las predicciones o incluso cargar los modelos preentrenados. Esta librería permite guardar los modelos entrenados con distintos tipos de extensiones ".h5" o ".tf".

En estos ficheros guardaremos toda la información relacionada con el modelo que hemos creado, incluyendo los pesos del mismo. El proceso que finalmente nosotros seguiremos será, crear los modelos en el entorno Kaggle, guardar estos modelos en formato ".tf", e integrarlos directamente en la aplicación web.

6.4. Gradient-weighted Class Activation Mapping

El *deep learning* nos ha facilitado una precisión sin precedentes en la clasificación, segmentación y detección de objetos, pero uno de los mayores problemas es la interpretabilidad del modelo, un componente esencial en este tipo de proyectos.

En la práctica, los modelos de *deep learning* son tratados como métodos de "caja negra", en los que muchas veces no tenemos claro donde está mirando la red neuronal a la hora de introducir una imagen o qué neuronas se han activado durante la predicción.

Para solucionar estos problemas y ayudar a las personas que practican continuamente con este tipo de metodologías en sus proyectos, Ramprasaath R. Selvaraju junto con otros Co-Autores, crearon Gradient-weighted Class Activation Mapping, o más conocido como, Grad-CAM [32].

Grad-CAM busca qué partes de la imagen han llevado a una red neuronal convolucional a su decisión final. Este método consiste en producir mapas de calor que representan las clases de activación sobre las imágenes recibidas de entrada. Una clase de activación se asocia a una clase de salida específica, y dependiendo de la clase de salida que elija aumentará o disminuirá la intensidad del pixel.

Por ejemplo, si una imagen se utiliza en una red convolucional de perros y de gatos, la visualización de Grad-CAM permite generar un *heatmap* (mapa de calor) para la clase “gato”, indicando en qué medida corresponden las diferentes partes de la imagen a un gato, y también un mapa de calor para la clase “perro” que indique en qué medida corresponden las partes de la imagen a un perro. En la Figura 6.1 podemos ver el resultado de aplicar esta técnica.

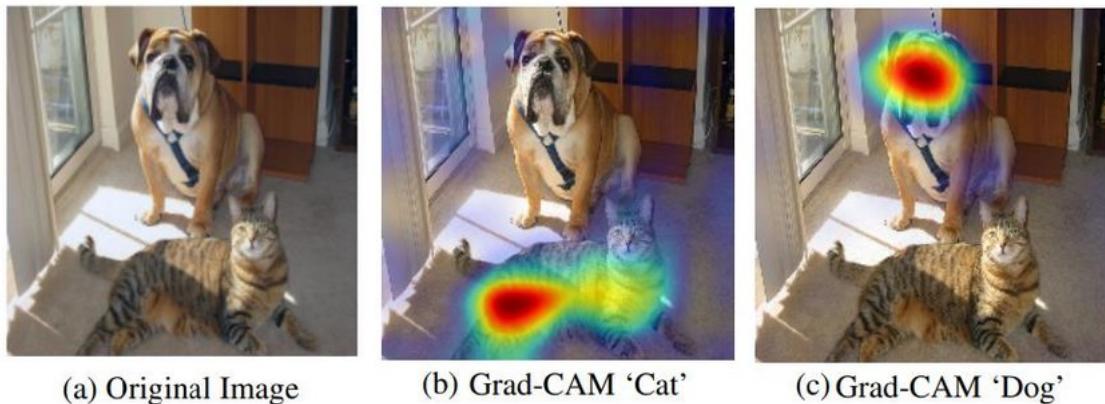


Figura 6.1: Grad-CAM de un perro y un gato

6.5. Aplicación web

Para finalizar la última fase de la metodología CRIPS-DM descrita en el Capítulo 2.6, llevaremos a cabo una pequeña descripción de los distintos componentes de la aplicación que hemos llevado a cabo.

En primer lugar, es necesario comentar que la interfaz que hemos desarrollado es bastante simple. Está compuesta por un botón donde podamos seleccionar los ficheros que deseemos para clasificarlos, posteriormente deberemos seleccionar el modelo que queremos usar para predecir y clasificar la imagen.

En la Figura 6.2, vemos una representación gráfica de como se ve la interfaz una vez carguemos la página web. A la derecha tenemos la leyenda de los colores que vamos a usar, la cual nos ayudará a hacernos una idea de en que lugares se ha fijado más el modelo seleccionado a la hora de entrenar.

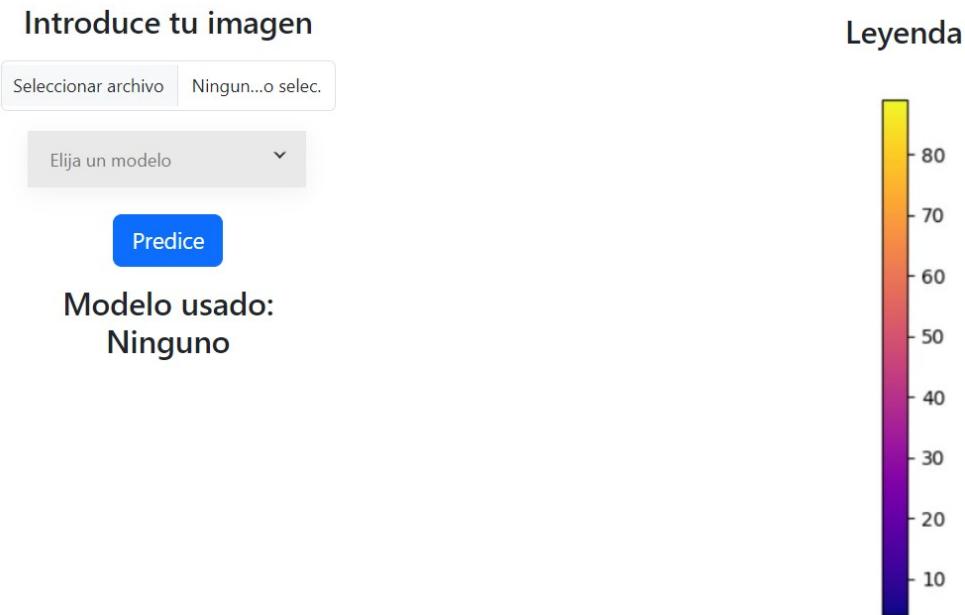


Figura 6.2: Interfaz de la aplicación.

Una vez hayamos seleccionado la imagen que queremos clasificar y predecir, debemos seleccionar el modelo a usar, y así la aplicación sea capaz de ejecutar el modelo preentrenado que hemos implementado.

En caso de que no hayamos seleccionado ningún modelo, nos aparecerá un mensaje en rojo indicando que debemos seleccionar alguno, como se puede ver en la Figura 6.3.

Introduce tu imagen

Seleccionar archivo Ningun...o selec.

Elija un modelo

Debes seleccionar un modelo para predecir

Predice

Modelo usado:
Ninguno

Figura 6.3: Mensaje de error.

Si se cumplen las condiciones de haber seleccionado una imagen y un modelo, pulsaremos el botón de "predecir". Donde nos aparecerá el porcentaje de probabilidad de embolia que tiene la imagen, y como hemos comentado en la Sección 6.4, a la derecha de la imagen original, nos aparecerá la imagen con la técnica Grad-CAM. Tal y como podemos ver en la Figura 6.4.

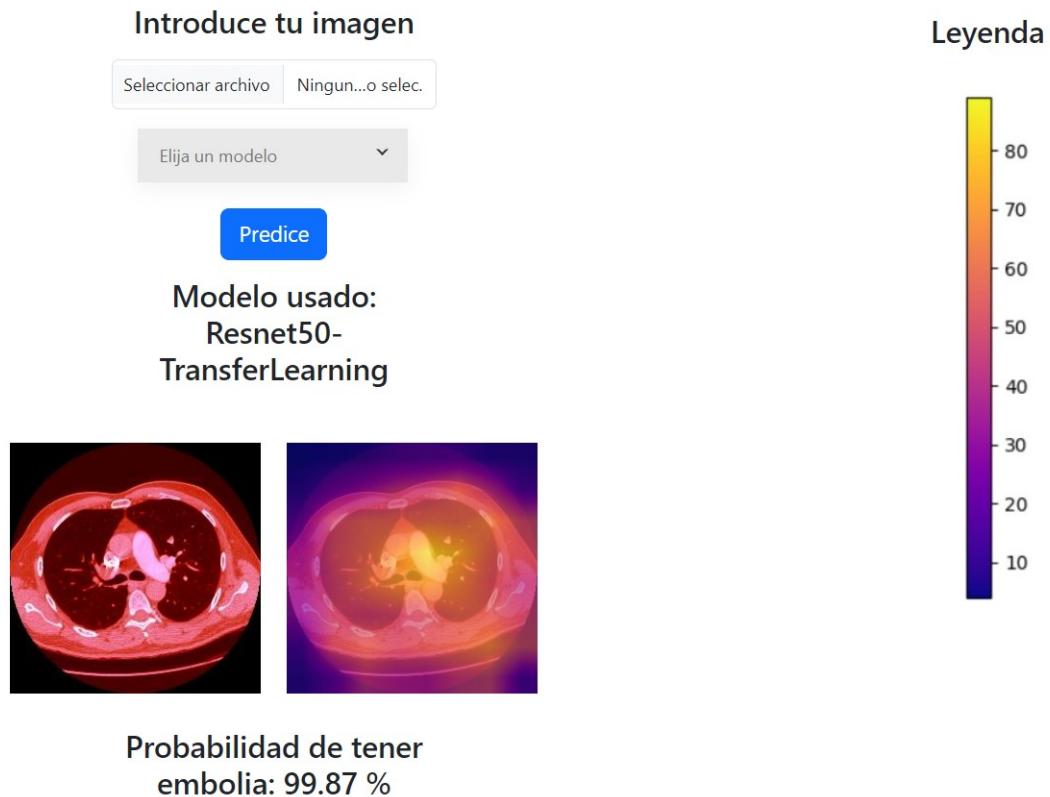


Figura 6.4: Interfaz de la aplicación con una imagen que presenta embolia.

Como vemos, los colores de la leyenda que son más amarillos, significa que el modelo es donde más se ha fijado a la hora de predecir la decisión final. Vemos que donde más se centra el modelo es en la parte central y la parte derecha del Tórax del paciente, lo que nos puede dar una pequeña idea de donde se encuentra el embolismo pulmonar.

En cambio, si ahora introducimos una imagen que no presenta embolia, en la Figura 6.5 vemos el resultado.

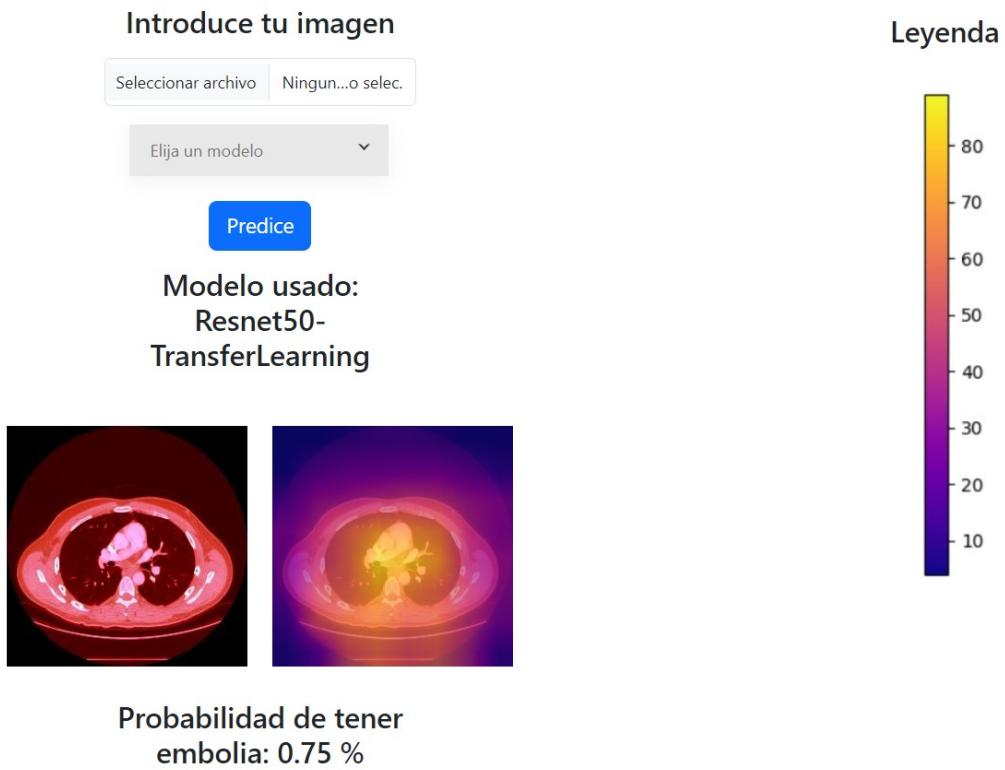


Figura 6.5: Interfaz de la aplicación con una imagen no que presenta embolia.

Vemos que el modelo se ha fijado en la parte central del Tórax del paciente, al igual que antes, pero esta vez vemos que ha recogido información de toda la imagen, sin centrarse en un punto característico como la izquierda, o la derecha, clasificando la imagen como que no presenta embolia aparente.

6.6. Conclusiones y futuras mejoras

Realizada la aplicación con las funcionalidades que se han implementado, se puede concluir que la interfaz cumple con los objetivos propuestos. No obstante, como mejora para el futuro se deberá desplegar el modelo en un servidor en la nube, y así no depender de un servidor que solamente tiene actividad localmente en nuestro ordenador, donde además se requiere tener instalado el software requerido para su uso.

También algunas mejoras a implementar podrían ser:

- Ser capaz de introducir un solo paciente con todas sus imágenes, ordenarlas y que detecte donde tiene la embolia a través una serie de *slices* superpuestas una encima de otra, de tal forma que se puedan ver los cambios que se producen en la corriente sanguínea.

- Señalar exactamente donde se encuentra la embolia en caso de tenerla.
- Predecir el resto de características de la imagen.
- Realizar pruebas con usuarios reales, y a través de su *feedback* añadir funcionalidad o modificar las ya existentes, y así tener un producto que satisfaga los requisitos.

Capítulo 7. Conclusiones

En este capítulo final se manifestarán las conclusiones obtenidas tras la realización de este proyecto. Pese a que en cada Parte se han ido obteniendo unas conclusiones de forma individual (Sección 5.4 y 6.6) en este último capítulo se expondrán todas las conclusiones obtenidas de forma conjunta. Adicionalmente, se verán las competencias de la intensificación de *Computación* del Grado en Ingeniería Informática que se han trabajado y se enumeraron en la Sección 1.3.

7.1. Conclusiones y trabajos futuros

Al comienzo de este proyecto se estudiaba la peligrosidad que puede suponer la presencia de embolias pulmonares. Motivados por esto, se decidió hacer uso de las posibilidades que brinda el *deep Learning* para intentar facilitar los medios que se utilizan en la medicina actual y así crear una aplicación para que sea utilizable por cualquier persona.

Tras realizar una iteración completa siguiendo la metodología CRISP-DM para la creación de un modelo y así obtener una demo de la aplicación, se concluye que es un problema resoluble por medio de técnicas de aprendizaje profundo.

La aplicación creada muestra la efectividad de los modelos generados para detectar la presencia de la enfermedad. No obstante, aún tiene mucho margen de mejora, como aumentarle las funcionalidades y realizar un proceso software acorde para obtener el mejor producto posible.

Por lo tanto, tras la realización de este trabajo se determina que las embolias pulmonares se pueden prevenir y tratar a tiempo haciendo uso de tecnologías de la información y de inteligencia artificial.

7.2. Competencias adquiridas

En la Sección 1.3 se listaban las competencias específicas de la tecnología de Computación que se esperaban trabajar en este proyecto. Una vez concluido este se verán cuáles competencias y cómo se han desarrollado.

- Capacidad para **evaluar la complejidad computacional** de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

Esta competencia ha sido tratada en el Capítulo 5 donde se han propuesto diversos modelos con los que experimentamos basándonos en las conclusiones obtenidas en el análisis de los datos (Capítulo 3).

- Capacidad para **conocer los fundamentos**, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.

Tratada con el desarrollo de la aplicación en el Capítulo 6. Previamente, se ha realizado un estudio comparativo que ha permitido seleccionar los modelos basados en deep learning a incluir en la aplicación.

- Capacidad para **adquirir, obtener, formalizar y representar el conocimiento humano** en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes entornos inteligentes.

Competencia tratada en el preprocesamiento de los datos explicado en el Capítulo 4.

- Capacidad para **conocer y desarrollar técnicas de aprendizaje computacional** y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

El estudio y creación de los modelos se realiza en el Capítulo 5, siendo capaces de procesar la consulta del paciente usando las imágenes que presenta cada uno.

Capítulo A. Planificación del proyecto

Para este TFG se ha seguido la metodología explicada en la Sección 2.6 dónde solo se ha llevado una iteración del ciclo de vida del proyecto. La duración de cada fase en el tiempo de desarrollo de este proyecto se muestra en la Figura A.1.

A continuación, detallaremos la duración de cada una de las fases que hemos realizado.

1. **Comprensión del negocio.** Esta fase del proyecto corresponde al Capítulo 1. Los objetivos y requisitos del proyecto fueron definidos en las primeras semanas del proyecto (*Febrero 2023*) en las reuniones que se realizaban de forma periódica con los tutores.
2. **Comprensión de los datos.** Esta fase corresponde a los Capítulos 2 y 3. El estudio se realizó desde mediados de *Febrero 2023* hasta finales del mismo mes.
3. **Preparación de los datos.** Esta fase corresponde al Capítulo 4. El preprocesamiento de los datos se realizó desde finales de *Febrero 2023* hasta principios de *Mayo 2023*.
4. **Modelado.** Esta fase corresponde al Capítulo 5. Este proceso se realizó desde finales del mes de *Abril 2023* hasta finales de *Mayo 2023*.
5. **Evaluación.** Esta fase corresponde al Capítulo 5. Se llevó a cabo desde finales del mes de *Mayo 2023* hasta finales del mes *Junio 2023*.
6. **Despliegue.** Esta fase corresponde al Capítulo 6. Se llevó a cabo a principios de *Julio 2023*.

De forma paralela se ha ido documentando el proceso en esta memoria.

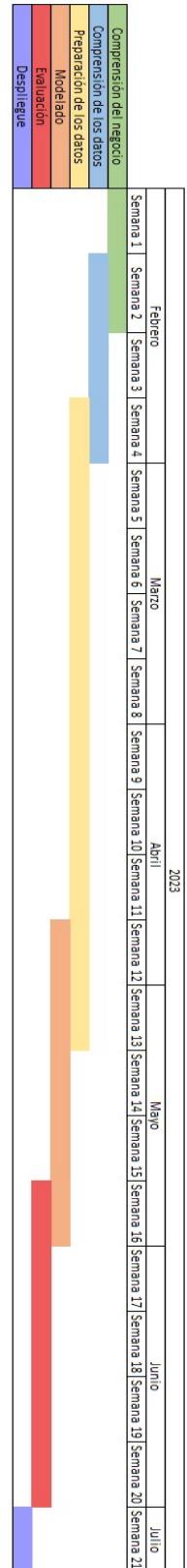


Figura A.1: Planificación del proyecto.

Capítulo B. Recursos Software

A continuación se lista el software usado para el desarrollo de este TFG. Se acompañará cada recurso con una pequeña descripción, así como de que secciones del desarrollo se han usado.

- **Flask 2.3.2.** Framework escrito en Python el cual permite crear aplicaciones web de forma rápida y sencilla.
- **Jupyter notebooks.** Aplicación que permite crear documentos que contienen código ejecutable y texto. Utilizado para en la realización del análisis exploratorio de los datos y los modelos.
- **Kaggle.** Plataforma online para desarrollar proyectos de ciencia de datos y *Machine Learning*, donde se albergan gran cantidad de competiciones, permitiéndonos ejecutar libretas Jupyter en la nube de forma gratuita.
- **Matplotlib 3.2.1.** Biblioteca de Python para la generación de gráficos.
- **Microsoft Excel.** Software para llevar a cabo hojas de cálculo desarrollado por Microsoft. Utilizada para la planificación del proyecto en el Capítulo A del Anexo.
- **Overleaf.** Editor colaborativo de LaTeX basado en la nube donde se ha realizado este documento.
- **Pandas 1.1.1.** Biblioteca de software para la manipulación y análisis de datos para Python. Utilizado para la carga y tratamiento de los datos con formato de DataFrame.
- **Pydicom 2.0.0** Paquete de Python que nos permite trabajar con las imágenes DICOM.
- **Python 3.7.** Lenguaje de programación utilizado en las libretas Jupyter y la implementación de los modelos
- **Python 3.9.6.** Lenguaje de programación utilizado para la construcción de la aplicación.

-
- **Sklearn** Biblioteca para aprendizaje automático para Python.
 - **TensorFlow 2.3.0.** Biblioteca de Redes Neuronales de Código Abierto escrita en Python. Utilizada para la creación del modelo de redes neuronales.
 - **Visual Studio Code.** Entorno de desarrollo donde hemos llevado a cabo la aplicación, y algunas otras tareas con nuestras imágenes.
 - **Windows 11.** Sistema operativo desarrollado por Microsoft. Todo el software, a excepción de Kaggle, ha sido ejecutado sobre este sistema operativo.

Capítulo C. Recursos Hardware

Los recursos hardware usados para el desarrollo de este proyecto se listan a continuación:

- **CPU.** Intel(R) Core(TM) i7-10710U CPU a 1.10GHz.
- **RAM.** 16GB a 2667MHz.
- **Tarjeta gráfica.** NVIDIA GeForce GTX 1650 with Max-Q Design.

Referencia bibliográfica

- [1] Zuhair Akhtar. Xception: Deep learning with depth-wise separable convolutions. <https://iq.opengenus.org/xception-model/>.
- [2] Frederick A Anderson, H Brownell Wheeler, Robert J Goldberg, David W Hosmer, Nilima A Patwardhan, Borko Jovanovic, Ann Forcier, and James E Dalen. A population-based perspective of the hospital incidence and case-fatality rates of deep vein thrombosis and pulmonary embolism: the Worcester DVT study. *Archives of internal medicine*, 151(5):933–938, 1991.
- [3] Jason Brownlee. Dropout regularization in deep learning models with keras. <https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/>.
- [4] Errol Colak, Felipe C Kitamura, Stephen B Hobbs, Carol C Wu, Matthew P Lungen, Luciano M Prevedello, Jayashree Kalpathy-Cramer, Robyn L Ball, George Shih, Anouk Stein, et al. The rsna pulmonary embolism ct dataset. *Radiology: Artificial Intelligence*, 3(2):e200254, 2021.
- [5] Kushal Das. Introduction to flask. <https://pymbook.readthedocs.io/en/latest/flask.html>.
- [6] Sociedad Española de Trombosis y Hemostasia. ¿sabes qué es la trombosis? <https://www.covid-19.seth.es/trombosis/>.
- [7] Computerphile Dr Mike Pound. Canny edge detector - computerphile. <https://youtu.be/sRFM5IEqR2w>, 2015.
- [8] educba. Keras maxpooling2d. <https://www.educba.com/keras-maxpooling2d/>.
- [9] Álvaro Fernández García et al. Aplicación de técnicas de aprendizaje automático para la extracción automática de la región de interés en imágenes. 2021.

-
- [10] Miguel Franqueira Varela. Algoritmo de pre-procesamiento de imágenes para la estimación de la edad. 2021.
 - [11] Ligdi González. ¿qué es el perceptrón? perceptrón simple y multicapa. <https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/>.
 - [12] Dr Paul Harper. Pulmonary embolism diagnosis – ctpa. <https://www.coagulationconversation.com/medical/having-a-ctpa/>.
 - [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
 - [14] Kürkciyan Stepan, Meron Giora, Sterz Fritz, Janata Karin, Domanovits Hans, Holzer Michael, Berzlanovich Andrea, C Bankl Hans, and N Laggner Anton. Pulmonary embolism as cause of cardiac arrest. *Archives of Internal Medicine*, 160(10):1529, 2000.
 - [15] MathWorks. ¿qué son las redes neuronales convolucionales? <https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html>.
 - [16] Kevin McLean. How to use keras.layers.flatten(). <https://kevinmclean.medium.com/how-to-use-keras-layers-flatten-c3f29ed1b686>.
 - [17] Rodrigo F Mello and Moacir Antonelli Ponti. *Machine learning: a practical approach on the statistical learning theory*. Springer, 2018.
 - [18] Massimo Miniati, Caterina Cenci, Simonetta Monti, and Daniela Poli. Clinical presentation of acute pulmonary embolism: survey of 800 cases. *PloS one*, 7(2):e30891, 2012.
 - [19] Aditya Mishra. Metrics to evaluate your machine learning algorithm. <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>.
 - [20] U.S. Department of Health and Human Services. Coágulos sanguíneos. <https://medlineplus.gov/spanish/bloodclots.html>.
 - [21] Radiological Society of North America. Classify pulmonary embolism cases in chest ct scans. <https://www.kaggle.com/competitions/rsna-str-pulmonary-embolism-detection/overview/description>.
 - [22] Radiological Society of North America. Classify pulmonary embolism cases in chest ct scans. <https://www.kaggle.com/competitions/rsna-str-pulmonary-embolism-detection/data>.
 - [23] Radiological Society of North America. Rsna pulmonary embolism detection challenge. <https://www.rsna.org/education/ai-resources-and-training/ai-image-challenge/rsna-pe-detection-challenge-2020>.
-

- [24] Pablo Palacios. Rsna-str pe detection train jpegs (256x256) for binary classification. <https://www.kaggle.com/datasets/pablopalacios2001/data-binary>.
- [25] Pablo Palacios. Rsna-str pe detection train jpegs with sobel filter (256x256) for binary classification. <https://www.kaggle.com/datasets/pablopalacios2001/data-sobel-filter>.
- [26] Ian Pan. Rsna-str pe detection train jpegs (256x256). <https://www.kaggle.com/datasets/vaillant/rsna-str-pe-detection-jpeg-256>.
- [27] Paul-Louis Pröve. Squeeze-and-excitation networks. <https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eadc7>.
- [28] Martin Riva. Batch normalization in convolutional neural networks. <https://www.baeldung.com/cs/batch-normalization-cnn>.
- [29] Adrian Rosebrock. Keras conv2d and convolutional layers. <https://pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/>.
- [30] Adrian Rosebrock. Vggnet, resnet, inception, and xception with keras. <https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>.
- [31] Cynthia Sadera, Sharon Halliburton, Ladan Panahi, and George Udeani. Introductory chapter: Pulmonary embolism. In *New Knowledge about Pulmonary Thromboembolism*. IntechOpen, 2022.
- [32] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.
- [33] Juan Miguel Sierra Ramos. Introducción a las redes neuronales artificiales. 2022.
- [34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [35] Doctoralia Internet SL. ¿qué es el formato dicom? las claves del estándar en imágenes médicas. <https://clinic-cloud.com/blog/formato-dicom-que-es-estandar-imagenes-medicas/>.
- [36] Princeton University Stanford Vision Lab, Stanford University. image-net. <https://www.image-net.org/>.
- [37] Juan Bosco Mendoza Vega. Tutorial: Xgboost en python. <https://medium.com/@jboscomendoza/tutorial-xgboost-en-python-53e48fc58f73>.

-
- [38] Vaibhav Verdhan. Supervised learning with python. *Apress, Springer, Berkeley, CA*, 2020.
 - [39] Wikipedia contributors. Sobel operator — Wikipedia, the free encyclopedia, 2023. [Online; accessed 12-June-2023].
 - [40] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1. Manchester, 2000.