Facultad de Ciencias

Uso de la Privacidad Diferencial con Datos de Imagen Médica extraídos de Imágenes en formato DICOM

(Use of Differential Privacy with Data from Medical Images extracted from Images in DICOM format)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN CIENCIA DE DATOS

Autor: Lucía Albon Aparicio

Director\es: David Rodriguez Gonzalez

Septiembre - 2023

Resumen

Este trabajo de fin de máster se enfoca en abordar la preocupación por la privacidad de los datos de imágenes médicas obtenidos de archivos DICOM. Con el propósito de preservar la privacidad del paciente mientras se comparten estos datos para diagnóstico, análisis e investigación, se explora la aplicación de técnicas de privacidad diferencial. Para ello, se lleva a cabo una revisión bibliográfica para comparar estrategias de privacidad diferencial para datos de imagen médica, evaluando su efectividad en proteger la privacidad sin sacrificar la utilidad de los datos. El proyecto incluye la extracción y preprocesamiento de datos de imágenes médicas, la implementación de técnicas de privacidad diferencial y la evaluación de su impacto en la calidad y utilidad de los datos para aplicaciones médicas. Los resultados contribuyen a la comprensión de cómo preservar la confidencialidad en un entorno de datos médicos digitales sin comprometer su valor.

Palabras clave: Imagen médica, DICOM, Desidentificación, Privacidad Diferencial

Abstract

This master's thesis focuses on addressing the privacy concerns related to medical image data obtained from DICOM files. The application of differential privacy techniques is explored with the aim of preserving patient privacy while sharing this data for diagnosis, analysis and research purposes. To accomplish this, a literature review is conducted to compare differential privacy strategies for medical image data, evaluating their effectiveness in safeguarding privacy without sacrificing data utility. The proyect encompasses the extraction and preprocessing of medical image data, the implementation of differential privacy techniques, and the assessment of their impact on the quality and usefulness of the data for medical applications. The results contribute to the understanding of how to maintain confidentiality of digital data in a medical environment without compromising its value.

Key words: Medical images, DICOM, Deidentification, Differential Privacy

Índice

1. Introducción

1.1. Motivación

Los metadatos de las imágenes médicas en formato DICOM contienen información relevante sobre el tipo, la calidad y el origen de las imágenes, así como datos personales de los pacientes y, potencialmente, del personal sanitario que haya participado en el examen, como su nombre, edad o diagnóstico. La forma más común (tradicional) de desidentificar estos datos es utilizar los métodos basados en reglas, y es esto lo que proporciona el estándar DICOM [1].

El uso de métodos alternativos englobados en el *Statistical Disclosure Control* (k-anonimity, differential privacy...) no han sido casi explorados en este ámbito, pero pueden resultar vitales si se pretenden compartir datos para investigación manteniendo la privacidad de los mismos.

La privacidad diferencial es una técnica que permite proteger la información personal de los individuos en conjuntos de datos. En este proyecto, se propone utilizar esta técnica para proteger la privacidad de los pacientes en datos de imagen médica extraídos de imágenes en formato DICOM. Con ello, se pretende preservar la utilidad de las imágenes para investigación dificultando también la posible reidentificación de los pacientes.

1.2. Objetivos

- Investigar el uso de la privacidad diferencial en el contexto de datos de imagen médica.
- Desarrollar un método para aplicar la privacidad diferencial a datos de imagen médica extraídos de imágenes en formato DICOM.
- Evaluar el rendimiento del método desarrollado en términos de precisión y privacidad.

1.3. Implementación

Se utilizan técnicas y algoritmos existentes y se adaptan al contexto específico del proyecto. Una vez desarrollado el método, se implementará utilizando el lenguaje de programación Python [2].

1.4. Alcance del trabajo

En este trabajo se tratan exclusivamente los datos de la cabecera de los ficheros DICOM, cuya estructura se explica en la sección 2.1. De estos se eliminan las secuencias, debido a que a la hora de tratar los datos, complican la estructura del dataset. También se establece el valor de los datos de los píxeles como 'NA' ya que estos han de ser tratados de manera diferente al resto de los datos.

1.5. Resultados esperados

Se espera que el método desarrollado permita proteger la privacidad de los pacientes en datos de imagen médica extraídos de imágenes en formato DICOM, sin comprometer la precisión del análisis, para facilitar el que se puedan compartir en el futuro.

2. Conceptos Teóricos

2.1. Imágenes médicas (DICOM)

El estándar de Comunicaciones y Digitalización de Imágenes en Medicina (**DICOM**, por sus siglas en inglés) es el estándar internacional para las imágenes médicas y la información relativa a ellas. Especifica un protocolo de intercambio de datos no propietario, el formato de la imagen digital y la estructura de las imágenes biomédicas, así como la información específica de las mismas [3].

2.1.1. Formato de los datos

El formato DICOM agrupa la información en datasets, y se compone de una cabecera y datasets de imagen guardados en un único fichero, como se muestra en la Figura 2.

Los primeros paquetes de información constituyen la **cabecera**. Contienen información demográfica sobre el paciente y toda la información necesaria para que el ordenador pueda interpretar correctamente la imagen (parámetros de adquisición, dimensiones, colores...). Cada línea de esta contiene cuatro elementos [4]:

- **Las etiquetas** son únicas para cada atributo y se componen de dos números de 4 dígitos separados por una coma.
- **El nombre del atributo**
- **Los valores de representación (VR)**, que se pueden ver recogidos en una tabla en el anexo A, identifican el tipo de valor que se corresponde a cada atributo (número entero, fecha, nombres personales...).
- **El valor** correspondiente a cada atributo. Este puede ser un único elemento o una lista de elementos.

En la imagen que se muestra a continuación (Figura 1) se puede ver un ejemplo de como se visualizarían los datos DICOM con todos los elementos descritos presentes.

```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 214
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: Digital Mammography X-Ray Image Storage - For Presentation
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.840.113681.2216075982.713.3414352039.554.1
(0002, 0010) Transfer Syntax UID                 UI: Explicit VR Little Endian
(0002, 0012) Implementation Class UID            UI: 1.2.410.200003.2020819.5.2.1
(0002, 0013) Implementation Version Name         SH: 'MAROTECH REGISTER 5.0'
(0002, 0016) Source Application Entity Title     AE: 'MAROTECH'
-------------------------------------------------
(0008, 0005) Specific Character Set              CS: 'ISO_IR 100'
(0008, 0008) Image Type                          CS: ['DERIVED', 'SECONDARY']
(0008, 0016) SOP Class UID                       UI: Digital Mammography X-Ray Image Storage - For Presentation
(0008, 0018) SOP Instance UID                    UI: ANONYMOUS
```

Figura 1: Ejemplo real de los datos un fichero en formato DICOM [5].

La información de la cabecera está ligada al fichero DICOM para que no se separe de los datos de la imagen. Si estos se separasen, el ordenador no sabría que estudio se ha realizado ni a quién pertenece, y por lo tanto, no podría mostrar la imagen correctamente [4].

Después de la cabecera está un único atributo con todos los **datos de intensidad de los píxeles** de la imagen. Estos datos consisten en una serie de ceros y unos que, junto con la información de la cabecera, permiten la reconstrucción de la imagen [4].

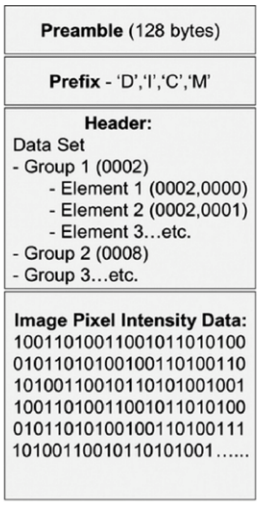

Figura 2: Estructura de un fichero de imagen en formato DICOM [6].

2.1.2. Protocolo no propietario

En el contexto de DICOM, el término "no propietario"se refiere a que el protocolo de intercambio de datos definido por DICOM no está bajo el control exclusivo de una empresa o fabricante específico. Esto quiere decir que DICOM es un estándar abierto y público que puede ser implementado por cualquier entidad sin restricciones o barreras de propiedad intelectual. Al ser no propietario, permite la interoperabilidad y el intercambio de información entre diferentes sistemas y fabricantes, facilitando así la comunicación y el intercambio de imágenes médicas de manera efectiva en el campo de la medicina.

DICOM es una marca registrada de la Asociación Nacional de Fabricantes de Equipos Electrónicos (NEMA, por sus siglas en inglés) para sus publicaciones de las normas relacionadas con la comunicación digital de información médica [1].

2.1.3. Implementación

DICOM se implementa en casi todos los dispositivos de radiología, imágenes cardiológicas y radioterapia (rayos X, tomografía computarizada, resonancia magnética, ultrasonido, etc.), y cada vez más en dispositivos de otros campos médicos como la oftalmología y la odontología. Con cientos de miles de dispositivos de imágenes médicas en uso, DICOM es uno de los estándares de mensajería sanitaria más ampliamente implementados en el mundo, y actualmente hay miles de millones de imágenes DICOM en uso para la atención clínica.

Desde su primera publicación en 1993, DICOM ha revolucionado la práctica de la radiología, permitiendo reemplazar las placas de rayos X con un flujo de trabajo completamente

digital. DICOM ha permitido aplicaciones avanzadas de imágenes médicas que han cambiado el funcionamiento de la medicina clínica. Es el estándar que hace que las imágenes médicas funcionen tanto para los médicos como para los pacientes [1].

2.2. Desidentificación

Para poder entender el término desidentificación, primero hay que definir una serie de conceptos:

La **anonimización**, en el ámbito médico, se define como la eliminación irreversible del vínculo que existe entre el individuo y sus datos médicos, haciendo que resulte prácticamente imposible restablecer dicho vínculo [7].

La **pseudonimización** es el proceso reversible que consiste en sustituir campos de datos identificables, como por ejemplo el nombre, fecha de nacimiento o dirección, por un conjunto de datos con seudónimos [8].

La **desidentificación** es un término utilizado en el ámbito de la privacidad y protección de datos. Hace referencia al proceso de eliminar o suprimir cualquier información que pueda identificar directa o indirectamente a una persona en un conjunto de datos [9].

2.2.1. Qué se debe desidentificar

En el ámbito de la imagen médica, algunos de los elementos que pueden considerarse un riesgo a la privacidad y confidencialidad son:

- Metadatos
  - Estructurados
    - texto
    - numérico
    - binario
  - No estructurados
    - texto
    - binario
- Datos de píxeles
  - Contenido
  - Características
    - rostros
    - genitales internos y externos
    - Características estructurales distintivas o inusuales

        - ◇ directas (iris, retina, huellas dactilares)
        - ◇ indirectas (estructura del cráneo, pelvis, defectos de nacimiento)
      - Modificaciones (tatuajes)
      - Objetos físicos (dispositivos médicos implantados, dentaduras, joyería)
  - Metadatos integrados en la imagen
      - Texto no estructurado
        - ◇ texto genérico
        - ◇ cursiva, notas escritas a mano
        - ◇ logos
      - Escondidos (marcas de agua)

2.2.2. Categorías de técnicas de desidentificación

Los métodos de desidentificación se pueden categorizar en dos grupos:

- **Basados en reglas** (lista de elementos a proteger o eliminar)
- **Basados en estadística** (analiza el riesgo de identificación de un individuo)

2.2.2.1 Métodos basados en reglas

Los métodos basados en reglas son los más utilizados cuando se trata de la desidentificación de imágenes médicas. Las reglas pueden ser listas de elementos de datos que se han de eliminar, conservar o modificar de alguna manera, o pueden ser más complejas y especificar un comportamiento basado en más de un elemento de datos al mismo tiempo.

Dado un conjunto de datos estructurados, se pueden dar dos formas de tratarlos: generar una lista de todos los elementos que se deben mantener (*whitelist*) o generar una lista de aquellos elementos que se deben eliminar o modificar ya que no son seguros (*blacklist*). Con cualquiera de estas dos aproximaciones es importante tener en cuenta los elementos cuyo estado es desconocido, ya que pueden ser potencialmente inseguros, o pueden contener información crítica que puede ser útil a la hora de realizar análisis [9].

Un enfoque evidente para preparar una lista de elementos de datos DICOM que presenten un riesgo potencial es revisar manualmente el estándar de Diccionario de Datos o Módulos de elementos de datos descritos como:

- **Identificadores directos** (nombre e ID del paciente, números de teléfono...).
- **Identificadores indirectos** (edad y sexo del paciente, código postal...).

Definir correctamente cuáles de los atributos corresponden a cada una de estas categorías es una tarea complicada de llevar a cabo. Esto se debe a las posibles especificaciones que pueda tener el estudio que se lleva a cabo, especialmente teniendo en cuenta la posible inclusión

de nuevos atributos en el estándar DICOM. Es por ello que se recomienda utilizar una lista o conjunto de reglas existentes de una fuente autoritativa, como es el caso del estándar DICOM [10].

2.2.2.2 Métodos estadísticos

Cuando se trata de microdatos, se utilizan los métodos estadísticos. Los microdatos son registros que contienen información de individuos. Los metadatos derivados de imágenes médicas se consideran microdatos cuando dichas imágenes se atribuyen a personas de forma individual y contienen información que define características concretas de estos [9].

El método estadístico, se debería utilizar en conjunto con el método basado en reglas. Esto se debe a que el método estadístico solo se puede aplicar una vez que se hayan empleado las reglas necesarias para eliminar todos los identificadores directos e indirectos, así como los atributos sensibles que aparecen en los metadatos, los datos de los píxeles y todo dato que acompañe [9].

2.3. Privacidad Diferencial

La **Privacidad Diferencial** (DP, por sus siglas en inglés) es una técnica de privacidad que se utiliza para proteger información sensible, preservando su utilidad y la extracción de información útil. Establece que un algoritmo es diferencialmente privado si, viendo su resultado, un observador no puede saber si los datos de un individuo concreto están incluidos en la base de datos usada para llegar a dicho resultado. Esto se logra agregando ruido aleatorio a los datos, lo que aumenta la dificultad de la identificación de registros individuales [11].

Los objetivos de la privacidad diferencial son [11]:

- **Utilidad**: permitir el "análisis estadístico" de los conjuntos de datos. Es decir, permitir inferir conocimiento sobre los datos, aplicar modelos de ML...
- **Privacidad**: proteger los datos a nivel individual, especialmente ante ataques basados en el uso de información auxiliar.

2.3.1. Privacidad Diferencial explicada

La privacidad diferencial, más que una herramienta, se trata de un criterio compuesto por múltiples herramientas para satisfacer el análisis de información personal sensible. Proporciona una probabilidad matemática de la garantía de la protección de la privacidad frente a numerosos posibles ataques [12].

Elementos a tener en cuenta:

- La garantía de privacidad diferencial
- Parámetro de pérdida de privacidad (ϵ)
- Interpretación de la garantía

Una computación protege la privacidad de los individuos en los datos si su resultado no revela ninguna información específica de ningún sujeto de datos individuales. La privacidad diferencial formaliza esta idea como una definición matemática.

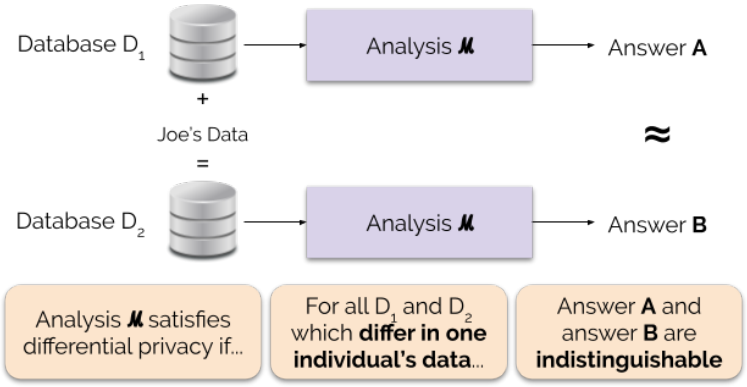

Figura 3: Esquema ejemplificando el funcionamiento de la privacidad diferencial [13].

El **parámetro de pérdida de privacidad** determina cuánto ruido se debe añadir al cómputo, y se representa mediante la letra griega ϵ. Al cambiar el valor de este parámetro, se determina el nivel de privacidad de los datos y, por consecuencia, el nivel de exactitud de los resultados obtenidos al aplicar un algoritmo a estos datos con ruido [12].

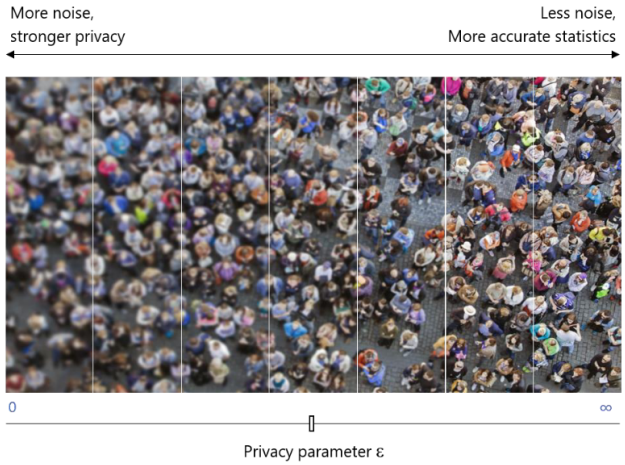

Figura 4: Introducir una cantidad elevada de ruido aumenta el nivel de privacidad, pero hace que disminuya su utilidad debido a que es más difícil obtener resultados estadísticos fiables (lado izquierdo de la imagen). El parámetro de perdida de privacidad (ϵ) es el que se modifica para ajustar la cantidad de ruido que se añade. Hay una relación inversa entre ϵ y privacidad por lo que a menor valor de ϵ mayor nivel de privacidad, y viceversa [14]

2.3.2. Posibles aplicaciones de Privacidad Diferencial en DICOM

Estos son algunos de los casos en los que se podría aplicar la privacidad diferencial en datos que se encuentran en formato DICOM:

- **Investigación médica, colaboración y difusión de datos:** A menudo los investigadores necesitan acceder a grandes y diversos conjuntos de datos para sus estudios médicos. Al aplicar la privacidad diferencial a los archivos DICOM, las instituciones pueden compartir datos desidentificados que preservan las características generales del conjunto de datos, a la vez que evitan la identificación de pacientes individuales [15].
- **Estudios de Salud de la Población:** La privacidad diferencial puede permitir el análisis a nivel de población a partid de datos DICOM sin revelar información sensible sobre pacientes individuales. Esto es muy valioso para las iniciativas de salud públicas y políticas [15].
- **Segunda Opinión Remota:** Al pedir una segunda opinión de un especialista de otro centro o institución, los pacientes pueden necesitar compartir sus imágenes médicas para que puedan ser valoradas. Aplicando privacidad diferencial puede ayudar a proteger la identidad del paciente a la vez que proporcionar la información necesaria al especialista para que pueda ofrecer su opinión [15].

3. Metodología y desarrollo

La metodología propuesta consta de tres partes:

- **Revisión bibliográfica**: En esta etapa se realizará una revisión exhaustiva de la literatura sobre el uso de privacidad diferencial en el contexto de datos de imagen médica. Se buscarán estudios y trabajos previos que hayan aplicado esta técnica en conjuntos de datos similares, y se analizarán sus resultados y metodologías.
- **Desarrollo e implementación del método**: A continuación, se desarrollará un método para aplicar la privacidad diferencial a datos de imagen médica extraídos de imágenes en formato DICOM. Se utilizarán técnicas y algoritmos existentes y se adaptarán al contexto específico del proyecto. Una vez desarrollado el método, este se implementará utilizando Python como lenguaje de programación.
- **Evaluación del método**: Finalmente, se evaluará el rendimiento del método desarrollado utilizando conjuntos de datos reales. Se medirá la precisión del análisis y se evaluará el nivel de privacidad alcanzado. Los resultados obtenidos se compararán con los obtenidos por otros métodos existentes.

3.1. Revisión bibliográfica

En esta sección se realiza una revisión bibliográfica de la literatura existente sobre el uso de privacidad diferencial en el contexto de imagen médica.

3.1.1. Qué es la desidentificación de datos y por qué es importante

La desidentificación de datos se considera un enmascaramiento dinámico con el fin de romper el vínculo entre los datos y el individuo al que pertenecen inicialmente. Al eliminar o modificar datos que son identificadores directos mediante técnicas de desidentificación se permite compartir y reutilizar los datos de forma más sencilla y segura. HIPAA (*Health Insurance Portability and Accountability Act*) determina dos técnicas para la desidentificación de los datos: Salvaguarda y Determinación por expertos [16].

El método de salvaguarda determina la eliminación de 18 tipos de identificadores para evitar que un individuo pueda ser identificado. Este método, conocido por su simplicidad y bajo coste, puede ser demasiado restrictivo o demasiado permisivo. Esto quiere decir que por un lado puede quitarle utilidad a los datos, pero por otro puede hacer disponibles demasiados identificadores indirectos [16].

El método de determinación por expertos consiste en aplicar estadística y principios científicos a los datos para lograr un bajo riesgo de reidentificación. Este método permite adaptar el proceso de desidentificación a las circunstancias en las que se quiere aplicar, haciendo que esté método sea muy flexible y garantize la mayor utilidad posible de los datos [16].

La desidentificación de los datos es especialmente importante en el campo de la medicina y se lleva a cabo en dos procesos:

1. El primer paso se corresponde con la clasificación de identificadores directos e indirectos. Los identificadores que son únicos para un solo individuo (número de seguridad social, dni, número de teléfono...) se consideran identificadores directos. El resto son identificadores indirectos (altura, peso, etnicidad...) [16].
2. Una vez clasificados los identificadores se pueden aplicar técnicas de enmascarado de los datos para desidentificarles. La técnica más común es la pseudonimización, pero esta no se puede aplicar sola ya que sólo se aplica a identificadores directos [16].

Hay dos métodos de desidentificación principales: generalización y aleatorización.

La generalización (k-anonimato) es una técnica que se implementa una vez se hayan enmascarado todos los identificadores directos. Esta reduce el riesgo de reidentificación ocultando a los individuos en grupos y suprimiendo identificadores indirectos para grupos más pequeños que un número predeterminado, k. Previene la necesidad de suprimir datos, aumentando así la utilidad de los mismos [16].

La aleatorización (privacidad diferencial) es una técnica que, al igual que la de generalización, se aplica después de que hayan sido enmascarados los identificadores directos. Hay dos formas en las que se puede aplicar la privacidad diferencial: de forma local o de forma global [16].

- La privacidad diferencial local es un método de aleatorización de datos que se aplica generalmente a atributos sensibles y ofrece una garantía matemática contra los ataques de inferencia basados en atributos, al limitar la información personal que un atacante puede deducir mientras preserva cierta utilidad analítica y permite a las personas negar atributos específicos en sus registros [16].
- La privacidad diferencial global es un método que aleatoriza datos agregados, garantizando matemáticamente la protección contra ataques de inferencia basados en identidad, atributos, participación y relaciones, permitiendo a las personas negar su participación en el conjunto de datos consultado en su totalidad [16].

3.1.2. Control de Divulgación Estadística (SDC)

Statistical Disclosure Control (SDC), es decir, el Control de Divulgación Estadística, es una técnica de reducción del riesgo de reidentificación de los registros que contienen información sobre datos de personas. En el caso concreto de este trabajo, se trata de imágenes médicas y los metadatos que las acompañan. Esta técnica solo es aplicable sobre **datos estructurados**, ya que si estos no se encuentran en dicho formato, es necesario aplicar técnicas de extracción de la información para identificar y separarles [17].

La divulgación de la identidad ocurre cuando se encuentra una relación entre los registros de un individuo y la identidad del mismo mediante el uso de identificadores indirectos, debido a que los identificadores directos han de ser previamente eliminados o reemplazados [17].

Los **principios** en los que está basada esta técnica son los siguientes:

- El riesgo y probabilidad de reidentificación se puede estimar.
- Los datos pueden ser modificados para reducir ese riesgo a un nivel aceptable, conservando gran parte de su utilidad.

3.1.3. Privacidad Diferencial para el Análisis de Datos Médicos

Cuando se trata de datos médicos, la privacidad diferencial se aplica principalmente para la publicación y minería de datos. Eso puede prevenir, en la fase de publicación, el filtrado de datos privados que se podrían obtener al hacer un análisis con conocimientos previos del contenido. En la fase de minería de datos, puede resistir los ataques de filtrado causados por el adversario utilizando MIA (*Membership Interference Attack*) sobre el modelo [15].

En la imagen 5 se muestra un esquema con las principales aplicaciones de privacidad diferencial en diferentes ámbitos médicos. Al estar este trabajo basado en datos de imagen médica se mirará únicamente este enfoque.

En este artículo se revisan varias propuestas:

- Ziller et al. proponen un marco de software de código abierto basado en la aplicación del algoritmo DP-SGD para abordar la clasificación de imágenes médicas y las tareas de segmentación semántica de aprendizaje profundo [18].
- Yuan et al. explotaron el aprendizaje profundo colaborativo con el mecanismo de ruido Gaussiano para llevar a cabo experimentos en el conjunto de datos de imágenes de rayos-x (neumonía), y encontraron que la pérdida de precisión fue mínima, teniendo un impacto limitado en los resultados [19].
- Adnan et al. señalaron que el aprendizaje federado con privacidad diferencial ha demostrado ser el marco viable y confiable para el análisis de imágenes médicas mediante colaboración de aprendizaje automático [20].

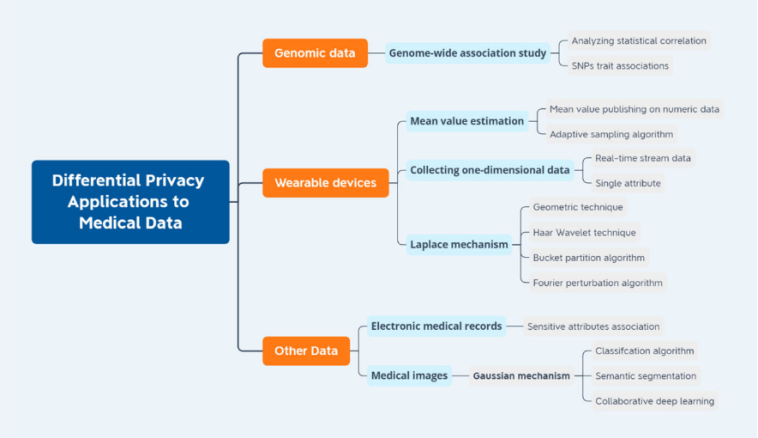

Figura 5: Aplicación de privacidad diferencial sobre datos médicos: aplicaciones actuales en investigación de datos sobre el genoma, aparatos médicos portátiles, registros digitales e imagen médica, etc. [21].

3.1.4. Técnicas y algoritmos de privacidad diferencial en datos DICOM

La implementación de la privacidad diferencial en datos DICOM conlleva unas complicaciones con respecto a otro tipo de datos, debido a la naturaleza específica de las imágenes médicas y su información asociada [17]. Algunas de las técnicas que se han explorado para aplicar privacidad diferencial a imágenes DICOM son las siguientes:

- **Mecanismo Laplaciano**
- **Mecanismo Exponencial**
- **Respuestas aleatorizadas**

Como se ha mencionado anteriormente, hay que tener cuidado al aplicar cualquier método de desidentificación, especialmente aplicando privacidad diferencial, ya que hay que garantizar que los datos sean suficientemente útiles para su análisis, asegurando a su vez la protección de la privacidad de los pacientes [17].

La elección de la técnica adecuada dependerá del contexto concreto y los requisitos de privacidad y utilidad de los datos. Además, la implementación de cualquier técnica de privacidad diferencial debe cumplir las leyes y regulaciones de protección de datos médicos que se apliquen a cada caso [17].

3.1.4.1 Respuestas Aleatorizadas

Para entender este concepto, lo primero que hay que saber es que la privacidad diferencial proporciona privacidad mediante la introducción de aleatoriedad. Una de las primeras técnicas implementadas fue el proceso de **respuestas aleatorizadas**, una técnica diseñada para recolectar información estadística sobre comportamientos ilegales o vergonzosos [22].

En la Figura 6 se muestra el método que se implemento cuando se hizo esa recolección de datos estadísticos. Como se puede ver, las respuestas no son incriminatorias ya que un simple “Sí” se da con una probabilidad de $\frac{1}{4}$, independientemente si esa es la respuesta verdadera o no.

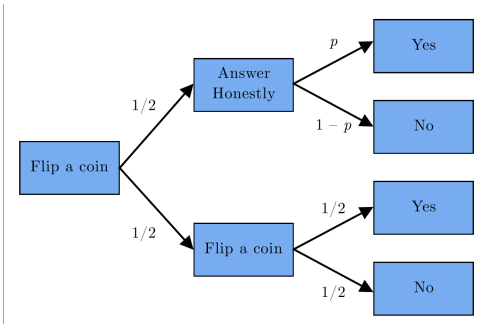

Figura 6: El método de respuestas aleatorizadas se realizó por primera vez usando de una moneda. Si al lanzar una moneda cae en cruz, el participante deberá contestar honestamente. Si, por lo contrario, cae en cara, habrá que tirar otra moneda y contestar "Sí"si cae en cara y "No"si cae en cruz [22].

La privacidad se garantiza mediante la negación plausible de cualquier resultado. Esto quiere decir que la manera en la que se obtienen las respuestas influye en cómo pueden ser interpretadas de manera válida [22].

3.1.4.2 Mecanismo de Laplace

El **mecanismo de Laplace** consiste en añadir ruido al resultado de una consulta a partir de la distribución de Laplace. De acuerdo con este mecanismo, para una función $f(x)$ que devuelve como resultado un número, la siguiente definición satisface ϵ- privacidad diferencial [23]:

$$F(x) = f(x) + Lap\left(\frac{s}{\epsilon}\right)$$

Figura 7: Definición del mecanismo de Laplace, donde s es la sensibilidad de f, y Lap(S) es una muestra de la distribución Laplace con centro en 0 y escala S [23].

Cuando hablamos de este mecanismo, es importante conocer el concepto de "*sensitivity*" (sensibilidad), y más concretamente, la **sensibilidad** l_1 de una función. Este parámetro define la magnitud por la cual los datos de un único individuo pueden modificar la función en el peor de los casos y, por lo tanto, la incertidumbre en la respuesta que se debe introducir para poder ocultar la participación de ese individuo concreto. Dicho de otra manera, la sensibilidad de una función es el límite superior que define cuánto se deben alterar las respuestas para mantener la privacidad [22].

Este mecanismo se utiliza principalmente con resultados numéricos y funciona añadiendo ruido a la propia respuesta.

3.1.4.3 Mecanismo Exponencial

Si queremos obtener una respuesta más exacta manteniendo la privacidad diferencial, una de las soluciones es utilizar el mecanismo exponencial. Este mecanismo está diseñado para aquellas situaciones en las que se quiere elegir la mejor respuesta, pero añadir ruido al propio resultado puede afectar de manera negativa su valor. Se define qué elemento es el mejor mediante la especificación de una **función de valoración que genera una puntuación para cada elemento**, y también genera un listado de elementos de los cuales elegir. Para satisfacer la privacidad diferencial a veces, este mecanismo, elige un elemento que no tiene la puntuación más alta [23].

3.1.5. Bases de datos SQLite

SQLite es una librería de uso interno que se compone de un motor de base de datos SQL autocontenido, lo cual implica que funciona sin la necesidad de servidores. El código de SQLite está en dominio público, por lo que su uso es gratuito, independientemente del uso que se le vaya a dar. Es la base de datos más ampliamente desplegada en el mundo [24].

Existen muchas ventajas asociadas al uso de SQLite para el almacenamiento de datos. Guardar datos de ficheros DICOM en este tipo de bases de datos para la posterior aplicación de técnicas de desidentificación a los mismos puede tener una serie de ventajas que se detallan a continuación [25]:

- **Centralización de los datos**: Almacenar los datos en una base de datos centralizada facilita su gestión y el acceso a los mismos. Esto puede ser útil para aplicaciones médicas y de investigación que requieren un acceso rápido y eficiente a la información.
- **Control de acceso y seguridad**: Una base de datos centralizada permite implantar un control de acceso más efectivo. Se puede definir quién tiene acceso a los datos y qué nivel de permisos tiene. Esto es esencial para garantiza que solo las personas autorizadas puedan acceder a la información médica sensible.
- **Escalabilidad y eficiencia**: Las bases de datos están diseñadas para manejar grandes volúmenes de datos de manera eficiente. Las consultas y el análisis de los datos también pueden realizarse de manera eficiente en una base de datos centralizada en comparación con archivos DICOM individuales.
- **Integración y análisis**: Al tener los datos en una base de datos, es más fácil integrarlos con otras fuentes de datos médicos y realizar análisis avanzados, como la minería de datos o el aprendizaje automático, para obtener información valiosa sobre diagnósticos, tratamientos y resultados.
- **Mantenimiento**: Administrar y mantener una base de datos de este tipo es más sencillo que gestionar un gran volumen de archivos DICOM individuales. Además, es más fácil realizar copias de seguridad y restauraciones de datos.

Además de todas estas ventajas que pueden tener un carácter más general, hay otras que pueden ser más específicas al contexto de este trabajo, que son:

- **Privacidad Diferencial**: La privacidad diferencial es una técnica que permite compartir datos mientras se protege la privacidad de los individuos. Al almacenar los datos en una base de datos centralizada, se pueden aplicar mecanismos de privacidad diferencial para limitar la divulgación de información sensible mientras se permite el acceso a datos útiles para análisis y estudios médicos.
- **k-anonimato**: El k-anonimato es una técnica de anonimización que busca garantizar que cada entrada en la base de datos sea indistinguible de al menos k-1 otras entradas. Al aplicar k-anonimato a una base de datos centralizada de datos DICOM, se puede proteger la privacidad de los pacientes reduciendo la capacidad de identificar a individuos en función de sus datos médicos.

3.2. Desarrollo e Implementación del Método

3.2.1. Obtención de los Datos

Los ficheros en formato DICOM que se utilizan en este trabajo se han obtenido del Archivo de Imágenes de Cáncer (TCIA) [26].

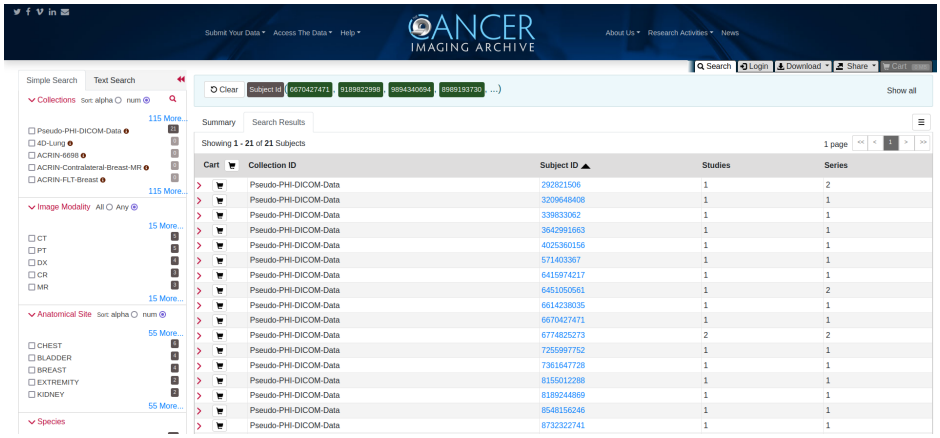

Figura 8: Archivo de Imágenes de Cáncer (Cancer Imaging Archive) donde se puede acceder a diferentes tipos de imágenes de cáncer y se pueden descargar en formato DICOM. [26]

La Iniciativa de Desidentificación de Imágenes Médicas (MIDI), patrocinada por el Instituto Nacional del Cáncer, generó un conjunto de datos de evaluación de algoritmos de desidentificación significativamente grande. Las imágenes DICOM fueron seleccionadas del TCIA, y añadieron información de salud protegida sintética para simular exámenes clínicos. Tras la creación del conjunto completo, extrajeron 21 registros para crear el conjunto de datos de evaluación publicable, que está disponible en el TCIA en la colección denominada ***Pseudo-PHI-DICOM-Data*** (tercer registro de la imagen 9).

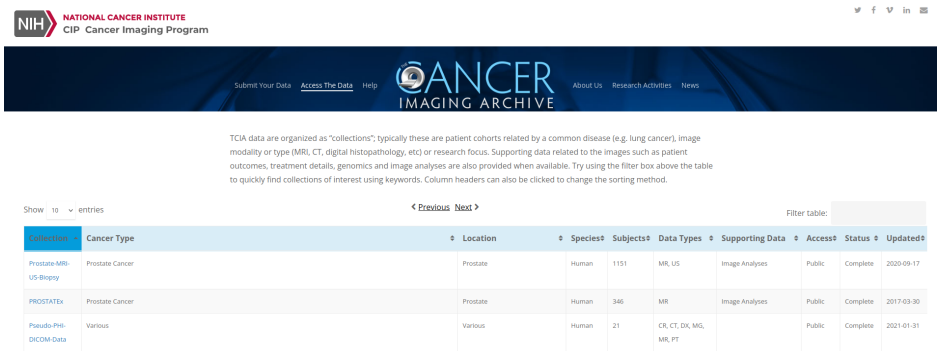

Figura 9: Captura de imagen sacada del portal TCIA donde se encuentra el conjunto de datos de imágenes DICOM utilizados en este trabajo [26] [27].

DATA SET DESCRIPTION						
Modality	**Patients**	**Studies**	**Series**	**Images**	**Anatomy (# Studies)**	**Manufacturer (# Studies)**
CT	5	5	5	268	BLADDER (4) CHEST (1)	GE MEDICAL SYSTEMS (2) PHILIPS (1) SIEMENS (1) TOSHIBA (1)
MR	3	3	5	150	KIDNEY (2) PELVIS (1)	GE MEDICAL SYSTEMS (1) SIEMENS (2)
PT	5	5	6	1,203	[BLANK] (1) BREAST (2) EXTREMITY (2)	GE MEDICAL SYSTEMS (4) SIEMENS (1)
DX	4	4	4	10	CHEST (4)	GE MEDICAL SYSTEMS (1) PHILIPS (3)
CR	3	3	4	4	CHEST (2) UTERUS (1)	FUJIFILM (3)
MG	2	2	2	58	BREAST (2)	LORAD (1) VICTRE (1)
Total	21	22	26	1,693	22	22

Figura 10: Información sobre los datos que se encuentran en el dataset de imágenes DICOM, sacada del artículo de Rutherford et al., creadores de la colección de datos [27].

3.2.2. Código

Todo el código, en lenguaje Python, se encuentra en el Anexo B. Inicialmente se generó el código para un único fichero, y una vez conseguido el resultado deseado se modificó para poder ser aplicado sobre varios ficheros a la vez.

El código desarrollado para este trabajo se divide en las siguientes partes:

1. Búsqueda de todos los directorios de ficheros DICOM dado un directorio origen.
2. Lectura de los ficheros DICOM.
3. Identificación y eliminación de secuencias.
4. Anonimización de datos sensibles.
5. Lista de los nombres de atributos únicos.
6. Creación de una base de datos SQLite.
7. Inserción de los datos en la base de datos.
8. Aplicación de privacidad diferencial.

Cada una de estas partes consta de una función que define los argumentos de entrada y el proceso que se quiere llevar a cabo. Esto se hizo así con el fin de permitir que el código sea lo más replicable posible.

A continuación, se explican estas partes en más detalle.

3.2.2.1 Librerías de Python utilizadas

Lo primero que se hizo es importar las librerías de Python necesarias para poder desarrollar el código, que se describen a continuación:

- **pydicom**: Facilita leer ficheros DICOM y transformarlos en estructuras de Python para facilitar su manipulación [28].
- **NumPy**: Paquete fundamental para realizar computación científica en Python [29].
- **pandas**: Biblioteca de código abierto que ofrece estructuras de datos de alto rendimiento y herramientas de análisis de datos fáciles de usar para el lenguaje de programación Python [30].
- **os**: Proporciona funciones para interactuar con el sistema operativo, permitiendo tareas como navegación de directorios o la manipulación de rutas de archivos entre otras [31].
- **random**: Genera números pseudoaleatorios para varias distribuciones [32].
- **sqlite3**: Permite interactuar con bases de datos SQlite, un sistema de gestión de bases de datos ligero. Permite la creación, manipulación y consulta de bases de datos SQLite directamente desde programas Python [33].

3.2.2.2 PARTE 1: Búsqueda de los directorios de los ficheros DICOM

El objetivo de esta parte del código, que se puede ver en el anexo B.1, es encontrar todos los archivos DICOM en un directorio específico y almacenar la ruta de cada uno de ellos en una lista. Para lograr esto se crea una función llamada ***get_dicom_files*** que toma como argumento el directorio origen donde se debe comenzar la búsqueda de los archivos DICOM.

Se utiliza un bucle *for* junto con *os.walk()*, de la *librería os* mencionada en la sección 3.2.2.1, para recorrer el directorio y sus subdirectorios de manera recursiva. **os.walk()** es una función que genera una tupla (root, dirs, files) para cada directorio que visita, donde *root* es la ruta al directorio actual, *dirs* es una lista de nombres de subdirectorios y *files* es una lista de nombres de archivos en el directorio actual. Si el archivo encontrado tiene la terminación .dcm que le identifica como un archivo DICOM, se añade a una lista que se inicializa fuera del bucle.

3.2.2.3 PARTE 2: Lectura de los ficheros

Esta segunda parte del código, que se encuentra en el anexo B.2, se encarga de leer los archivos DICOM cuyos nombres se encuentran en la lista generada en la parte anterior y guardar los datos en una lista, donde cada elemento constituye los datos de cada archivo.

Se crea una función llamada ***read_dicom_files*** que toma como argumento el nombre de la lista que contiene los directorios de los archivos DICOM. Se utiliza un bucle *for* que itera por cada uno de los elementos de la lista y, utilizando **pydicom.dcmread()**, se leen y se

insertan los datos en una lista nueva, donde cada conjunto de datos de un archivo constituye un elemento de la misma.

3.2.2.4 PARTE 3: Identificación y borrado de secuencias

En el anexo B.3 se encuentra la función ***process_dicom_data*** que se encarga de identificar y eliminar las secuencias que aparecen en los datasets DICOM que se encuentran en la lista generada con la función de la sección 3.2.2.3. Esta función toma una lista de conjuntos de datos DICOM como argumento y, opcionalmente, una lista de etiquetas de secuencias para eliminar.

Dentro de la función, se itera a través de cada conjunto de datos DICOM. Para cada uno de estos, se crea uno nuevo y se copian todos los atributos excepto las secuencias especificadas. Además de esto, se recorren todos los elementos (atributos) en el conjunto de datos DICOM actual utilizando un bucle. Aquí es donde se determina si un atributo es una secuencia o no. La identificación de una secuencia se basa en el atributo conocido como "Representación de Valor" (VR). Si el VR de un elemento es igual a "SQ", significa que es una secuencia, y por lo tanto el atributo debe ser eliminado. Cuando se identifica un elemento como una secuencia, su etiqueta (identificador de elemento) se agrega al conjunto que se inicializa al comienzo de la función, lo que permite realizar un seguimiento de todas las secuencias presentes en los datos DICOM originales. Sin embargo, si el elemento no se identifica como una secuencia y su etiqueta no se encuentra en la lista proporcionada, se agrega ese elemento al conjunto de datos nuevo. Esto garantiza que solo las secuencias especificadas se eliminen de los conjuntos de datos DICOM procesados.

El código configura el orden de bytes y la codificación de VR del conjunto de datos nuevo y guarda el resultado en un nuevo archivo DICOM.

Finalmente, devuelve una lista de los nuevos conjuntos de datos DICOM procesados junto con una lista de etiquetas únicas de secuencia.

3.2.2.5 PARTE 4: Anonimización de atributos sensibles

Esta parte del código, mostrado en el anexo B.4, se encarga de anonimizar datos sensibles de los archivos DICOM previamente modificados.

Al igual que en las secciones anteriores, se genera una función que toma como entrada la lista de datos DICOM modificados. Dentro de cada conjunto de datos se realizan las siguientes acciones:

- **Eliminación de Etiquetas Privadas**: Si el elemento de los datos tiene etiquetas privadas, es decir, no estándar, se eliminan del conjunto de datos.
- **Eliminación de atributos con valores vacíos**: Si un atributo contiene un valor nulo, estos se eliminan ya que no proporcionan ninguna información al conjunto.
- **Anonimización del Identificador del Paciente (PatientID)**: Si el elemento de datos tiene la etiqueta (0x0010, 0x0020) correspondiente al Identificador del Paciente, su valor se cambia a "id" para anonimizarlo.

- **Anonimización de los Datos de Píxeles (PixelData)**: Si el elemento de datos tiene la etiqueta (0x7FE0, 0x0010) correspondiente a los Datos de Píxeles, su valor se establece en "NA" para ocultar los datos de píxeles. Esto se hace porque estos han de ser tratados de manera diferente al resto de los datos, lo cual va más allá del alcance de este proyecto.
- **Anonimización de otros datos sensibles**: La ocupación, el nombre de la institución, los números de teléfono, las direcciones y los nombres constituyen datos privados identificables. Debido a esto, se establecen como "anonymous" para ocultar la información y evitar la identificación directa del individuo, ya sea paciente o personal médico participante.
- **Generalización de Fechas**: Si el elemento de datos representa una fecha (VR = DA), se mantiene solo el año y se establece el día y el mes en "01-01" para anonimizar la fecha completa. Esto se hace para generalizar la fecha manteniendo el año real.

La función devuelve la lista de datos DICOM modificados y anonimizados.

3.2.2.6 PARTE 5: Lista de los nombres de atributos únicos

Esta sección del código, que se encuentra en el anexo B.5, define una función llamada ***extract_unique_attribute_names_with_vr*** que opera sobre una lista de conjuntos de datos DICOM. Su objetivo es extraer nombres únicos de atributos DICOM junto con sus Representaciones de Valor (VR).

Primero, se crea un conjunto para almacenar estas combinaciones únicas. A continuación, la función itera a través de cada conjunto de datos en la lista y examina cada elemento (atributo) dentro de ellos.

Para cada elemento, verifica su VR. Si el VR es "DA" (Fecha), "IS" (Número entero) o "SL" (Número largo con signo), se conserva el VR original; de lo contrario, se asigna un VR en blanco. Después, la función agrega el nombre del atributo y su VR (ya sea el VR original o en blanco) como una tupla al conjunto.

Finalmente, se convierte el conjunto en una lista y se devuelve, proporcionando una lista de combinaciones únicas de nombres de atributos y VR extraídas de los conjuntos de datos. Esto se utilizará en la próxima sección de código como los nombres de las columnas de la tabla donde se almacenarán los datos dentro de una base de datos SQLite.

3.2.2.7 PARTE 6: Creación de la base de datos

Esta sección del código, que corresponde al anexo B.6, se encarga de crear una base de datos SQLite para almacenar datos DICOM anonimizados. La función ***create_database*** toma dos argumentos: la ruta donde se creará o conectará la base de datos SQLite y una lista de nombres de atributos que se utilizarán para definir la estructura de la tabla en la base de datos.

Primero, se establece una conexión a la base de datos SQLite especificada, o se crea una nueva si esta no existe, y se crea un cursor para ejecutar consultas SQL.

Después, se procede a definir la estructura de la tabla que se utilizará para almacenar los datos DICOM. Esto se hace tomando como nombre de las columnas la lista de los atributos único extraída con el código definido en la sección 3.2.2.6. Se realizan ajustes en los nombres de columna, como reemplazar espacios, comillas, barras y otros caracteres que puedan causar conflictos. Además, se determinan los tipos de datos de las columnas en la tabla. Los valores de elementos DICOM con VR siendo 'IS' (Integer String) o 'SL' (Signed Long) se almacenan como INTEGER (número entero) en la base de datos, mientras que los valores 'DA' (Date) se almacenan como DATE (fecha). El resto se almacena como TEXT (texto).

A continuación, se crea una sentencia SQL para crear la tabla en la base de datos. Esta sentencia CREATE TABLE incluye las columnas determinadas por los nombres de los atributos.

La sentencia SQL se ejecuta para crear la tabla en la base de datos SQLite y, posteriormente, se cierra la conexión con la base de datos.

3.2.2.8 PARTE 7: Inserción de datos en la base de datos

La función ***insert_dicom_data***, cuyo código se encuentra en el anexo B.7, se encarga de insertar los datos DICOM anonimizados en la base de datos SQLite creada en la parte anterior. Esta función toma dos argumentos de entrada: la ruta/nombre de la base de datos y una lista de datos DICOM.

Lo primero que se hace es establecer una conexión con la base de datos, especificada en los argumentos de entrada, y crear un objeto cursor para poder realizar consultas SQL.

Después, se crea un bucle para iterar por los elementos de la lista de datos y se preparan las declaraciones SQL para insertar los mismos en la base de datos. Para ello, se inicializan tres listas vacías. La primera se utiliza para almacenar las declaraciones SQL completas para cada objeto DICOM, la segunda es para construir la parte inicial de la declaración SQL (INSERT INTO) y la última se utiliza para almacenar los valores correspondientes a cada columna en la base de datos.

Para cada elemento de la lista de datos DICOM, se realizan una serie de transformaciones en el nombre de la columna para eliminar caracteres que puedan causar conflictos o problemas en la consulta SQL. Se itera a través de cada uno de estos para determinar si los atributos que contiene se corresponden con los nombres de las columnas de la tabla donde se quieren insertar los datos. En caso de que coincidan se inserta el valor en la columna correspondiente. En cambio, si son distintos, se establece el valor como vacío o nulo ('').

Una vez relacionados los valores con sus columnas correspondientes, se construye la declaración SQL completa combinando la parte INSERT INTO y los valores. Para cada conjunto de datos se genera una sentencia y se añaden a una lista.

Finalmente, se ejecuta cada una de las sentencias para insertar los datos de cada conjunto en una fila de la tabla. Se llama a *conn.commit()* para confirmar los cambios en la base de datos y, después, se cierra la conexión a la misma.

3.2.2.9 PARTE 8: Aplicación de Privacidad Diferencial

Aplicar privacidad diferencial a datos en formato DICOM implica introducir ruido en los datos de forma controlada para proteger la privacidad del individuo y, a su vez, preservar la utilidad de los datos para su análisis.

En esta parte del código, que se encuentra en el anexo B.8, se define una función llamada **apply_differential_privacy**, que toma como argumentos cinco parámetros: la ruta de la base de datos, el nombre de la tabla, los nombres de las columnas a las que se aplicará la privacidad diferencial, el valor de *epsilon* (que controla la cantidad de privacidad) y la sensibilidad de la consulta.

En primer lugar, la función conecta a la base de datos SQLite especificada y crea un cursor para ejecutar comandos SQL en la base de datos. Se itera a través de las columnas especificadas, y para cada una de ellas se ejecuta una consulta SQL para recuperar los valores de esa columna junto con su identificador de fila. Para cada fila en la columna, se calcula el ruido de Laplace y se le suma a los datos originales. Esto introduce ruido de manera controlada para proteger la privacidad de los datos.

Después de calcular los valores ruidosos para una columna, se ejecuta una consulta SQL de actualización para modificar el valor en la base de datos.

Finalmente, después de procesar todas las columnas y filas, los cambios se confirman en la base de datos utilizando *conn.commit()* y se cierra la conexión con la base de datos.

3.3. Evaluación del Método

En el apartado anterior (sección 3.2.2) se define en detalle el código implementado en este trabajo para leer, preprocesar, anonimizar y aplicar privacidad diferencial a datos extraídos de la cabecera de una serie de ficheros en formato DICOM.

Debido a que los ficheros utilizados provienen de diferentes estudios en los que se utilizaron diferentes dispositivos de radiología, imágenes cardiológicas y radioterapia (PET, CT, MRI...), el número de atributos total que se encontraban dentro de estos era abundante. Es por esto que se tuvo que realizar una investigación extensa sobre qué atributos se consideran sensibles e identificadores directos, cuáles se podrían clasificar como pseudoidentificadores, y cuáles no se consideran un riesgo a la privacidad del individuo.

Antes de poder aplicar la privacidad diferencial a este tipo de datos fue necesario implementar **métodos de anonimización y enmascaramiento** sobre los datos considerados como identificadores directos. En el contexto de imágenes DICOM, los atributos se pueden clasificar como público o privados, siendo los primeros los que están definidos en el estándar DICOM y los segundos los que están definidos por implementaciones específicas de sistemas DICOM y no están estandarizados en el estándar DICOM. Al no encontrarse estos en el estándar, se decidió eliminarlos. Del resto de atributos, se enmascararon los pertenecientes a personas, los números de teléfono y las direcciones, sustituyendo sus valores por la palabra *anonymous*, y el id de la persona por la palabra *id*, aunque en la práctica se le asignaría un identificador aleatorio para poder distinguir los datos de diferentes individuos sin ser ligados directamente a ellos. Con esto se garantiza que no haya presente ningún atributo que se pueda ligar directamente a la persona a la que pertenecen los datos. Las fechas también se trataron de tal forma que se mantiene el año pero se generalizan el mes y día, sustituyendo los valores originales por '01-01' para ayudar a mantener los datos los más seguros posibles.

Una vez hecho esto se guardaron todos los datos en una **base de datos**. Con esto se pretendía hacerlos más accesibles y fáciles de manipular, así como aumentar su seguridad al estar contenidos en una base de datos centralizada en la que, si se llegase a hacer pública, se le podría dotar de una serie de restricciones de acceso.

Se intentó aplicar **k-anonimato** sobre los atributos (columnas de la base de datos) que se considerasen como pseudoidentificadores, pero debido a la falta de conocimiento del contenido de los datos y, se cree, debido a la propia naturaleza de los datos seleccionados, no se obtenían los resultados esperados, por lo que después de realizar varias pruebas también se descartó el uso de este método.

En el momento de aplicar **privacidad diferencial**, objetivo principal de este trabajo, se presentaron una serie de dificultades. En primer lugar, se valoró el uso de la librería *OpenDp*. Esta librería proporciona una estructura para aplicar técnicas de privacidad diferencial en proyectos Python y permite mejorar la privacidad de los datos al agregar ruido controlado o modificar los resultados de consultas para proteger la información sensible mientras se permite el análisis estadístico [34]. Al intentar instalar la librería junto con sus dependencias necesarias, el programa donde se estaba ejecutando el código Python no era capaz de reconocer los métodos que se querían utilizar por lo que, después de varios intentos de solventarlo, se decidió proseguir sin la implementación de esta.

Como sustitución de esta librería, se utilizaron las librerías de *NumPy* y *random*. Sin embargo, usando el método generado con estas también surgieron una serie de problemas. Al aplicarlo, por ejemplo, sobre las columnas “Patient_Sex” y “Patient_Weight”, ambos considerados pseudoidentificadores, se observó que el aparecían errores tras la ejecución. Estos indicabas que el tipo de dato al que se quería aplicar la privacidad diferencial no era el esperado. Tras hacer varias pruebas se dedujo que el método sólo aceptaba valores que correspondían a números enteros. Esto quiere decir que se debe hacer un procesamiento de los datos más extenso del realizado para que la privacidad diferencial se pueda aplicar correctamente.

4. Conclusión

La gestión de datos en formato DICOM utilizando Python presenta desafíos significativos debido a la complejidad que presenta este tipo de datos médicos y a la variabilidad de los atributos presentes en diferentes ficheros. La clasificación precisa de estos atributos y la implementación de técnicas de anonimización y desidentificación son fundamentales para cumplir con las regulaciones de privacidad y garantizar la protección de la información del paciente, especialmente cuando se carece de conocimientos previos sobre el tipo de datos presentes en estos ficheros médicos.

El objetivo principal de este trabajo era investigar el uso de la privacidad diferencial en datos de imágenes DICOM. Sin embargo, a pesar del desarrollo de un método que inicialmente parecía efectivo, se ha demostrado que su implementación es más compleja de lo anticipado. La revisión bibliográfica realizada reveló que no existe un método claro para clasificar los atributos en identificadores directos, indirectos o datos no sensibles en el contexto de datos DICOM, excepto el estándar proporcionado por NEMA [10], el cual puede resultar complejo de entender. La mayoría de los estudios revisados coinciden en que se debe seguir el estándar y las especificaciones propuestas por cada estudio, lo que dificulta la desidentificación genérica de datos DICOM para uso público.

La mera aplicación de técnicas de privacidad diferencial no es suficiente. Es necesario combinarlas con otros métodos que aborden los riesgos potenciales de desidentificación a partir de la propia imagen, así como evaluar adecuadamente el impacto de los atributos privados en la privacidad del paciente. Además, se debe abordar el procesamiento adecuado de secuencias de datos, que presentan una estructura más compleja, y tratar los datos de píxeles de manera diferente a otros tipos de datos. Estos desafíos trascienden el alcance de este trabajo pero son fundamentales para su futura investigación.

En resumen, la desidentificación de datos en archivos DICOM es fundamental para proteger la privacidad del paciente y requiere un estudio exhaustivo para aplicar con éxito diversas técnicas de anonimización. Si bien la privacidad diferencial puede ser beneficiosa, su implementación es delicada debido a la necesidad de tratar previamente los datos identificables, tanto directos como indirectos, para evitar cualquier filtración que comprometa la privacidad del paciente. Este trabajo sienta las bases para investigaciones futuras en esta área de vital importancia.

Bibliografía

[1] NEMA. “DICOM Standard”. Available from: https://www.dicomstandard.org/.

[2] “Welcome to Python”. Available at: https://www.python.org/.

[3] BIDGOOD ET AL. “Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging”. Journal of the American Medical Informatics Association / Volume 4 / Number 3, May / Jun 1997.

[4] Dandu Ravi Varma. “Managing DICOM images: Tips and tricks for the radiologist”. Indian Journal of Radiology and Imaging / Vol 22 / Issue 1, February 2012.

[5] Ankush Malik. “Decoding DICOMs: A basic overview of what DICOMs are and how to process them.” Medium. Available from: https://aiau.medium.com/decoding-dicoms-b6cab5a31186, May 2021.

[6] Scientific Figure on ResearchGate. “Comparative Study of DICOM Files Handling Software’s: Study Based on the Anatomage Table”. Available from: https://www.researchgate.net/figure/Structure-of-a-DICOM-image-file-11_fig1_337669670.

[7] Kushida CA et al. “Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies.” Medical Care DOI: 10.1097/MLR.0b013e3182585355), July 2012.

[8] Emanuel Trucco Emilia R. Jefferson. “Chapter 20 - The challenges of assembling, maintaining and making available large data sets of clinical data for research”. Version of Records. DOI: https://doi.org/10.1016/B978-0-08-102816-2.00021-6, November 2019.

[9] Ivoline Ngong. “Maintaining Privacy in Medical Data with Differential Privacy”. Available from: https://blog.openmined.org/maintaining-privacy-in-medical-data-with-differential-privacy/, May 2020.

[10] NEMA. “Attribute Confidentiality profiles.” DICOM standard part 15 appendix E.

[11] Harvard University. “Differential Privacy”. Available from: https://privacytools.seas.harvard.edu/differential-privacy.

[12] Kobbi Nissim et al. “Differential Privacy: A Primer for a Non-technical Audience”. This document is the product of a working group of the Privacy Tools for Sharing Research Data project at Harvard University (http://privacytools.seas.harvard.edu), February 2020.

[13] David Darais Joseph Near and Kaitlin Boeckl. “Differential Privacy for Privacy-Preserving Data Analysis: An Introduction to our Blog Series”. CIBERSECURITY INSIGHT a NIST blog. Available from: https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-privacy-preserving-data-analysis-introduction-our, July 2020.

[14] Andreas Kopp. “Microsoft SmartNoise Differential Privacy Machine Learning Case Studies”, 2021.

[15] WeiKang Liu et al. “A Survey on Differential Privacy for Medical Data Analysis”. Annals of Data Science: https://doi.org/10.1007/s40745-023-00475-3, May 2023.

[16] Sophie Stalla-Bourdillon. “What is Data De-identification and Why is It Important?” IMMUTA, Available at: https://www.immuta.com/blog/what-is-data-de-identification/, April 2023.

[17] David A. Clunie et al. “Report of the Medical Image De-Identification (MIDI) Task Group - Best Practices and Recommendations”, April 2023.

[18] Alexander Ziller et al. “Medical imaging deep learning with differential privacy”. Scientific Reports nature (portfolio): https://doi.org/10.1038/s41598-021-93030-0, 2021.

[19] Danni Yuan et al. “Collaborative Deep Learning for Medical Image Analysis with Differential Privacy”. Available at: https://mason.gmu.edu/ mwei2/publication/19gc-yzw.pdf.

[20] Adnan et al. “Federated learning and differential privacy for medical image analysis”. Nature Scientific Reports. DOI: https://doi.org/10.103/s41598-022-05539-7, 2022.

[21] WeiKang Liu et al. “A Survey on Differential Privacy for Medical Data Analysis”. Available from: https://www.researchgate.net/figure/Differential-privacy-application-to-medical-data_fig2_371473842.

[22] C. Dwork and A. Roth. “The Algorithmic Foundations of Differential Privacy”. Foundations and Trends in Theoretical Computer Science Vol. 9, Nos. 3-4 (2013). DOI: 10.1561/0400000042, 2014.

[23] Chiké Abuah Joseph P. Near. “Programming Differential Privacy”. A book about differential privacy, for programmers, November 2022.

[24] SQLite. “About SQLite.” Available at: https://www.sqlite.org/about.html, April 2022.

[25] SQLite. “SQLite As An Application File Format.” Available at: https://www.sqlite.org/aff_short.html, April 2022.

[26] Clark K et al. “The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository”. Journal of Digital Imaging. 26(6):1045-57. DOI: 10.1007/s10278-013-9622-7, December 2013.

[27] M. et al. Rutherford. “Dataset from Medical Imaging De-Identification Initiative (MIDI)”. The Cancer Imaging Archive, DOI: https://doi.org/10.7937/s17z-r072, 2021.

[28] “Pydicom”. Available at: https://pydicom.github.io/, January 2023.

[29] NumPy Developers. “NumPy documentation”. Available at: https://numpy.org/doc/stable/, 2008-2022.

[30] Inc. NumFOCUS. “pandas documentation”. Available at: https://pandas.pydata.org/docs/, August 2023.

[31] Python Software Foundation. “os — Interfaces misceláneas del sistema operativo”. Available at: https://docs.python.org/es/3.10/library/os.html, 2001-2023.

[32] Python Software Foundation. “random — Generate pseudo-random numbers”. Available at: https://docs.python.org/3/library/random.html, 2001-2023.

[33] Python Software Foundation. “sqlite3 — DB-API 2.0 interfaz para bases de datos”. Available at: https://docs.python.org/es/3/library/sqlite3.html, 2001-2023.

[34] Salil Vadhan Michael Hay, Marco Gaboardi. “OpenDP”. Available at: https://pypi.org/project/opendp/, August 2023.

[35] NEMA. “DICOM PS3.5 2013 - Data Structures and Encoding”. Available at: https://dicom.nema.org/dicom/2013/output/chtml/part05/sect_6.2.html, 2013.

Anexo

A. Representación de Valores (VR) DICOM

A continuación se muestra una tabla con los códigos VR que pueden tener cada uno de los atributos [35].

VR Code	VR Name
AE	Application Entity
AS	Age String
AT	Attribute Tag
CS	CodeString
DA	Date
DS	Decimal String
DT	Date Time
FL	Floating Point Single
FD	Floating Point Double
IS	Integer String
LO	Long String
LT	Long Text
OB	Other Byte String
OD	Other Double String
OF	Other Float String
OW	Other Word String
PN	Person Name
SH	Short String
SL	Signed Long
SQ	Sequence of Items
SS	Signed Short
TM	Time
UI	Unique Identifier (UID)
UL	Unsigned Long
UN	Unknown
US	Unsigned Short
UT	Unlimited Text

B. Código Python

```python
# Import necessary libraries
import os
import pydicom
import sqlite3
import pandas as pd
from opendp.trans import make_base_stability
from opendp.meas import make_base_laplace
from opendp.typing import L1Distance
import random
import numpy as np
```

B.1. Búsqueda de directorios de ficheros DICOM

```python
#########################################################
# PART 1: Find all the dicom file directories           #
#########################################################

# Find dicom files in a given directory and store the file directories in a list
def get_dicom_files(dicom_dir):
    file_names = []
    for root, dirs, fnames in os.walk(dicom_dir):
        file_names += list(os.path.join(root, f) for f in fnames if f.endswith('.dcm'))

    print(f'There are {len(file_names)} DICOM files in the speicfied directory.')
    return file_names
```

B.2. Lectura de los ficheros

```python
#########################################################
# PART 2: Read dicom files                              #
#########################################################

# Read dicom files and store the data for each file in a list
def read_dicom_files(dcm_files):
    data_list = []  # Create an empty list to store DICOM data
    for file in dcm_files:
        data = pydicom.dcmread(file, force=True)
        data_list.append(data)  # Append the data to the list

    print('All data has been saved to data_list.')
    return data_list
```

B.3. Identificación y eliminación de secuencias

```
#################################################################
# PART 3: Identify and remove sequences                         #
#################################################################

def process_dicom_data(dcm_data, sequences_to_remove=None):
    # Create a set to store the unique tags (element identifiers) of
    attributes that are sequences
    unique_sequence_attribute_tags = set()

    # Create an empty list to store the new DICOM datasets
    new_dcm_data = []

    # If sequences_to_remove is not provided, initialize it as an empty
    list
    if sequences_to_remove is None:
        sequences_to_remove = []

    # Iterate through each DICOM dataset in the list
    for idx, dataset in enumerate(dcm_data):
        # Create a new dataset without the specified sequences
        new_dcm = pydicom.Dataset()

        # Iterate through all elements (attributes) in the DICOM dataset
        for elem in dataset:
            # Check if the attribute is a sequence based on its Value
    Representation (VR)
            if elem.VR == 'SQ':
                # Add the tag (element identifier) to the set
                unique_sequence_attribute_tags.add(elem.tag)
            else:
                # Check if the element's tag is not in the
    sequences_to_remove
                if elem.tag not in sequences_to_remove:
                    new_dcm.add(elem)

        # Set byte order and VR encoding
        new_dcm.is_little_endian = True  # Set to True for little endian
    (most common)
        new_dcm.is_implicit_VR = True    # Set to True for implicit VR
    encoding

        # Generate the new filename for each dataset
        new_filename = f"modified_dicom_file_{idx}.dcm"

        # Save the modified DICOM dataset to the new file
        new_dcm.save_as(new_filename)

        # Append the modified DICOM dataset to the list
        new_dcm_data.append(new_dcm)

    # Return the new dataset
    return new_dcm_data
```

B.4. Anonimización de datos sensibles

```
########################################
# PART 4: Anonimize sensitive data     #
########################################

def anon_callback(dcm_mod_data):
    for dcm_data in dcm_mod_data:
        for data_element in dcm_data:
            # Remove private tags
            if data_element.tag.is_private:
                del dcm_data[data_element.tag]
            # Remove tags that have empty values
            if data_element.value == '':
                del dcm_data[data_element.tag]
            # Anonymize PatientID
            if data_element.tag == (0x0010, 0x0020):
                data_element.value = "id"
            # Anonymize PixelData (set it to "NA")
            if data_element.tag == (0x7FE0, 0x0010):
                data_element.value = "NA"
            # Anonymize Person Names, Telephone Numbers and Addresses
            if data_element.VR == "PN":
                data_element.value = "anonymous"
            elif "Telephone" in data_element.name:
                data_element.value = "anonymous"
            elif "Address" in data_element.name:
                data_element.value = "anonymous"
            # Dates -> Maintain only the year (generalization)
            elif data_element.VR == "DA":
                data_element.value = data_element.value[:4] + '0101'

    return dcm_mod_data
```

B.5. Lista de todos los atributos únicos

```
#########################################################
# PART 5: List of unique attributes + VR                #
#########################################################

def extract_unique_attribute_names_with_vr(dcm_anon):
    attribute_names_with_vr = set()

    # Iterate through each DICOM dataset in the list
    for dataset in dcm_anon:
        for elem in dataset:
            # Check for specific VR values
            vr = elem.VR if elem.VR in ["DA", "IS", "SL"] else ""
            # Add the attribute name and VR as a tuple to the set
            attribute_names_with_vr.add((elem.name, vr))
    # Convert the set to a list
    unique_attributes_list = list(attribute_names_with_vr)
    # Return the list containing unique attribute name and VR pairs
    return unique_attributes_list
```

B.6. Creación de una base de datos SQLite

```
############################################################
# PART 6: Create SQLite database to store the DICOM data #
############################################################

def create_database(database_path, attribute_names):
    # Connect to the SQLite database or create one if it doesn't exist
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()

    # Create a table to store DICOM data with dynamic column names
    create_table_sql = 'CREATE TABLE IF NOT EXISTS dicom_data (id INTEGER
    PRIMARY KEY'

    # Use attributes as column names
    for elem in attribute_names:
        # Replace certain character that create conflicts
        column_name = elem[0].replace(' ', '_')
        column_name = column_name.replace("'s", '')
        column_name = column_name.replace("/", '')
        column_name = column_name.replace("__", '_')
        column_name = column_name.replace("___", '_')
        column_name = column_name.replace("-", '')
        column_name = column_name.replace(")", '')
        column_name = column_name.replace("(", '')

        data_type = 'TEXT'
        if elem[1] == 'IS' or elem[1] == 'SL':
            data_type = 'INTEGER'
        elif elem[1] == 'DA':
            data_type = 'DATE'

        # Add the attribute name to the CREATE TABLE statement
        create_table_sql += f', "{column_name}" {data_type}'

    create_table_sql += ');'
    #print(create_table_sql)

    try:
        # Execute the CREATE TABLE statement
        cursor.execute(create_table_sql)
        conn.commit()
    except sqlite3.Error as e:
        print(f"Error creating table: {e}")
    finally:
        # Close the connection
        conn.close()
```

B.7. Inserción de los datos en la base de datos

```
#########################################################
# PART 7: Insert data into database                     #
#########################################################

def insert_dicom_data(database_path, dcm_anon):
    conn = sqlite3.connect(database_path)
    cursor = conn.cursor()

    for dcm in dcm_anon:
        statement_list = []
        # Generate the INSERT INTO statement for each DICOM dataset
        insert_sql = 'INSERT INTO dicom_data ('
        values_sql = 'VALUES ('
        values = []

        for elem in dcm.iterall():
            # Replace certain characters that may create conflicts
            column_name = elem.name.replace(' ', '_')
            column_name = column_name.replace("'s", '')
            column_name = column_name.replace("/", '')
            column_name = column_name.replace("__", '_')
            column_name = column_name.replace("___", '_')
            column_name = column_name.replace("-", '')
            column_name = column_name.replace(")", '')
            column_name = column_name.replace("(", '')

            # Check if the element exists in the DICOM dataset
            if elem.tag in dcm:
                insert_sql += f'"{column_name}", '
                values.append(str(dcm[elem.tag].value))
            else:
                # If the element does not exist, add a blank value
                insert_sql += f'"{column_name}", '
                values.append('')  # Append to values list

        insert_sql = insert_sql.rstrip(', ') + ')'
        values_sql += ', '.join(f'"{value}"' for value in values)
        values_sql += ');'

        # Combine the INSERT INTO and VALUES clauses
        sql_statement = f'{insert_sql} {values_sql}'

        statement_list.append(sql_statement)

        try:
            # Execute the SQL statement with the concatenated values
            for i in statement_list:
                cursor.execute(sql_statement)
        except sqlite3.Error as e:
            print(f"Error inserting DICOM data: {e}")

    # Commit the changes and close the connection
    conn.commit()
    conn.close()
```

B.8. Aplicación de privacidad diferencial

```
############################################################
# PART 8: Differential Privacy application                 #
############################################################

def apply_differential_privacy(db_path, table_name, column_names, epsilon
    , sensitivity):
    # Connect to the SQLite database
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()

    for column_name in column_names:
        # Retrieve the data from the database
        cursor.execute(f"SELECT rowid, {column_name} FROM {table_name}")
        rows = cursor.fetchall()

        for row in rows:
            row_id, column_data = row
            # Calculate the scale parameter for Laplace noise
            scale = sensitivity / epsilon

            # Generate Laplace noise
            laplace_noise = np.random.laplace(0, scale, 1)[0]

            # Add noise to the column
            noisy_data = column_data + laplace_noise

            # Update the database with the noisy column
            cursor.execute(f"UPDATE {table_name} SET {column_name} = {
    noisy_data} WHERE rowid = {row_id}")

    # Commit the changes and close the connection
    conn.commit()
    conn.close()
```