Minería de Datos con Preservación de Privacidad como Servicio mediante Cifrado Homomórfico en el contexto de Big Data

Shanel Daniela Reyes Palacios, Cinvestav
Miguel Morales Sandoval, INAOE
José Juan García Hernández, Cinvestav
Heidy M. Marin Castro, UDLAP
José Luis González Compeán, Cinvestav

Reporte Técnico No. CCC-24-001

28 de Mayo de 2024

Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.

Minería de Datos con Preservación de Privacidad como Servicio mediante Cifrado Homomórfico en el contexto de Big Data

Abstract

La minería de datos es un campo de la ciencia de datos que tiene como uno de sus objetivos el descubrimiento de patrones, tendencias o reglas en conjuntos de datos, lo que resulta útil para la toma de decisiones. Dos de las tareas más representativas en minería de datos son la agrupación y la clasificación. La primera se refiere a identificar patrones o estructuras inherentes en un conjunto de datos dado, agrupando elementos que son más parecidos entre sí. La segunda tarea consiste en entrenar a un algoritmo para aprender a clasificar los datos en distintas categorías, de acuerdo con las características que dichos datos presentan.

El concepto de Big Data se ha acuñado para referirse a cualquier cantidad voluminosa de datos estructurados, semiestructurados y no estructurados, producidos y/o disponibles desde diversas fuentes tales como medios digitales, redes sociales, redes de sensores, servicios web, aplicaciones comerciales, entre otros. Esta producción acelerada de datos en formato digital se ha acentuado en gran parte debido al desarrollo y auge de los modelos del Internet de las Cosas y del Cómputo en la Nube.

Ante esta creciente disponibilidad de datos, de su volumen y por ende del poder computacional para procesarlos mediante algoritmos y modelos de minería de datos, ha surgido el concepto de minería de datos como servicio (DMaaS, por sus siglas en inglés), donde el propietario de los datos delega al proveedor del servicio en la nube, los requerimientos de almacenamiento y de procesamiento implicados por la ejecución de algoritmos de minería de datos. Sin embargo, cuando la naturaleza de los datos es sensible, por ejemplo datos médicos, financieros o personales, o debido a regulaciones o leyes en materia de protección de datos en posesión de terceros, o incluso por decisión del propietario de los datos, se requiere garantizar servicios de seguridad y privacidad, ya que el acceso y uso de datos confidenciales puede llevar al proveedor del servicio o al mismo propietario de los datos a incurrir en violaciones o ataques. En la literatura se han propuesto algunos enfoques para garantizar la seguridad y la privacidad de los datos en entornos DMaaS, mediante el uso de métodos de Minería de Datos con Preservación de Privacidad (PPDM, por sus siglas en inglés). Sin embargo, existen aún limitantes en lo que se refiere a eficiencia, seguridad y factibilidad de estos métodos en el contexto de DMaaS.

En este reporte se presentan el diseño, construcción, implementación y resultados de validación y evaluación de una plataforma basada en la nube como entorno de ejecución eficiente, seguro y factible de tareas de minería de datos con preservación de privacidad bajo el contexto de DMaaS y Big Data. A este enfoque de procesamiento le denominamos Minería de Datos con Preservación de Privacidad como Servicio (PPDMaaS). Esta plataforma se construyó incorporando mecanismos de procesamiento paralelo y distribuido para soportar grandes volúmenes de datos. Soporta distintos PPDM basados en cifrado homomórfico, capaces de operar a distintos niveles de seguridad. Los PPDM (generalmente fuera del

contexto PPDMaaS) se han validado usando niveles de seguridad obsoletos (menos de 128 bits) y se han ejecutado en un contexto local con conjuntos de datos mucho menores a los que se pueden encontrar en un contexto de Big Data. La plataforma propuesta le permite al propietario de datos simplificar un flujo PPDMaaS que involucra las tareas de aseguramiento de datos, carga de datos protegidos al proveedor del servicio, ejecución de los PPDM del lado del proveedor del servicio y acceso seguro al modelo descubierto, este último, cifrado en el lado del proveedor y con posibilidad de descifrarse en el lado del usuario final, autorizado.

Un prototipo de la plataforma propuesta se construyó usando tecnología de virtualización, por lo que puede desplegarse en cualquier infraestructura. Para las pruebas de validación y evaluación, el prototipo usó 12 servidores. Sobre el prototipo PPDMMaaS se ejecutaron diferentes PPDM tanto para la tarea de agrupamiento como de clasificación, usando 16 conjuntos de datos obtenidos del repositorio UCI [1], para evaluar la utilidad y eficacia. Esto es, se validó que los resultados de minería de datos no se afectaran por el uso de datos cifrados en lugar de los datos en claro.El prototipo fue evaluado en términos desempeño. Esto es importante debido al sobre-costo por las tareas de cifrado y por la sobrecarga en los algoritmos PPDM al usar datos cifrados en lugar de los datos en claro, siendo los datos cifrados más grandes y las operaciones subyacentes (sobre datos cifrados y no sobre los datos en claro) más complejas. Para ello y con el objetivo de apegarse a un escenario de Big Data, se usaron 50 conjuntos de datos artificiales suficientemente grandes, en el orden de giga bytes.

Con base en los resultados experimentales, se concluye la viabilidad de la plataforma propuesta, en términos de simplicidad de uso, eficiencia, seguridad, flexibilidad y escalabilidad.

Índice

1. Introducción

Con el predominio del internet, el comercio electrónico, redes sociales, y un proceso de digitalización en aumento, la cantidad de datos en formato digital ha crecido considerablemente en los últimos años. Se estima que todos los días se crean aproximadamente 2.5 quintillones de bytes de datos [2]. En el 2020, existían aproximadamente 44 zettabytes de datos en el mundo; para el 2025, la cantidad estimada de datos creados diariamente se estima será de 463 exabytes. Por lo que es probable que el volumen de datos para 2025 sea de alrededor de 175 zettabytes [3].

Los datos han demostrado ser un activo muy importante para descubrir conocimiento que sirva para la toma de decisiones. Hoy en día, se manejan conceptos como *economía basada en datos* o *economía basada en conocimiento* [4]. El análisis y extracción de conocimiento a partir de grandes cantidades de datos ha mostrado resultados importantes. Por ejemplo, en el campo de la salud, el análisis realizado en grandes conjuntos de datos (proporcionados por aplicaciones como registros médicos, electrónicos y sistemas de decisiones clínicas) pueden permitir que los profesionales de la salud brinden soluciones más efectivas y asequibles para los pacientes al tomar en cuenta datos históricos y las correlaciones que estos pudieran tener con otros casos [5].

Bajo el contexto de la disponibilidad de grandes volúmenes de datos y de las tareas para su manejo y análisis, ha surgido el concepto de Big Data [6], que se refiere a conjuntos de datos de rápido crecimiento con tamaños que superan la capacidad de las herramientas de bases de datos tradicionales para almacenarlos, administrarlos y analizarlos. Este análisis es difícil de realizar utilizando el análisis de datos tradicional, debido a las características de las cinco V's del Big Data: alto *V*olumen, baja *V*eracidad, alta *V*elocidad, alta *V*ariedad y alto *V*alor [7]. Bajo este escenario, el objetivo de las técnicas de análisis de datos avanzadas es descubrir información, patrones ocultos y correlaciones desconocidas en conjuntos de datos masivos [8]. Por ejemplo, un análisis detallado de los datos históricos de los pacientes podría conducir a la detección de enfermedades destructivas en una etapa temprana, lo que permitiría una cura o un plan de tratamiento personalizado y óptimo [9]. Dentro de las técnicas de análisis de datos, la minería de datos se ha vuelto fundamental para que muchas industrias extraigan información no trivial de conjuntos de datos enormes para respaldar sus operaciones centrales y procesos de toma de decisiones.

Sin embargo, analizar un alto volumen de datos demanda una gran capacidad de almacenamiento y a la vez, un mayor poder de procesamiento para preparar y ejecutar algoritmos de análisis concretos. En respuesta a esta demanda, el cómputo en la nube ha impulsado el crecimiento del análisis y la gestión de datos como servicio, ya que garantiza flexibilidad a los propietarios de los datos con respecto a la escalabilidad del cómputo y el almacenamiento [10]. Gracias a ello, cada vez más propietarios de datos están dispuestos a reducir sus costos no solo de almacenamiento, sino también de procesamiento, delegando estas tareas a un proveedor de servicios en la nube. Este contexto ha permitido el surgimiento del concepto de Minería de Datos como Servicio (DMaaS, por sus siglas en inglés), donde una o varias entidades hacen disponibles sus datos a un proveedor del servicio para su análisis [11]. En este escenario, la externalización de los datos a un tercero honesto, pero curioso, conlleva una pérdida de las garantías de confidencialidad de dichos datos [12]. Debido a esto, la preservación de la confidencialidad y privacidad de los datos se convierte en uno de los requerimientos de mayor prioridad en DMaaS [13].

1.1. Preservación de privacidad en minería de datos como servicio

La minería de datos se encarga de identificar patrones y tendencias de datos para obtener información útil para la toma de decisiones. Las tareas más importantes en la minería de datos son: clasificación, regresión,

agrupamiento y asociación [14]. A su vez, la minería de datos como servicio es la encargada de integrar los conceptos de minería de datos con el cómputo en la nube. De manera general, un modelo de minería de datos como servicio involucra tres entidades:

1. *Propietario de los datos (PD):* Posee un conjunto de datos masivo, pero no dispone de recursos computacionales para la aplicación de algoritmos de minería de datos sobre dichos datos.

2. *Usuario del análisis de datos (UAD):* Accede a los resultados de los algoritmos de minería de datos. Este usuario puede ser el mismo PD u otra entidad externa.

3. *Proveedor del Servicio (PS):* Cuenta con los recursos computacionales para ofrecer el servicio de minería de datos sobre un conjunto masivo de datos delegado. Tiene acceso a los datos del propietario y también a los resultados de los algoritmos de minería de datos ejecutados.

DMaaS ofrece una solución rentable para el PD. Sin embargo, para los usuarios de dominios como instituciones médicas, organizaciones gubernamentales y establecimientos financieros, tanto sus datos sin procesar como los resultados del algoritmo de análisis pueden contener información confidencial. Por ello, compartir datos con un PS externo y permitirle que tome la custodia de los datos sensibles y de los resultados del análisis de los mismos es un problema de seguridad y privacidad. Cabe resaltar que el requerimiento de seguridad y privacidad de datos se está considerando cada vez más, e incluso existen leyes y regulaciones en relación con el manejo de datos sensibles que varían en alcance en cada país.

1.2. Problemática

El cómputo en la nube resulta atractivo y rentable para un PD cuando el procesamiento y análisis se realiza sobre grandes cantidades de datos. Sin embargo, existe un problema en la seguridad y privacidad de los datos al hacerlos disponibles a un tercero no confiable, que es honesto, pero curioso (PS) [15] (ver Fig. 1.2). Bajo este escenario, se requiere que los datos sean transferidos al PS, por lo que el PD perderá el control sobre la privacidad y la confidencialidad de los mismos. En este escenario es donde se presenta el problema abordado en este trabajo.

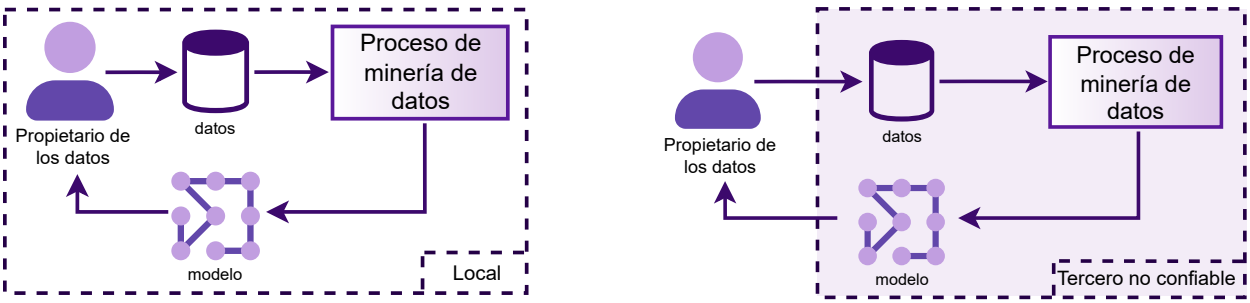

Figura 1. Escenario de minería de datos. A la derecha, sin riesgo de seguridad y privacidad (confiable), ya que los datos y el procesamiento están bajo el control del PD. A la izquierda, los datos y el procesamiento están bajo el control del PS (tercero no confiable).

El problema de seguridad y privacidad en minería de datos, en el contexto mostrado en la Fig. 1.2 se ha abordado mediante métodos de Minería de Datos con Preservación de Privacidad (PPDM, por sus siglas en inglés) [16]. Estos PPDM suponen que los datos del PD se transforman en una representación que no comprometa su confidencialidad, pero que dicha transformación no afecta el uso de los datos para seguir realizando las tareas de minería de datos (sin pérdida de utilidad). Una de las transformaciones más usadas y que ha demostrado ser de las más efectivas para PPDM es el cifrado [17].

En términos generales, el cifrado de datos [18] es una técnica de la criptografía que ha demostrado su eficiencia para prevenir accesos no autorizados a datos. El cifrado produce una transformación de los datos mediante el uso de llaves, que generalmente son secretos entre el que cifra los datos y el que está autorizado a acceder a los datos mediante el descifrado. En el contexto de DMaaS, el cifrado ha sido una solución al problema de seguridad y privacidad: el PD cifra sus datos antes de compartirlos al PS, el cual debe ser capaz de operar los datos cifrados como si lo hiciera con los datos en claro. Esto supone que los datos cifrados deben mantener propiedades que puedan ser explotadas por el algoritmo de minería de datos (por ejemplo, agrupamiento o clasificación), sin comprometer la privacidad.

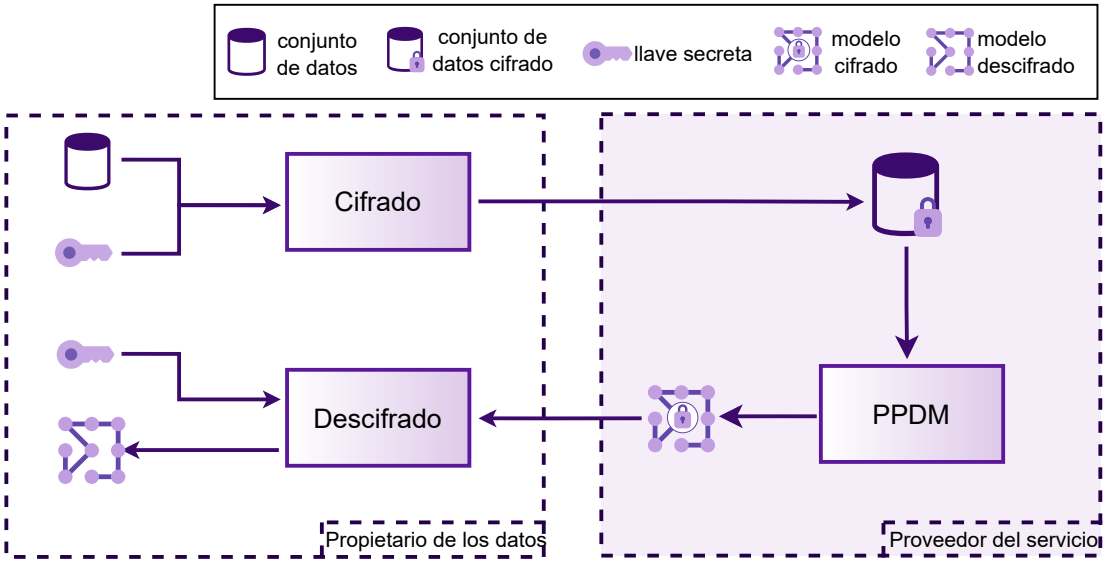

Figura 2. Vista general de *DMaaS* y uso de *PPDM* para garantizar la privacidad de los datos.

En la Figura 2 se describe el flujo para obtener un modelo de minería de datos de manera segura, bajo un enfoque PPDM basado en cifrado. PD cuenta con un conjunto de datos y sus llaves para cifrar y descifrar, previamente generadas. Por cuestiones de seguridad y confidencialidad de los datos, el cifrado y el descifrado son acciones realizadas por el mismo PD. Una vez generado el conjunto de datos cifrados, PD los hace disponibles al PS, quien ejecuta el algoritmo PPDM con sus recursos computacionales. El modelo obtenido del lado del PS por la ejecución del PPDM también estará cifrado, por lo que solo podrá ser accedido por PD haciendo uso de su llave de descifrado. El modelo en claro puede ser accedido por algún UAD siempre que cuente con la correspondiente llave de descifrado. La distribución de esta llave a los UAD no está considerada en los PPDM, pero podría implementarse mediante un mecanismo de seguridad como el re-cifrado proxy (PRE, por las siglas de Proxy Re-encryption) [19].

En este trabajo, la problemática abordada se centra en el contexto de DMaaS, donde el PD necesita compartir sus datos con un PS, pero haciéndolo de manera directa podría resultar en violaciones a la seguridad

y privacidad de los datos, además de que se requiere garantizar la no perdida de utilidad, desde la perspectiva del PS.

En este contexto, se plantean varios desafíos, tales como: a) *Seguridad*: se deben implementar mecanismos robustos para proteger los datos durante la transferencia y análisis de los mismos, asegurando que solo las partes autorizadas pueden acceder a ellos. b) *Privacidad*: es necesario preservar la privacidad de los datos del PD, garantizando que la información sensible no se revele durante el proceso de análisis. c) *Eficiencia*: es necesario contar con formas eficientes de compartir los datos de manera segura y privada desde el PD al PS, sin comprometer la utilidad de los datos durante el proceso de análisis.

El desafío principal radica en encontrar un equilibrio entre la necesidad de compartir los datos para el análisis y la necesidad de protegerlos, por lo que, la solución propuesta aborda estos desafíos, teniendo en cuenta, además, las limitaciones técnicas y los recursos disponibles tanto para el PD como para el PS garantizando un intercambio de datos seguro y útil en el contexto de DMaaS.

1.3. Contribuciones

La contribución principal de este trabajo radica en el diseño, construcción e implementación de una plataforma basada en la nube que soporta PPDMaaS. Esta plataforma incorpora mecanismos de procesamiento paralelo y distribuido para manejar grandes volúmenes de datos, así como distintos métodos de Minería de Datos con Preservación de Privacidad basados en cifrado homomórfico, con diferentes niveles de seguridad.

Se realizó una extensa evaluación experimental utilizando conjuntos de datos de tamaño significativo, lo cual permite una validación realista de la plataforma en el contexto de Big Data. Los resultados obtenidos sugieren que la plataforma para PPDMaaS tiene la capacidad de hacer frente a los desafíos de seguridad y privacidad en entornos de DMaaS y Big Data. Esto indica que la plataforma tiene el potencial de abordar eficazmente estos desafíos y de proporcionar soluciones efectivas para garantizar la seguridad y privacidad de los datos en entornos de análisis de datos en la nube y de Big Data.

1.4. Organización del documento

El resto del documento está organizado de la siguiente manera. En la Sección 2 se presentan algunos conceptos relacionados con el problema y enfoque de solución bajo estudio, como algoritmos de agrupamiento y de clasificación de datos, el cifrado homomórfico y los algoritmos PPDM. En la Sección 3 se discute el trabajo relacionado encontrado en la literatura. En la Sección 4 se describen detalles acerca de los PPDM seleccionados para evaluación de la plataforma propuesta, tanto para la tarea de agrupamiento como para la de clasificación. En la Sección 5 se describe la arquitectura de la plataforma propuesta para PPDMaaS, que incluye la descripción de los componentes implementados, los componentes necesarios para su funcionamiento, así como las interacciones entre estos componentes. En la Sección 6 se presenta los experimentos para validación y evaluación de una prototipo de la plataforma propuesta, describiendo los detalles de los mismos, datos de prueba, métricas y configuraciones seleccionadas. En la Sección 7 se discuten los resultados obtenidos tanto para la evaluación de utilidad como la de desempeño, considerando la especificación de los experimentos descritos en la Sección 6. Finalmente, la Sección 8 presenta las conclusiones y el trabajo futuro en el corto plazo.

2. Marco conceptual

En esta sección se describen conceptos relevantes relacionados con la minería de datos, así como la seguridad y privacidad de datos.

2.1. Minería de datos y principales tareas

La minería de datos es un campo de la estadística y las ciencias computacionales que se refiere al proceso de descubrir patrones, modelos o algún otro tipo de conocimiento en grandes volúmenes de conjuntos de datos [20, 21].

Un proceso de minería de datos generalmente consiste en cuatro etapas:

1. **Recopilación:** Los datos de interés para el objetivo de análisis se identifican y reúnen desde su ubicación (base de datos, sistema de archivos, sistemas de almacenamiento, etc.), generalmente, se trata de una combinación de datos estructurados y no estructurados [22, 23].

2. **Preprocesamiento:** Esta etapa incluye un conjunto de pasos que permite preparar los datos para mejorar su calidad [24, 5]: 1) La *limpieza* es el primer paso en la minería de datos. Datos con ruido o errores pueden producir resultados inexactos o confusión en la interpretación [25]. Por lo que la limpieza de datos incluye tareas como completar datos faltantes o eliminar datos ruidosos [24]. 2) La *integración* es una etapa que combina datos de múltiples fuentes de datos heterogéneas en un almacén de datos coherente y proporciona una vista unificada de los datos. Estas fuentes pueden incluir varios cubos de datos, bases de datos o archivos planos [26]. La integración de datos se puede realizar utilizando herramientas de migración de datos como Oracle, Data Service Integrator y Microsoft SQL [27]. 3) La *transformación* es la etapa donde los datos se consolidan para que la tarea de minería sea más eficiente y los patrones sean más fáciles de entender [28]. La transformación de datos implica el mapeo de datos y el proceso de generación de código. Las estrategias para la transformación de datos son: suavizado, agregación, normalización y discretización [29]. Y 4) la *selección de los datos* es la etapa donde se determina el tipo y la fuente de datos apropiados y los instrumentos adecuados para recopilar datos [30].

3. **Minería:** Una vez que se preparan los datos, se elige la técnica adecuada para realizar la minería de datos que permitirá identificar patrones y conocimientos interesantes a partir de los datos [31].

4. **Análisis y representación:** Este paso implica analizar los resultados de la etapa de minería e identificar los patrones de interés que representan el conocimiento, basado en medidas de interés. Los resultados de la minería de datos se utilizan para crear modelos analíticos que pueden ayudar a la toma de decisiones [21]. Mediante técnicas específicas de visualización se pueden representar e interpretar de mejor forma los resultados obtenidos en la etapa de minería [32].

La minería de datos abarca diferentes tareas para extraer datos y convertirlos en resultados útiles para diferentes aplicaciones de ciencia de datos. Entre las tareas más populares de minería de datos se encuentran:

- **Clasificación:** Asigna categorías o clases a datos de acuerdo con las características que estos presentan [33]. Se requiere un entrenamiento previo para que un clasificador pueda asignar la clase que le corresponde a nuevas instancias. Por ejemplo, un clasificador podría recibir como entrada una imagen, y con base en su entrenamiento previo, podría clasificar a esa imagen de acuerdo con una categoría

(paisaje, rostro, objeto, etc.) de acuerdo con las características que tenga esta nueva instancia. Los árboles de decisión, Naive Bayes, vecino más cercano y la regresión logística son algunos ejemplos de métodos para la tarea de clasificación [34].

- **Agrupamiento:** Dado un conjunto de datos, el agrupamiento hace una separación de esos datos en grupos o clústeres, basada en la similitud entre los datos [35]. Un algoritmo de agrupamiento usa una medida de similitud para colocar en el mismo grupo a aquellos datos con una similitud alta [36], sin necesidad de tener un entrenamiento previo, como en el caso de los clasificadores. El agrupamiento podría ser usado por un clasificador para iniciar con la fase de entrenamiento, siempre que sea posible asignar una clase de manera manual o automática a los grupos creados por la agrupación. Ejemplos de algoritmos de agrupamiento son k-means, agrupamiento jerárquico y los modelos de mezcla gaussiana [37].

- **Regresión:** Es un método estadístico utilizado para determinar la relación entre dos variables (regresión lineal simple) o, tres o más variables (regresión múltiple) [38]. Por ejemplo, se puede usar esta técnica para determinar como algunos factores del ambiente de una zona afectan la salud de una persona. Existen dos tipos de variables [39]: 1) variable dependiente: es el factor principal que se intenta comprender o predecir, y 2) variables independientes: los factores que tienen un impacto en la variable dependiente.

- **Reglas de asociación:** Son declaraciones *si-entonces* que permiten identificar relaciones entre los elementos de datos. Los criterios de soporte y confianza se utilizan para evaluar las relaciones: el soporte mide la frecuencia con la que aparecen los elementos relacionados en un conjunto de datos, mientras que la confianza refleja la cantidad de veces que una declaración *si-entonces* es precisa [40].

Para este trabajo, se han considerado abarcar tanto la tarea de agrupamiento como la de clasificación, dentro del enfoque de solución de contar con métodos de DMaaS seguros y eficientes.

2.1.1. Tarea de agrupamiento de datos

Dado un conjunto de datos $D = \{d_1, d_2, \cdots, d_n\}$, la tarea de agrupamiento consiste en formar grupos C_j con datos en D, similares entre sí, de manera automática. El objetivo es conformar subconjuntos disjuntos de D, con base en la similitud de dichos datos [41], de manera que los grupos formados deberían tener una alta cohesión. La tarea de agrupamiento de datos permite: a) analizar datos, b) encontrar posibles datos inconsistentes (outliers), y c) segmentar datos en grupos similares para facilitar un proceso de análisis más elaborado o usando otra técnica de aprendizaje automático o de inteligencia artificial [42].

k-means [43, 44] es uno de los algoritmos más populares para la tarea de agrupamiento. Permite agrupar un conjunto de datos, de manera simple e intuitiva [45]. Los grupos determinados con este algoritmo dependen de la selección inicial de *centroides*, los cuales son valores de referencia contra los que cada uno de los datos del conjunto se compara. Cada centroide define un grupo, y los datos se agregan al grupo cuyo centroide es el más parecido a ellos, con base en una medida de similitud. El algoritmo es iterativo, los centroides se recalculan en cada iteración de acuerdo con los datos que están contenidos hasta ese momento en dicho grupo. El proceso iterativo termina hasta que los centroides de la iteración actual ya no son diferentes a los centroides de la iteración siguiente. La selección de los centroides iniciales puede influir en la convergencia del algoritmo y en la calidad de los grupos finales [46].

Así, el funcionamiento de k-means se puede visualizar como un proceso de dos pasos llamado maximización de expectativas. El paso de *expectativa* asigna cada dato del conjunto a su centroide más

cercano. Luego, el paso de *maximización* calcula la media de todos datos en cada grupo y establece dicho resultado como el nuevo centroide de ese grupo. La estructura del algoritmo k-means se muestra en el Algoritmo 1.

Algorithm 1 k-means

```
1: procedure KMEANS(D, k)
2:     inicializar k centroides {cent_1, cent_2, ...cent_k}
3:     repeat
4:         for d ∈ D do
5:                 asignar d al grupo i siempre que sim(d, cent_i) es la más alta
6:         end for
7:         recalcular cada cent_i (media de cada grupo i)
8:     until cent_i no cambia, 1 ≤ i ≤ k
9: end procedure
```

k-means requiere del conjunto de datos original D y del número de agrupaciones k que intentará formar. El primer paso es seleccionar k datos de D como los k centroides iniciales. El segundo paso es asignar cada dato a su centroide más cercano. Esto se logra a partir del cálculo de la medida de similitud entre cada dato y los centroides. El dato se agregará al grupo del centroide más cercano o más similar. Posteriormente, se debe hacer un recálculo de los centroides de cada grupo, tomando el promedio de todos los datos que pertenecen a dicha agrupación. Este proceso de dos pasos se repetirá hasta que no existan más cambios en las agrupaciones (no haya cambios en los centroides de la iteración actual y la siguiente).

Otro de los algoritmos de agrupamiento más utilizado es el algoritmo nnc, o clustering de vecinos más cercanos, que realiza el agrupamiento basándose en la similitud de un dato con sus vecinos más cercanos. La estructura del algoritmo nnc se muestra en el Algoritmo 2. En primer lugar, el algoritmo toma dos parámetros de entrada: D y un $umbral$ μ. D representa el conjunto de datos a procesar, μ representa el límite a considerar para determinar si dos elementos en D son lo suficientemente cercanos. El proceso comienza creando el conjunto de agrupaciones C, inicialmente vacía. Se crea el primer grupo C_1 con d_1 como único elemento. En el ciclo principal (línea 6), se analiza cada elemento d_i, $i \geq 2$, comparando su distancia con todos los elementos en los grupos creados hasta el momento y contenidos en C. d_i se agregará a aquel grupo donde exista d_j tal que la distancia de d_i a d_j es la menor de todas y dicha distancia es menor que el umbral dado. Si no se cumple esta condición, se crea un nuevo grupo que contiene solo a d_i y el proceso se repite hasta que todos los elementos en D se han analizado y asignado a un grupo.

Algorithm 2 Algoritmo de agrupación nnc

```
1: procedure NNC(D,μ)
2:     C = Set of empty clusters
3:     C_1 = {d_1}
4:     C = C ∪ {C_1}
5:     k = 1
6:     for i = 2 to | D | do
7:         Find d_j in some cluster C_m ∈ C where δ = dis(d_i, d_j) is the smallest
8:         if δ < μ then
9:             C_m = C_m ∪ {d_i}
10:        else
11:            k = k + 1
12:            C_k = {d_i}
13:            C = C ∪ {C_k}
14:        end if
15:    end for
16:    Exit with C
17: end procedure
```

2.1.2. Tarea de clasificación de datos

El proceso de clasificación consiste en, dado un conjunto de datos $D = \{d_1, d_2, \cdots, d_n\}$, asignar una etiqueta de clase c a cada d_i en función de las características de d_i y con base en las clases ya asignadas a un conjunto de datos previo (conjunto de entrenamiento) [47].

Existen dos tipos principales de clasificación: 1) clasificación binaria: que implica clasificar las instancias en dos clases, y 2) clasificación multiclase: que implica clasificar las instancias en más de dos clases. Los algoritmos de clasificación comúnmente utilizados son: árboles de decisión, k-nn y máquinas de vectores de soporte.

k-nn es un algoritmo popular de clasificación. La nueva instancia a clasificar se compara con las instancias ya clasificadas por el algoritmo de entrenamiento y entonces, se le asigna la clase o categoría c de la o las instancias (promedio) con las que exista mayor similitud (vecinos más cercanos [48]). Las instancias ya clasificadas en el conjunto de entrenamiento y las nuevas instancias a clasificar consisten de una serie de valores que las caracterizan (vector de características), denotado como vc. En el Algoritmo 3 se muestra el pseudocódigo de k-nn, el cual toma como entrada: 1) un conjunto de datos de entrenamiento (T) que contiene instancias representadas por sus vectores característicos y sus respectivas etiquetas de clase $T = \{[vc_1, c_1], [vc_2, c_2], ..., [vc_n, c_n]\}$ siendo que pueden existir i, j tal que $c_i = c_j$; 2) la instancia a clasificar (vc_x); y 3) un valor k, que representa la cantidad de vecinos más cercanos a vc_x en T que se consideraran para tomar una decisión. Como salida, el algoritmo regresa la clase predicha c_x para vc_x. Como primer paso, se crea una lista L_d (línea 2) que se utilizará para almacenar las distancias entre vc_x y cada instancia en T. Cada instancia $vc_i \in T$ se compara con vc_x usando una métrica de distancia, comúnmente la distancia euclidiana o de Manhattan, $\delta_i = distance(vc_i, vc_x)$ (línea 4). Esta distancia se almacena junto con la etiqueta de clase c_i en L_d (línea 5). Luego se seleccionan los k vecinos más cercanos utilizando una función ***sortSelectTop***$_k$ (línea 7), que ordena la lista L_d y selecciona los k elementos con las menores distancias. Estos vecinos cercanos son aquellos que comparten características similares con vc_x y son fundamentales para determinar c_x. Una vez identificados los k vecinos más cercanos, se realiza una votación, donde se cuentan las ocurrencias de cada

etiqueta de clase entre los vecinos cercanos. Esto se hace con la función *countVotes* (línea 8). Finalmente, se determina la clase predicha para la instancia de entrada, seleccionando la clase con el mayor número de votos, a través de la función *getMajorityClass* (línea 9), devolviendo la etiqueta de clase c_x para vc_x.

Algorithm 3 k-NN

```
1:  procedure KNN(T, vc_x, k)
2:      L_d = []
3:      for each vc_i ∈ T do
4:          δ = distance(vc_i, vc_x)
5:          L_d.append([δ, c_i])
6:      end for
7:      neighbors = sortSelectTop_k(L_d, k)
8:      class_votes[] = countVotes(neighbors)
9:      c_x = getMajorityClass(class_votes)
10:     Exit with c_x
11: end procedure
```

2.2. Minería de datos con preservación de privacidad (PPDM)

La preservación de la privacidad durante los procesos de minería de datos se ha convertido en un requisito para el intercambio de información confidencial en términos de análisis, validación y publicación de datos.

Para proteger la privacidad, los datos deben protegerse antes de compartirlos con entidades no autorizadas para accederlos directamente, ya sea porque son datos sensibles o debido a regulaciones en materia de protección de datos. Una posible solución es removiendo solo aquellas partes de los datos que sean críticos (por ejemplo, nombre y el número de pasaporte). Sin embargo, muchas veces esto no es posible, y los datos deben permanecer y protegerse, por lo que se hace necesario implementar medidas de preservación de la privacidad para evitar este tipo de violaciones. Para abordar este problema surgen los métodos de minería de datos con preservación de privacidad, los cuales tienen el objetivo de proteger los datos realizando ciertos cambios para enmascarar o borrar los datos sensibles originales. Estos enfoques buscan lograr un equilibrio entre utilidad y privacidad en los datos. Los PPDM utilizan la distribución de datos y la partición distribuida horizontal o verticalmente a través de múltiples entidades.

En la literatura se destacan principalmente cuatro enfoques para la construcción de PPDM: anonimización, perturbaciones, aleatorización y criptografía, los cuales se describen en las subsecciones siguientes.

2.2.1. Anonimización

Esta técnica permite ocultar datos confidenciales y privados de los usuarios mediante la generalización para brindar privacidad. Estos datos anónimos son los que se utilizan para hacer el proceso de minería [49]. Las técnicas de anonimización se pueden dividir en dos tipos:

1 **Métodos basados en reducción de datos:** Estas técnicas reducen la cantidad de información detallada en los datos, lo que dificulta identificar a los individuos a los que se refieren los datos, manteniendo al mismo tiempo la utilidad general del conjunto para el análisis. Estos procedimientos tienden a evitar la presencia de individuos reconocibles, únicos o raros [50].

2 **Métodos basados en perturbación de datos:** Dichos métodos logran la protección de datos desde una doble perspectiva. En primer lugar, si se modifican los datos, la reidentificación por medio de vinculación de registros o algoritmos de coincidencia es más difícil e incierta. En segundo lugar, incluso cuando un intruso puede volver a identificar una unidad y no puede estar seguro de que los datos revelados sean consistentes con los datos originales [51].

Existen diversos algoritmos de anonimización como *k-anonymity*, *l-diversity* y *t-closeness*, diseñados para evitar la identificación.

- ***k-anonymity*:** En el método *k-anonymity*, la granularidad de la representación de datos se reduce adecuadamente para que algunos datos dados se asignen a un mínimo k de otros registros en los datos [52]. Necesita técnicas como la generalización y la supresión para realizar este proceso de desidentificación de los datos [53]. La generalización consiste en sustituir los valores de los atributos por valores semánticamente consistentes pero menos precisos. K-anonymity mantiene la exactitud de los datos a nivel de registro, pero da como resultado información menos específica que puede afectar la precisión de los algoritmos de aprendizaje automático aplicados en el conjunto de datos k-anónimos [54]. La supresión se refiere a eliminar un cierto valor de atributo y reemplazar las ocurrencias del valor con un valor especial, lo que indica que se puede colocar cualquier valor en su lugar. La supresión puede reducir drásticamente la calidad de los datos si no se utiliza correctamente.

  Una desventaja del método *k-anonymity* es que si hay una uniformidad de valores sensibles dentro de un grupo, entonces esos valores pueden inferirse para los datos alterados [55]. Aunque *k-anonymity* puede resolver el problema del ataque de divulgación de identidad, no puede resolver el problema del ataque de divulgación de atributos.

- ***l-diversity*:** El método *l-diversity* se diseñó para gestionar el problema del ataque de divulgación de atributos mediante la aplicación de una variedad intragrupo de valores sensibles para proporcionar anonimización [56]. El objetivo es hacer que sea lo suficientemente difícil para los adversarios usar combinaciones de atributos de datos para reconocer exactamente registros individuales. Una característica de este método es que trata todos los valores de un atributo dado de manera similar, independientemente de su distribución en los datos [57]. Según el método *l-diversity*, una clase de equivalencia debe tener l valores bien representados para los atributos sensibles [54]. *L-diversity* solo garantiza la diversidad de características sensibles dentro de cada grupo, pero no se resuelve el problema de que diferentes valores pueden pertenecer a la misma categoría. En otras palabras, no es resistente a los ataques basados en la similitud semántica entre valores.

- ***t-closeness:*** Con el fin de equilibrar las similitudes semánticas de los atributos dentro de cada grupo, se ha propuesto resolver las limitaciones del enfoque de *l-diversity* garantizando la proximidad t entre sí [58]. En consecuencia, en el método *t-closeness*, la distancia de la distribución de los atributos sensibles en cualquier clase de equivalencia a la distribución de los atributos en toda la tabla no excederá un valor umbral (t) [56]. Si bien el enfoque *t-closeness* brinda protección contra la divulgación de atributos, no puede proteger contra la divulgación de identidades. Además, limita la utilidad de la información divulgada; sin embargo, al establecer el umbral t en las aplicaciones, puede intercambiar beneficios y privacidad [59]. En la protección de la privacidad, los métodos de *t-closeness* y *k-anonymity* se utilizan juntos para proteger contra ataques a la divulgación y calidad de la identidad.

2.2.2. Perturbación

Los métodos PPDM basados en perturbación se encargan de transformar los datos mediante el uso de técnicas de distorsión de datos como la adición de ruido, la rotación y la proyección [55]. La perturbación de datos se considera una técnica relativamente fácil y efectiva para proteger los datos. Este método es aplicable a las entradas de datos numéricos, alterando los conjuntos de datos con un valor y una operación específicos [60].

El enfoque de perturbación de datos se clasifica en dos tipos:

- **Distribución de probabilidad:** Reemplaza los datos con otra muestra de la misma distribución. Dentro de este tipo podemos encontrar dos métodos [61]. El primero es el intercambio de datos, en el cual, la base de datos original se reemplaza con una base de datos generada aleatoriamente que tiene aproximadamente la misma distribución de probabilidad que la base de datos original. Siempre que se agregue una nueva entidad o se elimine una entidad actual, la relación entre esta entidad y el resto de la base de datos debe tenerse en cuenta al calcular una nueva perturbación [62]. El segundo se llama método de distribución de probabilidad. Este se compone de tres pasos: 1) Identificar la función de densidad subyacente de los valores de los atributos y estimar los parámetros de esta función, 2) Generar una serie de datos de muestra a partir de la función de densidad estimada del atributo confidencial. La nueva muestra debe ser del mismo tamaño que la de la base de datos. 3) Sustituir los datos generados del atributo confidencial por los datos originales en el mismo orden de clasificación [63].

- **Distorsión de valor:** Perturba los datos mediante ruido multiplicativo o aditivo u otros procesos aleatorios. Se considera más efectivo que el tipo anterior de perturbación. Este enfoque construye clasificadores de árbol de decisión donde a cada elemento se le asigna ruido aleatorio con una distribución gaussiana [64]. Este tipo de perturbación puede agregar una suma a todos los valores numéricos en su base de datos o utilizar una cifra determinada como base de su operación; dividiendo todos los valores numéricos por esta. Es importante seleccionar con cuidado la base utilizada para modificar los valores originales, ya que, si la base es demasiado pequeña, los datos no se anonimizarán lo suficiente y, si es demasiado grande, es posible que los datos no se reconozcan ni pueda extraerse su valor [65].

2.2.3. Aleatorización

La técnica de aleatorización de datos utiliza métodos de distorsión de datos para crear representaciones privadas de los registros. Permite modificar los datos según patrones aleatorios predefinidos. El algoritmo de aleatorización se elige de modo que las propiedades agregadas de los datos se puedan recuperar con suficiente precisión, mientras que las entradas individuales se distorsionan significativamente [66].

En general, se espera que la varianza del ruido agregado sea lo suficientemente grande como para que los valores del registro original no puedan ser fácilmente inferidos a partir de los datos distorsionados, lo que impide la recuperación de los registros originales, pero permite recuperar la distribución subyacente de los mismos.

En otros métodos, como *k-anonymity*, explicado anteriormente, el comportamiento general de los registros se aprovecha en el proceso de anonimización. Esto es muy útil, ya que significa que la aleatorización se puede realizar en el momento de la recolección de datos. Por lo tanto, no se requiere un servidor confiable (como en *k-anonymity*) para realizar las transformaciones en los registros. Esta es una ventaja clave de los métodos de aleatorización. Otra propiedad clave es que los registros originales no se utilizan después de la

transformación. Más bien, los algoritmos de minería de datos usan distribuciones agregadas de los datos para realizar el proceso de minería [64].

Se pueden obtener dos tipos de aleatorizaciones, las cuales son:

- **Perturbación aditiva**: Permite agregar ruido aleatorio, en este caso, se agrega ruido aleatorio a los registros de datos. Las distribuciones generales de datos se pueden recuperar de los registros aleatorios.
- **Perturbación multiplicativa**: Utiliza técnicas de proyección aleatoria o rotación aleatoria para perturbar los registros. Estas técnicas aplican transformaciones lineales a los datos, lo que cambia la estructura de los datos de manera más compleja que la simple adición de ruido aleatorio.

2.2.4. Cifrado

En criptografía, un cifrador es un objeto que realiza dos operaciones: *Enc* y *Dec*. Mediante la operación de cifrado *Enc*, transforma un texto legible M en algo ilegible CT, usando una llave o clave K_{enc}. Esto se denota como $CT \leftarrow Enc(K_{enc}, M)$. Mediante *Dec*, transforma CT en M nuevamente, usando una llave K_{dec}, que se denota como $M \leftarrow Dec(K_{dec}, CT)$. Un cifrador debe cumplir que:

$$Dec(K_{dec}, Enc(K_{enc}, M) = M \tag{1}$$

Los cifradores han sido los objetos más usados de la criptografía para garantizar el servicio de confidencialidad: al convertirse CT en algo ilegible, M se convierte privado y de acceso restringido solo para quien posea K_{dec} y, por tanto, tenga la capacidad de recuperar M a partir de CT mediante *Dec*. *Enc* debe ser tal que, en la práctica, es imposible obtener M a partir de CT sin el conocimiento de K_{dec}. Debido a ello, las claves y su generación juegan un papel crucial en los cifradores. Los algoritmos de generación de llaves deben ser tal que se garantice dicha seguridad, la cual generalmente se caracteriza por la dificultad para resolver un problema matemático. Así, mientras los datos se encuentren cifrados, permanecerán seguros y privados. Sí $K_{enc} = K_{dec}$, el cifrador se denomina simétrico, de lo contrario, el cifrador es asimétrico.

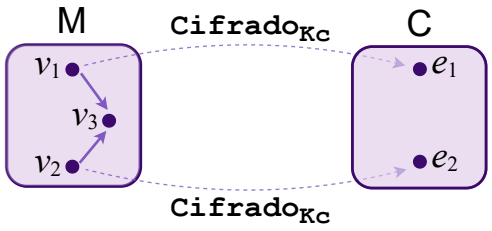

Figura 3. Concepto de cifrado, mapeos entre el espacio de mensajes $\mathcal{M}$ y espacio de textos cifrados $\mathcal{C}$.

M pertenece a lo que es conocido como el espacio de mensajes ($\mathcal{M}$) mientras que CT pertenece al espacio de textos cifrados ($\mathcal{C}$). Generalmente, tanto $\mathcal{M}$ como $\mathcal{C}$ son estructuras algebraicas. De esta forma, *Enc* y *Dec* se definen como mapeos de $\mathcal{M}$ a $\mathcal{C}$ y viceversa respectivamente, definidos por las correspondientes llaves criptográficas (ver Figura 3). Bajo el contexto previo, considere los operadores binarios ∇ y $\circ$ definidos sobre $\mathcal{M}$ y $\mathcal{C}$ respectivamente. Un esquema de cifrado se dice que es homomórfico si se cumple:

$$Dec\left(K_{dec}, Enc(K_{enc}, v_1) \circ Enc(K_{enc}, v_2)\right) = v_1 \nabla v_2 \quad (2)$$

Dicho de otra forma: si en $\mathcal{C}$ existen dos elementos $e_1 = Enc(K_{enc}, v_1), e_2 = Enc(K_{enc}, v_2)$ tales que $e_3 = e_1 \circ e_2$, entonces existe v_3 en $\mathcal{M}$ tal que $Dec(K_{dec}, e_3) = v_1 \nabla v_2 = v_3$.

En matemáticas, homomórfico describe la transformación de una estructura algebraica en otra, conservando las propiedades entre los elementos de ambos conjuntos. Así, los datos en un esquema de cifrado homomórfico conservan la misma estructura, independientemente de si las operaciones se realizan en el espacio de los datos cifrados o descifrados. De manera general, el cifrado homomórfico hace posible operar los datos en claro a través de sus versiones cifradas, manteniendo así su privacidad y seguridad.

En la literatura se han propuesto tres tipos principales de cifrado homomórfico [67]:

- Esquemas parcialmente homomórficos: permiten trabajar con ciertas funciones matemáticas, pero generalmente estableciendo una relación homomórfica entre un solo operador aritmético.

- Esquemas algo (*somewhat*) homomórficos: admiten operaciones limitadas que solo se pueden realizar un número determinado de veces.

- Esquemas totalmente homomórficos: permiten distintas funciones matemáticas, así como establecer relaciones homomórficas sobre más de un operador aritmético. Además de no tener un límite de veces en que se pueden aplicar. Se pueden dividir en 3 tipos según el cálculo a realizar. El primer tipo modela los cálculos como circuitos booleanos (bits). El segundo tipo modela el cálculo como aritmética modular (aritmética de reloj). El tercer tipo modela los cálculos como aritmética de punto flotante.

Existen otros tipos de cifrado especial que permiten realizar operaciones con los textos cifrados como si los realizarán con los textos en claro. Un ejemplo es el Cifrado con Preservación de Orden (OPE, por sus siglas en inglés), el cual permite realizar comparaciones de datos cifrados, es decir, garantiza que el orden numérico de los textos en claro persista en los textos cifrados generados. Si $e_1 = Enc(K_{enc}, v_1), e_2 = Enc(K_{enc}, v_2)$ y $v_1 < v_2$, entonces $e_1 < e_2$. Los esquemas OPE, en su mayoría, son esquemas de cifrado deterministas (mapeo uno a uno). Un adversario que conoce el espacio del mensaje o recopila datos estadísticos sobre el espacio del mensaje (distribución y frecuencia de los datos) es capaz de crear una correlación entre los valores reales y los cifrados.

El enfoque de solución PPDM basado en cifrado (homomórfico u OPE) ha demostrado ser efectivo para preservar la privacidad de los datos, al usar $\mathcal{CT}$ (ilegible) en lugar de $\mathcal{M}$, al tiempo que también preserva su utilidad, puesto que CT permite obtener los mismos resultados en el contexto de minería de datos que si se usara $\mathcal{M}$ [68]. Al ser $\mathcal{M}$ y $\mathcal{CT}$ estructuras algebraicas, los datos categóricos deben transformarse en elementos de $\mathcal{M}$ para poder cifrarlos. Los valores descifrados deber convertirse nuevamente en datos categóricos para recuperar la información original.

Aunque efectivas, las técnicas basadas en criptografía resultan ser muy costosas computacionalmente, pues los algoritmos de minería de datos deben modificarse para operar sobre textos cifrados $\mathcal{CT}$, los cuales siguen codificaciones distintas a $\mathcal{M}$, lo que afecta la eficiencia de la técnica PPDM y por consecuencia el despliegue eficiente de DMaaS.

2.3. Minería de datos como servicio (DMaaS)

Como se definió anteriormente, la minería de datos es un método que combina el análisis de grandes cantidades de datos para detectar relaciones, patrones, irregularidades y/o tendencias. Empleando diferentes

técnicas de análisis de datos, las organizaciones pueden realizar predicciones que les permita tomar mejores decisiones.

Por otro lado, el modelo *As a Service*, es un modelo emergente en el que se aprovecha el cómputo en la nube para subcontratar almacenamiento, procesamiento u otros recursos de infraestructura computacional a un tercero (tecnología basada en suscripción). Este tipo de tecnologías está diseñada para reducir el gasto inicial en tecnologías de información, para mejorar la flexibilidad en el acceso a recursos de cómputo y tecnologías asociadas actualizadas.

Entre los modelos como servicio más populares se encuentran PaaS (plataformas como servicio), SaaS (software como servicio), IaaS (infraestructura como servicio), y recientemente se agrega DMaaS, que permite ofrecer la minería de datos como un servicio.

Para que DMaaS sea viable, es necesario construir un sistema distribuido, que contempla arquitecturas para el desarrollo de los componentes de software, patrones de procesamiento para realizar de forma eficiente el procesamiento de los datos y la comunicación entre los componentes (nodos) que sirve para sincronizar y/o paralelizar tareas mediante la coordinación de estos a través de los modelos de comunicación.

2.3.1. Arquitecturas de software

Una arquitectura de software permite organizar de manera lógica los componentes de software y su interacción con otras estructuras. Se refiere a la estructura básica de cualquier sistema de software e incorpora cualquier aspecto que hace que un sistema funcione y se comporte como debería [69].

Algunos estilos de arquitectura se prestan naturalmente a sistemas altamente escalables, mientras que otros estilos se prestan a aplicaciones que permiten a los desarrolladores responder rápidamente a los cambios. Por ello, es necesario conocer las características, fortalezas y debilidades de cada estilo de arquitectura para elegir el que satisfaga sus necesidades y objetivos específicos. Algunas de las arquitecturas más conocidas y sus características, se presentan a continuación:

- **Arquitectura en capas:** Esta es la arquitectura más común, también se conoce como arquitectura de n niveles (ver Figura 4). Este tipo de arquitectura proporciona un enfoque modular, permitiendo separar los componentes en unidades [69]. Componentes similares generalmente se colocan en las mismas capas. Sin embargo, cada capa es diferente y contribuye a una parte diferente del sistema general [70]. Cada capa solo se comunica con su capa adyacente enviando solicitudes y obteniendo respuestas [71]. Una solicitud va de arriba hacia abajo y la respuesta va de abajo hacia arriba. La ventaja de esta arquitectura radica en que cada capa se puede modificar de forma independiente sin afectar al resto del sistema [72].

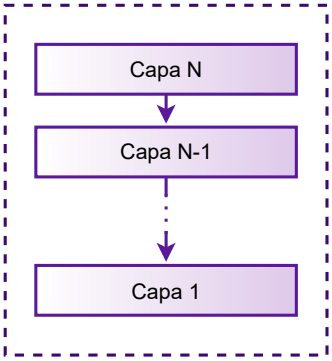

Figura 4. Arquitectura basada en capas.

La mayoría de las arquitecturas en capas constan de tres capas estándar: una capa de interacción con el usuario (acceso), una capa para el procesamiento y una capa que se ocupa del manejo de los datos [73].

Ventajas

- Es simple y fácil de aprender e implementar.
- Existe una mínima dependencia entre capas, ya que la función de cada capa está separada de las otras capas.
- Hacer pruebas es más fácil debido a que los componentes están separados, por lo que se puede probar cada uno de manera independiente.

Desventajas

- La escalabilidad es difícil debido a que la estructura no permite el crecimiento.
- Pueden ser difíciles de mantener debido a que una sola capa puede afectar todo el sistema, ya que funciona como una unidad.
- Existe una interdependencia entre las capas, ya que cada capa depende de la capa superior para recibir los datos.
- No es posible hacer procesamiento en paralelo.
- Cuanto más grande es, más recursos requiere para que las solicitudes pasen por varias capas y, por lo tanto, causarán problemas de rendimiento.

- **Arquitectura basada en objetos:** Esta arquitectura (Figura 5) concibe a un sistema como una colección de entidades llamadas objetos. A diferencia de la arquitectura en capas, la arquitectura basada en objetos no tiene que seguir ningún paso en una secuencia [74]. Cada componente es un objeto, y todos los objetos pueden interactuar a través de una interfaz. Bajo la arquitectura basada en objetos, tales interacciones entre componentes pueden ocurrir a través de una llamada directa [75]. Los objetos se conectan entre sí mediante el mecanismo de llamada a procedimiento remoto (RPC) o el mecanismo de invocación a método remoto (RMI) [76].

  Los servicios web y la API REST son ejemplos de arquitectura basada en objetos [77].

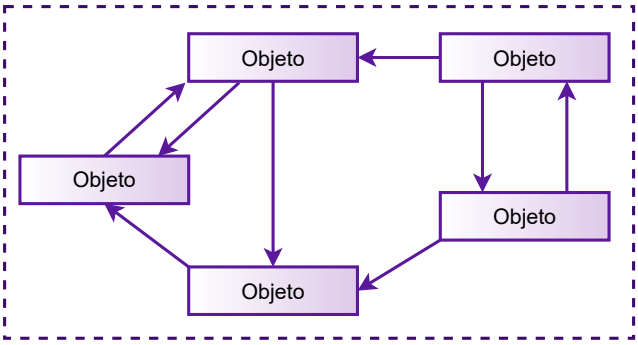

Figura 5. Arquitectura basada en objetos.

Ventajas

- Fácil de mantener, ya que permite mejorar la calidad del sistema debido a la reutilización de componentes.
- Permite la reutilización a través del polimorfismo y la abstracción.
- Capacidad para gestionar errores durante la ejecución.
- Es posible añadir nuevas funciones sin afectar al sistema completo.
- Mejora la capacidad de pruebas a través de la encapsulación.
- Esta arquitectura reduce el tiempo y el costo de desarrollo.

Desventajas

- Esta arquitectura tiene dificultades para determinar todas las clases y objetos necesarios para un sistema.
- Esta arquitectura no conduce a una reutilización exitosa a gran escala sin un procedimiento de reutilización explícito.

- **Arquitectura centrada en datos:** En esta arquitectura los datos están centralizados y son accedidos por otros componentes. La arquitectura se enfoca en el flujo de información a través de la organización y luego ajusta los procesos para agilizar el flujo [74].

  Este es un tipo de arquitectura (Figura 6) donde se asume que todos los datos están localizados en un solo lugar, un espacio de datos compartido. Todos los componentes están conectados a este espacio de datos y siguen el modelo de comunicación de *Publisher-Subscriber* (explicado en la sección 2.3.3). El productor produce elementos para el almacén de datos común y el consumidor puede solicitarle datos [78]. Luego, los datos requeridos se envían a los componentes. Este enfoque requiere una comprensión profunda de los datos [79]: de dónde provienen, quién los posee, quién los usa y cómo, cuánto tiempo deben conservarse, cuándo deben archivarse, qué tan confidencial es, etc.

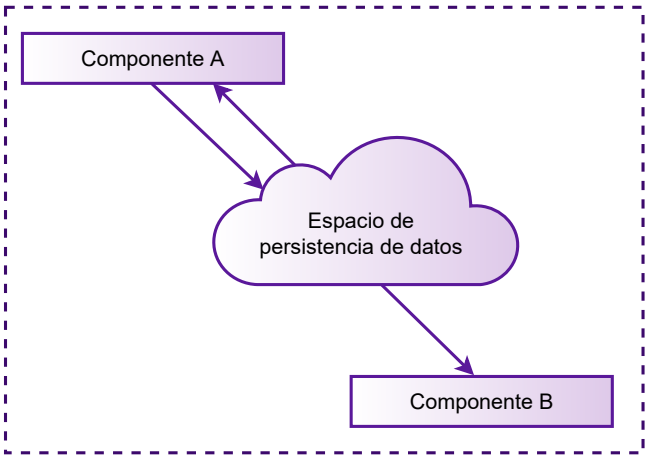

Figura 6. Arquitectura centrada en datos.

El objetivo principal de este estilo es lograr la integridad de los datos. La arquitectura centrada en datos consta de diferentes componentes que se comunican a través de repositorios de datos compartidos. Los componentes acceden a una estructura de datos compartida y son relativamente independientes, ya que interactúan solo a través del almacén de datos.

Ventajas

- Proporciona integridad de datos, copias de seguridad y de restauración.
- Reduce la sobrecarga de datos transitorios entre componentes de software.
- Tiene una forma eficiente de almacenar una gran cantidad de datos.
- Trabaja con una gestión centralizada que consiste en respaldo, seguridad y control de concurrencia.

Desventajas

- La evolución de los datos es difícil y costosa.
- Tiene alta dependencia entre la estructura de los datos, el almacén de datos y sus componentes de software.

- **Arquitectura basada en eventos:**

  En esta arquitectura (Figura 7) toda la comunicación se realiza a través de eventos. Cuando ocurre un evento, el sistema recibe la notificación y la distribuye a las entidades involucradas [79]. Los eventos se entregan al componente requerido cuando sea necesario [69]. Los componentes están acoplados de forma flexible. Esto significa que es fácil agregarlos, eliminarlos y modificarlos. Los componentes heterogéneos se comunican con el bus, independientemente de sus protocolos de comunicación [80].

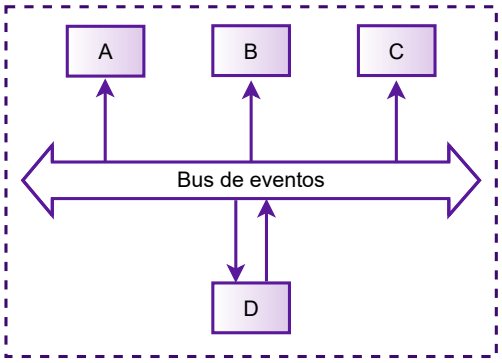

Figura 7. Arquitectura basada en eventos.

Las arquitecturas basadas en eventos tienen tres componentes clave: productor, consumidor e intermediario. El intermediario puede ser opcional, especialmente cuando tiene un solo productor y un solo consumidor que están en comunicación directa entre sí y el productor solo envía los eventos al consumidor [81]. Los servicios del productor y los servicios del consumidor se desacoplan, lo que les permite escalarse, actualizarse e implementarse de manera independiente [82].

2.3.2. Patrones de procesamiento

Los patrones de procesamiento pueden definirse como un conjunto de actividades, acciones, productos de trabajo y comportamientos similares seguidos en un ciclo de vida de desarrollo de software.

Estos patrones permiten dividir un problema en múltiples componentes de procesamiento que operan simultáneamente, con el objetivo de mejorar la eficiencia.

Los patrones de procesamiento más comunes se describen a continuación:

- **Manager - worker**

  Se basa en un enfoque simple que permite que las aplicaciones realicen el procesamiento simultáneo en múltiples máquinas o procesos a través de un manejador (*manager*) y múltiples trabajadores (*workers*) [83, 84]. En la Figura 8 se muestran los elementos que componen este patrón: 1) *Manager*: es el encargado de gestionar los trabajadores, permite dividir el trabajo entre ellos, iniciar su ejecución y obtener los resultados de cada uno de ellos. 2) *Worker*: es el encargado de procesar una tarea, implementar el cómputo en forma de un conjunto de operaciones requeridas y realizar el cómputo.

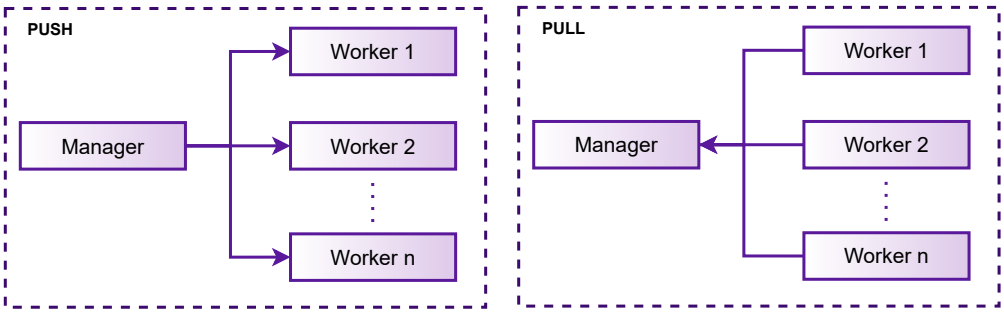

Figura 8. Patrón de procesamiento *Manager - worker*.

Hay dos patrones básicos para *manager-worker* con *workers* direccionables. En el patrón Push (izquierda de la Figura 8), la iniciativa la tiene el manager. En el patrón Pull (derecha de la Figura 8), la iniciativa la tienen los *workers* [85]. Los patrones básicos se pueden variar para acomodar diferentes tipos de problemas y diferentes infraestructuras para ejecutar a los trabajadores [86].

En este patrón, la misma operación se aplica simultáneamente a diferentes piezas de datos por componentes de trabajo [87]. Las operaciones en cada *worker* son independientes de las operaciones en otros *workers*. El manager es un administrador central que distribuye los datos entre los trabajadores a pedido. Por tanto, la solución se presenta como una red centralizada [88].

- **Divide and conquer**

  Este patrón está basado en la resolución recursiva de un problema, dividiéndolo en dos o más subproblemas similares o de un mismo tipo [83]. El resultado de cada subproblema no se almacena para referencia futura, mientras que, en un enfoque dinámico, el resultado de cada subproblema se almacena para referencia futura [89].

  Se utiliza el enfoque *divide and conquer* cuando el mismo subproblema no se requiere resolver muchas veces. A diferencia de los enfoques dinámicos, que se utiliza cuando el resultado de un subproblema se va a utilizar varias veces en el futuro [90].

  En la Figura 9 se observa el patrón *divide and conquer* de manera gráfica. Este modelo se divide en tres etapas [91]: 1) *Divide*: Implica dividir el problema en subproblemas más pequeños usando recursividad. 2) *Conquer*: Resuelve los subproblemas usando recursividad hasta lograr resolverlo. 3) *Combine*: Permite combinar las soluciones a los subproblemas para obtener una solución general al problema inicial.

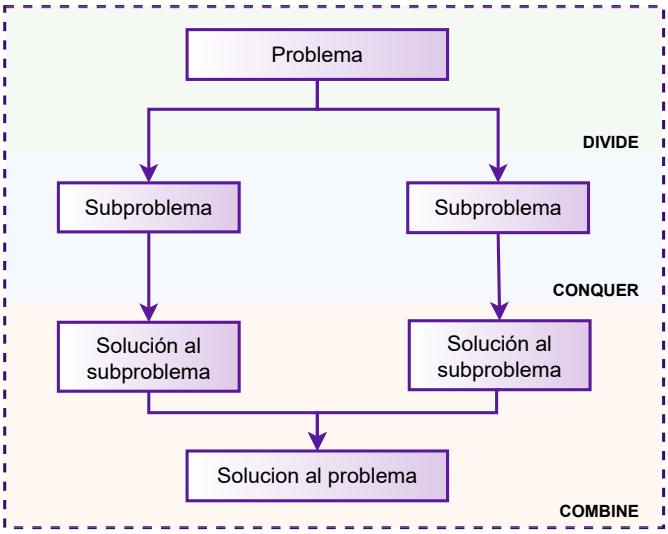

Figura 9. Patrón de procesamiento *Divide and conquer*.

Esta técnica es la base de los algoritmos eficientes para casi cualquier tipo de problema como, por ejemplo, algoritmos de ordenamiento (quicksort, mergesort, entre otros), multiplicar números grandes (Karatsuba), análisis sintácticos (análisis sintáctico *top-down*) y la transformada discreta de Fourier [92].

- **Pipes and filters**

  Este patrón se caracteriza por sucesivas transformaciones de flujos de datos. Permite la descomposición de un componente monolítico en elementos independientes más pequeños que se pueden reutilizar [93].

  En la Figura 10 se puede ver el funcionamiento de este patrón. En este, los datos fluyen en una misma dirección, el proceso se inicia con un conjunto de datos, llega a los puertos de entrada de un filtro, donde el procesamiento se realiza en el componente y luego pasa a través de una tubería al siguiente filtro y finalmente se obtiene un resultado.

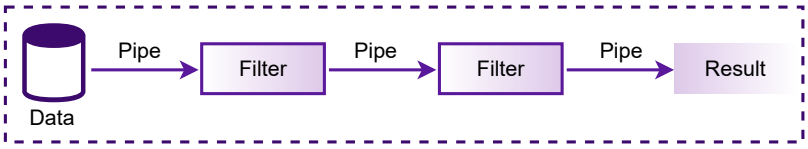

Figura 10. Patrón de procesamiento *pipes and filters*.

  Cada uno de los filtros o componentes realiza una única tarea. Para lograr un formato estándar de los datos que cada componente recibe y envía, los filtros se pueden combinar, al hacerlo, evita la duplicación de código y facilita el reemplazo de componentes, o la integración de componentes adicionales, si cambian los requisitos de procesamiento [94].

  El tiempo que se tarda en procesar una sola solicitud depende de la velocidad de los filtros más lentos del flujo. Uno o más filtros pueden ser cuellos de botella, especialmente si aparece una gran cantidad de solicitudes en un flujo de una fuente de datos en particular [95]. Una ventaja clave de la estructura de canalización es que brinda oportunidades para ejecutar instancias paralelas de filtros lentos, lo que permite que el sistema distribuya la carga y mejore el rendimiento [96].

  Este patrón permite mejorar el rendimiento, la escalabilidad y la reutilización al permitir que los elementos de tareas que realizan el procesamiento se implementen y escalen de forma independiente [97].

- **Fork - join**

  El patrón *Fork - join* permite ejecutar programas en paralelo, de esta manera, la ejecución se divide en puntos designados del programa para unirse en un punto posterior y reanudar la secuencia, tal como se ve en la Figura 11. Las secciones paralelas pueden bifurcarse recursivamente hasta que se alcance una determinada granularidad de la tarea [98].

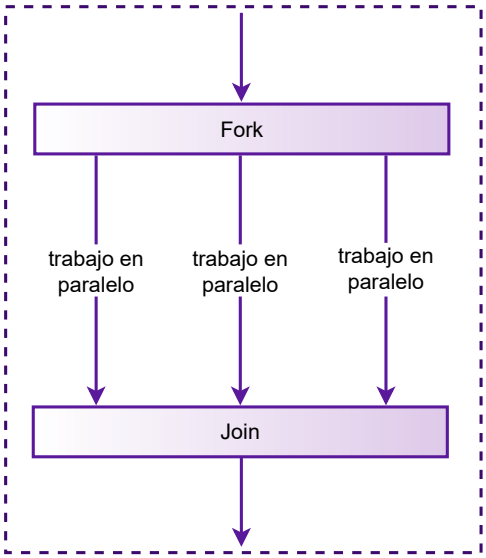

Figura 11. Patrón de procesamiento *Fork - Join*.

Al anidar los cálculos de *fork - join* recursivamente, se obtiene una versión paralela del paradigma *divide and conquer*. El flujo de control se bifurca (*fork*) en múltiples flujos que se unen (*join*) más tarde. Cada uno de los flujos es independiente y no están obligados a realizar cálculos similares [99].

Una de las principales cosas a considerar cuando se implementa un algoritmo que utiliza el paralelismo de *fork - join* es elegir un umbral, este determina si una tarea ejecutará un cálculo secuencial en lugar de subtareas paralelas [100]. Si este umbral es demasiado grande, es posible que el programa no cree suficientes tareas para aprovechar al máximo los procesadores o núcleos disponibles. Si es demasiado pequeño, la sobrecarga de creación y administración de tareas podría volverse significativa [101]. De manera general, para encontrar el umbral apropiado es necesario realizar experimentación previa.

2.3.3. Modelos de comunicación

Los modelos de comunicación son representaciones simplificadas de un proceso de comunicación entre componentes, que de manera general, tienden a utilizar un medio para transmitir mensajes [102].

En la Figura 12 se observan los componentes básicos para que exista la comunicación entre dos entidades, emisor y receptor. El emisor es el que se encarga de enviar los mensajes a un receptor a través de un canal. Cada mensaje debe ser codificado antes de llegar al canal, y posteriormente se decodifica para que el receptor lo pueda interpretar. Para devolver una respuesta, es necesario repetir estos pasos, codificar y decodificar. Es necesario recalcar que siempre existirá un tipo de ruido que puede deteriorar la transmisión de los mensajes.

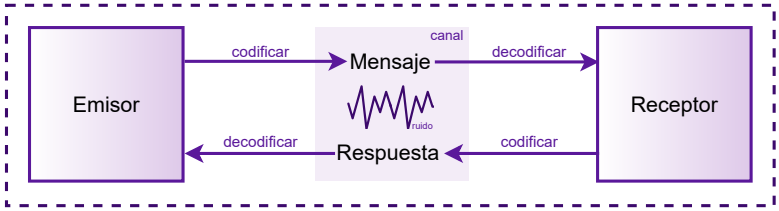

Figura 12. Modelo de comunicación.

Los modelos de comunicación más relevantes se mencionan a continuación:

- **Cliente - Servidor**

  Este modelo es el que se utiliza con mayor frecuencia en los sistemas distribuidos. Se refiere a un sistema que aloja, entrega y administra la mayoría de los recursos y servicios que solicita el cliente [103]. En este modelo, todas las solicitudes y servicios se entregan a través de una red, y también se lo conoce como modelo de computación en red o red de servidor-cliente [104]. Es una aplicación de red que divide tareas y cargas de trabajo entre clientes y servidores que residen en el mismo sistema o están vinculados por una red [105].

  La Figura 13 ilustra la estructura de este modelo de comunicación, en esta se observan dos componentes, cliente y servidor. De manera general, el cliente envía una solicitud al servidor y este devuelve una respuesta. Es particular, los clientes interactúan con los servidores en equipos host potencialmente separados para acceder a los recursos compartidos que administran. Los servidores pueden, a su vez, ser clientes de otros servidores [106]. Por ejemplo, un servidor web suele ser un cliente de un servidor de archivos local que administra los archivos en los que se almacenan las páginas web. Los servidores web y la mayoría de los demás servicios de Internet son clientes del servicio DNS, que traduce los nombres de dominio de Internet en direcciones de red [107].

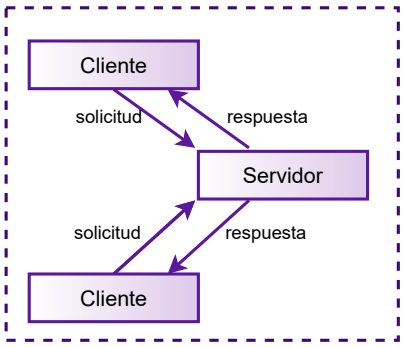

Figura 13. Modelo de comunicación *Cliente - Servidor*.

  El modelo cliente-servidor es un sistema centralizado que mantiene todos los datos y sus controles en un solo lugar, además, permite cambiar las capacidades de cada cliente o servidor por separado, lo que aporta un alto nivel de escalabilidad, organización y eficiencia [108].

- **Peer to peer**

El modelo *peer to peer*, también conocido como modelo P2P, consiste en una red descentralizada de nodos, es decir, nodos que son clientes y servidores a la vez [73]. Estos nodos tienen las mismas responsabilidades y permisos para trabajar con los datos. Las redes P2P distribuyen la carga de trabajo entre nodos, y todos los nodos contribuyen y consumen recursos dentro de la red sin necesidad de un servidor centralizado [107].

En la Figura 14 se observa la estructura de este modelo. En esta se cuenta con 5 nodos y cada uno de ellos se conecta con todos los demás nodos en la red. En su forma más pura, la arquitectura P2P está completamente descentralizada. Sin embargo, en la aplicación, a veces hay un servidor de seguimiento central superpuesto a la red P2P para ayudar a los nodos a encontrarse y administrar la red [104].

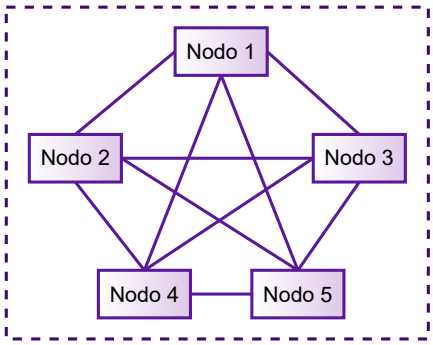

Figura 14. Modelo de comunicación *Peer to Peer*.

Existen tres tipos de P2P [109]: 1) Redes no estructuradas: En este tipo de redes los nodos no cuentan con un orden en particular, por lo que la comunicación entre nodos es aleatoria. Un ejemplo de este tipo de redes son las plataformas sociales donde los usuarios pueden unirse o abandonar la red regularmente. 2) Redes estructuradas: En este tipo de redes los nodos tienen una forma de interactuar entre sí de manera organizada. Esto permite que los usuarios encuentren y usen archivos de manera más eficiente en lugar de buscar al azar. Y, 3) Redes híbridas: Esta combinan la arquitectura P2P con el modelo *cliente-servidor*, lo que permite un servidor central con capacidades P2P [110].

- **Publisher-Subscriber**

  El modelo *Publisher-Subscriber* o *Pub-Sub* se utiliza en sistemas distribuidos como un modelo de comunicación para transmitir mensajes asíncronos y escalables que permiten separar los servicios que producen mensajes de los servicios que procesan mensajes [86]. Este modelo está basado en modelos de diseño explicados anteriormente, además utiliza colas de mensajes [111], y de eventos [85].

  En la Figura 15 se observa el modelo de comunicación *Publisher-Subscriber*. La clave de este modelo es que permite el paso de mensajes entre diferentes componentes del sistema sin que los componentes tengan conocimiento de los demás componentes, es decir, están desacoplados. Cuenta con dos tipos de componentes: 1) *Publisher*, que es el que crea y envía los mensajes, y 2) *Subscriber*, son los que reciben y consumen los mensajes del componente *publisher*. Además, se hace uso de un intermediario, que es el encargado de agrupar los mensajes en entidades conocidas como canales o *channels* [112].

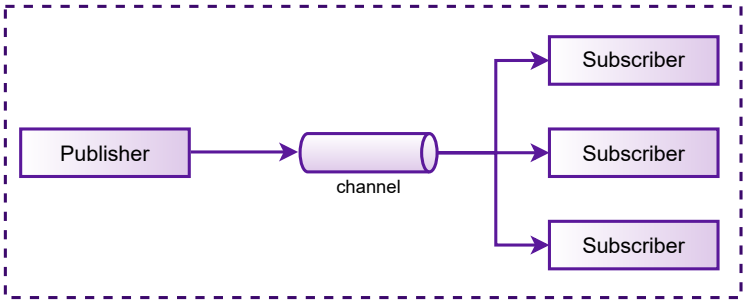

Figura 15. Modelo de comunicación *Publisher-Subscriber*.

Pub-Sub se usa para las canalizaciones de integración de datos y estadísticas de transmisión a fin de transferir y distribuir datos. Es tan eficaz como un *middleware* orientado a la mensajería para la integración del servicio o como una cola con el objetivo de paralelizar tareas [113].

2.3.4. Balanceadores de carga

En el cómputo en la nube, los balanceadores de carga permiten distribuir el tráfico y las cargas de trabajo de manera que ningún servidor o máquina tenga carga insuficiente, sobrecarga o inactividad [114].

Conseguir un equilibrio en la carga permite optimizar ciertos parámetros, como lo son: el tiempo de ejecución, el tiempo de respuesta y la estabilidad del sistema para mejorar el rendimiento general [115]. El objetivo general del balanceo de carga es minimizar el tiempo de respuesta para los usuarios de la aplicación y maximizar los recursos de la organización.

Existen diferentes algoritmos de balanceo de carga, los principales se describen a continuación:

- **Round Robin:**

  Este es uno de los métodos más simples para distribuir solicitudes en un grupo de servidores. Con este método, las peticiones se enrutan a los servidores disponibles de forma cíclica [116]. Primero, cada petición es asignada a un servidor, al finalizar la cantidad de servidores disponible, el balanceador regresa al principio de la lista y repite nuevamente el balanceo.

  El principal beneficio del balanceo de carga Round Robin es que es extremadamente simple de implementar [117]. Sin embargo, no siempre da como resultado la distribución más precisa o eficiente del tráfico, porque muchos balanceadores asumen que todos los servidores son iguales: en funcionamiento, manejando la misma carga y/o con la misma capacidad informática y de almacenamiento.

- **Pseudoaleatorio:**

  Tal como indica su nombre, este algoritmo permite emparejar clientes y servidores de forma aleatoria, es decir, utilizando un generador de números aleatorios subyacente [118]. En los casos en que el balanceador de carga recibe una gran cantidad de solicitudes, un algoritmo aleatorio podrá distribuir las solicitudes de manera uniforme a los nodos. Entonces, al igual que Round Robin, el algoritmo aleatorio es suficiente para clústeres que consisten en nodos con configuraciones similares (CPU, RAM, etc.).

- **Two choices:**

Este algoritmo es eficiente de implementar. La idea es no tener que comparar todas las colas para elegir la mejor opción, en cambio, solo se comparan dos [116].

El proceso de selección para este algoritmo es el siguiente: se eligen dos servidores de la lista de manera aleatoria, después se selecciona el servidor con menos carga (de acuerdo al número de peticiones actual), cuando la operación se completa, se reduce el número de peticiones en curso para el servidor.

El propósito de este, es ahorrarle al balanceador el costo de tener que verificar todos los servidores y al mismo tiempo hacer la mejor elección que haberlo hecho de manera aleatoria [119].

Evita el comportamiento de manada no deseado mediante el simple enfoque de evitar la peor cola y distribuir el tráfico con cierto grado de aleatoriedad.

3. Trabajo relacionado

Con base en la problemática abordada en este trabajo, se realizó una revisión sistemática de la literatura sobre técnicas PPDM basadas en criptografía. Se encontró que las tareas de minería de datos (PPDM) más utilizadas son agrupamiento y clasificación, las cuales se muestran en las Tablas 1 y 2, respectivamente, donde se describen características de algunos de los trabajos más relevantes y recientes en dicha área.

Autor	Año	Modelo de minería	DMaaS	Esquema	PPE	N. Seguridad
Wang *et al.* [120]	2022	DBSCAN	No	BGV	-	-
Yongkai *et al.* [121]	2021	k-means	No	Paillier	-	-
Wen-jie *et al.* [121]	2021	k-means	No	LWE-based	-	-
Yongkai *et al.* [122]	2021	k-means	No	WAP	OPE	-
Nidhya *et al.* [123]	2021	k-means	No	Paillier	DPE	-
Wei Wu *et al.* [124]	2021	k-means	No	YASHE	DPE	128 bits
Brandao *et al.* [125]	2021	k-means	No	CKKS	DPE	-
Gong *et al.* [126]	2021	k-means	No	QHE	-	-
Golda *et al.* [127]	2021	fuzzy c-means	No	-	-	-
Ila C *et al.* [128]	2021	-	No	-	-	-
Ying Zou *et al.* [129]	2020	k-means	No	BCP	DPE	-
Liping *et al.* [130]	2020	k-means, spectral	No	RSA	-	-
Abdulatif *et al.* [131]	2020	fuzzy, k-means	No	BGV	-	-
Lekshmy *et al.* [132]	2020	k-means	No	Paillier	-	-
Guo *et al.* [133]	2020	k-means	No	Paillier	-	-
Youwen *et al.* [134]	2020	k-means	Si	Paillier	-	112 bits
Catak *et al.* [135]	2020	k-means, spectral	No	Paillier	-	80 bits
X. Huo *et al.* [136]	2020	k-means	No	AES	-	-
Almutairi *et al.* [137]	2020	DBSCAN	Si	Liu	MUOPE	-
Shang *et al.* [138]	2019	k-means	No	Paillier	-	-
Minghuia *et al.* [139]	2019	k-means	No	Paillier	-	-
C. Wang *et al.* [140]	2019	k-means	No	Paillier	DPE	-
Othmane *et al.* [141]	2019	k-means	No	Paillier	DPE	-
Yunlu *et al.* [142]	2019	k-means	No	Liu	-	-
J. Cheon *et al.* [143]	2019	k-means	No	HEAAN	-	128 bits
Song *et al.* [144]	2019	k-means	No	Gentry	-	-
Georgios *et al.* [145]	2019	k-means	No	Gentry	-	128 bits
Paladhi *et al.* [146]	2019	fuzzy c-means	No	BGV	-	-
Bozhong *et al.* [147]	2019	spectral	No	Paillier	DPE	80 bits
Zhang *et al.* [148]	2019	fuzzy c-means	No	BGV	-	-
Igor *et al.* [149]	2018	DBSCAN	No	-	-	-
Almutairi *et al.* [150]	2018	NNC, k-means	No	Liu	OPE	-
Almutairi *et al.* [151]	2018	NNC	Si	Liu	OPE	-
Almutairi *et al.* [152]	2018	DBSCAN	Si	Paillier	MOPE	-

Cuadro 1: Trabajos relacionados con la preservación de privacidad en agrupamiento de datos, basados en cifrado homomórfico.

Autor	Año	Modelo de minería	DMaaS	Esquema	PPE	N. seguridad
Kumar *et al.* [153]	2023	Decision Tree, SVM	No	RSA	-	
Zhuoran *et al.* [154]	2022	Regresión lineal	No	SPDZ	-	128 bits
Xie *et al.* [155]	2022	KNN	No	ElGamal	-	80 bits
Osman *et al.* [156]	2022	Naive Bayes	No	El Gamal	-	
Skarkala *et al.* [157]	2021	Naive Bayes	No	Paillier	-	-
Pulido *et al.* [158]	2021	NN	No	-	-	-
Liu *et al.* [159]	2021	KNN	No	CKKS	-	-
Alkhelaiwi *et al.* [160]	2021	NNC	-	Paillier	-	-
Sarkar *et al.* [161]	2021	NN	No	Paillier	-	128 bits
Kuang *et al.* [162]	2021	NN	No	-	-	-
Sun *et al.* [163]	2021	NNC	No	BGV	DPE	80 bits
Zhang *et al.* [164]	2021	-	No	BCP	-	80 bits
Paul *et al.* [165]	2021	NN recurrentes	No	CKKS	-	128 bits
Mendoza *et al.* [166]	2021	Regresión logística	No	RNS	-	-
Huang *et al.* [167]	2021	SVM	No	CKKS	-	128 bits
Alzubi *et al.* [168]	2021	DNN	No	MHE-SFO	-	-
Yu *et al.* [169]	2021	Árboles de decisión	No	VHE	-	-
Baryalai *et al.* [170]	2021	NNC	No	Paillier	-	-
Chen *et al.* [171]	2021	Naive Bayes	No	BGV	-	-
Fan *et al.* [172]	2020	Regresión logística	No	Gentry	-	-
Jun Liu *et al.* [173]	2020	Regresión lineal	No	SPDZ	-	-
Ishiyama *et al.* [174]	2020	NNC	No	CKKS	-	-
Badawi *et al.* [175]	2020	NLP	No	CKKS	-	-
Syed *et al.* [176]	2020	DNN	No	Paillier	-	-
Ping *et al.* [177]	2020	SVC	No	Paillier	-	112 bits
Tho Thi *et al.* [178]	2020	KNN	No	RLEW	-	-
Qiu *et al.* [179]	2020	Regresión lineal	No	Paillier	-	80 bits
Dong *et al.* [180]	2020	Regresión lineal	No	Paillier	-	80 bits
Wang *et al.* [181]	2020	ELM	No	BGV	-	-
Jing Li *et al.* [182]	2020	Regresión lineal	No	-	-	-
Junyi Li *et al.* [183]	2020	Regresión logística	No	-	-	-
Haque *et al.* [184]	2020	KNN	Si	Paillier	-	80 bits
Chai *et al.* [185]	2020	Naive Bayes	No	Paillier	-	80 bits
Bajard *et al.* [186]	2020	SVM	Si	BGV	-	-
Obla *et al.* [187]	2020	NNC	No	CKKS	-	-
Yongshuang *et al.* [188]	2020	NN	No	Paillier	-	-
Ximeng Liu *et al.* [189]	2020	KNN	No	BGV	-	-
Park *et al.* [190]	2020	KNN	No	Paillier	-	80 bits
Zheng Li *et al.* [191]	2020	Regresión logistica	No	BGV	-	-
Chou *et al.* [192]	2020	NNC	No	BFV,CKKS	-	-
Sun *et al.* [193]	2020	Naive Bayes	No	BGV	-	-
Zorarpacı *et al.* [194]	2020	Naive Bayes	No	Paillier	-	128 bits
Yasumura *et al.* [195]	2019	Naive Bayes	Si	BGV	-	-
A. Wood *et al.* [196]	2019	Naive Baye	No	GKS	-	198 bits
Hesamifard *et al.* [197]	2019	NNC	No	Paillier	-	-
Xiao *et al.* [198]	2019	GBDT	No	CKKS	-	-
Malika *et al.* [199]	2019	NNC	No	TFHE	-	80 bits
Bellafqira *et al.* [200]	2019	MLP	No	Paillier	-	-
Jin Chao *et al.* [201]	2019	NNC	No	-	-	-

Autor	Año	Modelo de minería	DMaaS	Esquema	PPE	N. Seguridad
Marcano *et al.* [202]	2019	NNC	No	Gentry	-	-
Ye Li *et al.* [203]	2018	Árboles de decisión	No	Paillier	-	-
Yang Liu *et al.* [204]	2018	KNN	Si	Paillier	-	80 bits
Sangwook *et al.* [205]	2018	Naive Bayes	No	-	-	-
Haomiao *et al.* [206]	2018	KNN	No	VHE	-	-
Ping Li *et al.* [207]	2018	KNN, SVM, NBayes	No	DD-PKE	-	-
Bonte *et al.* [208]	2018	Regresión logística	No	FV	-	80 bits
Park *et al.* [209]	2018	Naive Bayes	No	-	-	60 bits
Ping Li *et al.* [210]	2018	Naive Bayes	No	Gentry	-	-
Wenjie Lu *et al.* [211]	2018	-	No	GM, Paillier	-	128 bits
A. Wood *et al.* [212]	2018	Naive Bayes	No	GKS	-	198 bits
J. Cheon *et al.* [213]	2018	Regresión logística	No	HEAAN	-	198 bits

Cuadro 2: Trabajos relacionados con la preservación de privacidad en clasificación de datos, basados en cifrado homomórfico.

En las siguientes secciones se discute el alcance de los trabajos descritos en las Tablas 1 y 2, en relación con métodos PPDM basados en cifrado homomórfico.

3.1. Ámbito de aplicación de DMaaS

La minería de datos como servicio (DMaaS), como ya se explicó en las secciones anteriores, permite proporcionar herramientas para el análisis de datos, a los propietarios de los datos a través de un tercero. Aunque los servicios en la nube brindan una infraestructura confiable para el almacenamiento de datos, el uso de dichos servicios conlleva problemas de confidencialidad y seguridad de datos. En consecuencia, se han propuesto algunos enfoques para abordar estos problemas [151, 152, 137, 134].

Almutairi *et al.* propusieron diferentes enfoques basados en el agrupamiento de datos, el primero de ellos, desarrollado en el 2017 [151], se enfocó en la agrupación en clústeres k-means que utiliza el concepto de una matriz de distancia actualizable (UDM). La ventaja ofrecida es que, a diferencia de los mecanismos de agrupación de k-means seguras comparables de la literatura, la participación del PD se restringe a descifrar una matriz que permite actualizar los centroides en cada iteración. En esta propuesta se consideró el escenario de múltiples propietarios de datos, a diferencia de los trabajos posteriores, en los que se trabaja con un único PD.

El segundo enfoque propuesto por Almutairi *et al.*, fue desarrollado en el 2018 [152]. En esta propuesta, a diferencia de la anterior, la participación del PD en las tareas de minería de datos como servicio es menor, ya que no es necesaria una vez que los datos se han cifrado. El modelo de minería de datos aplicado en este caso es la agrupación en clústeres de vecinos más cercanos (NNC) utilizando el concepto de matriz de distancia cifrada (EDM).

El tercer enfoque desarrollado en 2020 por Almutairi *et al.* [137] empleó matrices de distancia cifradas globales (GEDM) y conjuntos criptográficos. Cabe resaltar que los algoritmos de minería de datos estándar se pueden modificar ligeramente para trabajar con conjuntos criptográficos usando GEDM. Con este enfoque, al igual que el enfoque del 2018, no es necesaria la participación del PD en el lado del PS, una vez que los datos se encuentran cifrados. En este caso, los autores aseguran que la precisión de la agrupación en clústeres no se ve afectada.

Youwen *et al.* [134] también ofrecen DMaaS a través de un esquema de agrupamiento k-means, pero sobre datos distribuidos en redes *peer to peer*.

Por otro lado, Hammami *et al.* [214] propusieron un enfoque basado en la extracción de conjuntos de elementos cerrados dentro de un entorno de computación en la nube. El esquema propuesto adopta un cifrado homomórfico de clave simétrica para proteger la privacidad de los datos y combinarlo con una firma homomórfica para verificar la integridad de la agregación de datos. Además, durante el descifrado de datos agregados, el máster de estos sitios puede clasificar los datos cifrados y agregados basándose en claves de cifrado. Los demás trabajos propusieron un método

PPDM, pero no en el contexto de DMaaS, por lo que no consideraron la problemática asociada a la delegación de los datos a un tercero para análisis.

3.2. Modelo de minería de datos

A continuación se presenta un análisis de las diferentes técnicas de minería de datos seguras aplicadas a los enfoques propuestos mostrados en la Tabla 1 y 2.

Tanto Kim *et al.* [215] como Skarkala *et al.* [216] hacen uso de un clasificador Naive Bayes (NBC, por sus siglas del Inglés). NBC es un método de aprendizaje automático basado en la inferencia bayesiana, que asume la independencia condicional de las características [217] que describen a las entidades que se desean clasificar. En el caso de Kim *et al.* [215], se propuso un NBC que preserva la privacidad en un sistema de cifrado totalmente homomórfico (PP-NBC). A través del proceso de clasificación en PP-NBC, las operaciones se evalúan utilizando datos cifrados aplicando un esquema de cifrado totalmente homomórfico para que los proveedores de servicios puedan manejar los datos del cliente sin conocer sus valores reales. En cambio, Skarkala *et al.* [216] trabajaron en una variación más robusta de NBC, conocida como clasificador Naive Bayes aumentado en árbol (TAN, por sus siglas del Inglés). TAN, a diferencia de NBC permite la existencia de aristas adicionales entre atributos que representan la correlación entre ellos, es decir, permitir que se usen estructuras en forma de árbol para representar las dependencias entre atributos [218]. En el enfoque TAN, que preserva la privacidad, se requiere la participación de al menos tres propietarios de datos. Cuando las tres partes han enviado sus datos al servicio de minería de datos en la nube, se aplica la primitiva homomórfica. El servicio de análisis calcula las frecuencias totales de cada posible valor de atributo en relación con cada valor de clase, descifrando todos los mensajes recibidos simultáneamente. Sin embargo, el servicio en la nube no está en condiciones de asociar las frecuencias recibidas con los registros originales y no puede vincular los datos a sus propietarios debido a la ejecución del proceso de descifrado. A través del clasificador se generan correlaciones entre los atributos, dando como resultado la creación de una estructura de red que los representa.

Por otro lado, Qiu *et al.* [219] utilizaron Regresión Lineal que preserva la privacidad (PPLR). Los modelos de regresión lineal se emplean a menudo para explorar la relación entre un resultado continuo y variables independientes [220]. PPLR requiere de varios clientes y dos servidores no contaminantes. La idea es que cada cliente envíe sus datos en forma cifrada a un servidor, y dos servidores determinen de forma colaborativa el modelo de regresión en datos agrupados sin conocer su contenido. Una vez enviados los datos, el cliente puede estar fuera de línea, ya que no participa en ningún cálculo posterior.

Como se mencionó en anteriormente, Almutairi *et al.* propusieron tres enfoques diferentes. 1) en 2017 haciendo uso de k-means, 2) en 2018 [221] haciendo uso de NNC, y 3) en 2020 [222] utilizando el modelo DBSCAN.

- k-means: El algoritmo de agrupación k-means usa la distancia como métrica y dadas las k clases en el conjunto de datos, calcula la media de la distancia, dando el centroide inicial, con cada clase descrita por el centroide [223]. En este enfoque se propuso un mecanismo para la agrupación segura de k-means, haciendo uso del cifrado homomórfico y de una matriz de distancia actualizable (UDM). Más específicamente, la idea fue que los datos se almacenaran, utilizando almacenamiento de terceros, de forma cifrada, junto con una UDM asociada. La UDM contiene las distancias cifradas entre valores de datos, de modo que se puedan realizar comparaciones.
- *NNC*: La regla de decisión NNC asigna a un punto de muestra no clasificado la clasificación del más cercano de un conjunto de puntos previamente clasificados [224]. Este enfoque opera con la idea de una matriz de distancia cifrada (EDM), que integraron en un enfoque para la agrupación en clústeres de vecinos más cercanos seguros (SNNC). El algoritmo SNNC tiene un funcionamiento similar al NNC estándar, pero está diseñado para operar sobre datos cifrados sin participación adicional del usuario una vez que los datos se han subcontratado. La seguridad del enfoque de agrupamiento propuesto se basa en la seguridad de: 1) el esquema de Liu para cifrar los datos sin procesar, y 2) el orden de preservación de cifrado (OPE) usado para cifrar la matriz de distancia. Esta matriz mantiene distancias cifradas entre valores de datos, de modo que se puedan realizar comparaciones.
- *DBSCAN*: DBSCAN es un algoritmo de agrupación basado en la densidad de distribución de puntos de datos. El algoritmo puede identificar el grado de densidad de datos y clasificar los puntos de datos en la distribución [225]. El enfoque propuso la agrupación segura para DBSCAN, en este caso utilizaron Matrices de distancia cifradas

globales (GEDM), estas permiten la agrupación de datos colaborativos de terceros sin involucrar la participación del PD y sin pérdida de precisión. El GEDM es la unión de dos o más EDM que se generan de forma segura con una participación limitada del PD, usando un "método de agrupación"propuesto y el esquema de Conjunto Criptográfico, con el apoyo de un Tercero Semi-honesto (STP).

En el caso de Hao *et al.* [226] utilizaron redes neuronales para preservar la privacidad. Las redes neuronales se pueden aplicar para simular el comportamiento de una variable usando datos experimentales disponibles[227]. En este enfoque propusieron un esquema de aprendizaje de red neuronal profundo que preserva la privacidad basado en el método de descenso de gradiente estocástico mediante la integración del cifrado aditivamente homomórfico con privacidad diferencial. La técnica de privacidad diferencial se utiliza para evitar amenazas de colusión entre el servidor en la nube y múltiples usuarios.

Finalmente, Hammami *et al.* [214] trabajaron sobre la extracción de patrones cerrados frecuentes. La minería de patrones frecuentes se emplea como una subfase para obtener los elementos frecuentes que ocurren juntos en un conjunto de transacciones [228]. El enfoque propuesto trabajó con extracción de patrones cerrados frecuentes en un entorno distribuido, adoptaron un cifrado homomórfico de clave simétrica para proteger la privacidad de los datos y lo combinaron con una firma homomórfica para verificar la integridad de la agregación de datos.

3.3. Tipo de cifrado homomórfico

El principal desafío del cifrado homomórfico (HE, por sus siglas en inglés) en el contexto de DMaaS en general, y la agrupación de datos en particular, es que los esquemas de HE solo admiten un número limitado de operaciones aritméticas. De acuerdo con esto, el HE puede dividirse en tres tipos: completo, parcial y algo homomórfico. En el caso del cifrado completo o totalmente homomórfico, es posible utilizar dos operaciones aritméticas: la suma y la multiplicación (los operadores ∇ y $\circ$ en la sección 4). A diferencia del cifrado homomórfico parcial, el cual únicamente soporta un tipo de operación, ya sea la suma o la multiplicación (solo ∇ o $\circ$).

La mayoría de las propuestas consideran el cifrado parcial de Paillier [123, 132, 133, 134, 135, 138, 139, 140, 141, 147, 152, 157, 160, 161, 170], otros el de CKKS [125, 159, 165, 167, 174, 175, 187, 192, 198], RSA [130, 133], DD-PKE [207] y algunos construyen el esquema con el cifrador simétrico AES (no homomórfico) [136].

Sin embargo, otros consideran una mejor opción el cifrado homomórfico completo como el esquema de Liu [137, 142, 150, 151], el de Gentry [144, 145, 202, 210, 172], WAP [122], YASHE [124], HEAAN [143, 213], ElGamal [155], VHE [169, 206], GKS [212, 196], TFHE [199] o GM [211].

Una menor cantidad considera esquemas de cifrados algo homomórfico, tal como BGV [131, 146, 148, 163, 171, 181, 186, 189, 191, 193, 195], BCP [129, 164], RLEW [121, 178] o SPDZ [173, 154]

3.3.1. Niveles de seguridad

Medir la seguridad de los modelos propuestos requiere de conocer el nivel de seguridad con el que se trabajó en el cifrado utilizado, expresado en el tamaño de bits de la llave criptográfica. Este nivel de seguridad está en función del tamaño del grupo al que pertenece el problema matemático subyacente que enmarca la construcción criptográfica. Tanto para sumas como multiplicaciones, el grupo más utilizado en el grupo multiplicativo, el cual está definido por un número primo grande.

Para los grupos multiplicativos, el nivel de seguridad actual recomendado es de 2048 bits, esto es, el número primo que lo define debe ser al menos de un tamaño de 2048 bits. En los trabajos relacionados, solo el modelo propuesto por Skarkala *et al.* [216] reporta un nivel de seguridad de 1024 bits, el cual es insuficiente. Otros trabajos como el de Qiu *et al.* [219] y Kim *et al.* [215], que también trabajan sobre este grupo, no reportan el tamaño del grupo usado.

En el caso del grupo aditivo, el nivel de seguridad actual recomendado es de 224 bits. Pero, al igual que los modelos propuestos sobre el grupo multiplicativo, este nivel de seguridad no es reportado. Los modelos propuestos que trabajan sobre este grupo son los de: Hao *et al.* [226], y Hammami *et al.* [214].

3.3.2. Algoritmo de cifrado homomórfico

Un sistema de HE proporciona la capacidad de realizar varios tratamientos en datos cifrados sin utilizar la operación de descifrado. En el HE, determinadas funciones de agregación como la suma y el promedio se pueden aplicar sobre los datos cifrados. Los esquemas de HE son especialmente útiles cuando algunas partes no tienen la(s) clave(s) de descifrado, mientras que las otras partes necesitan realizar operaciones aritméticas en un conjunto de textos cifrados. Dentro de los esquemas de HE utilizados en la literatura, aplicados a PPDM, destacan el esquema de Liu y el cifrado de Paillier, los cuales se describen a continuación:

El esquema de Liu es probabilístico, por lo tanto, produce diferentes textos cifrados para el mismo texto plano cada vez que se aplica, incluso cuando se usa la misma clave criptográfica. Admite la suma y resta de textos cifrados y la división y multiplicación de textos cifrados. Además, transfiere valores de texto plano aleatoriamente a textos cifrados, en consecuencia, cualquier orden que pueda aparecer en los datos de texto plano no se transfiere a los textos cifrados.

En cuanto al algoritmo de Paillier, es un esquema de cifrado de clave pública que explota la primitiva homomórfica aditiva para lograr el anonimato y la desvinculación entre las partes y los datos personales. Permite dos tipos de cálculos: 1) adición de dos textos cifrados, y 2) multiplicación de un texto cifrado por un número de texto plano. El esquema de Paillier se explica con más detalle en la sección 4.1.1.

3.4. Discusión

El nivel de seguridad está relacionado intrínsecamente con la eficiencia del método criptográfico. Un método más seguro usará llaves más grandes, pero eso se traducirá en un costo computacional mayor, lo que llevará a tener un bajo desempeño del método. En los contextos actuales del cómputo en la nube, es deseable contar con despliegues eficientes de los métodos de cifrado, haciendo uso de técnicas de paralelismo u otros modelos que permitan contar con métodos tanto seguros como eficientes. Es en esta parte donde existe un área de oportunidad de investigación y desarrollo, y en la que se este proyecto de tesis busca contribuir.

Los métodos criptográficos homomórficos se definen en estructuras algebraicas donde se define un problema matemático. La dificultad para resolver ese problema determina el nivel de seguridad del método criptográfico. Por ello, las estructuras algebraicas deben ser de tamaño suficiente para que el problema sea difícil de resolver. Esta suposición se ha mantenido hasta estos tiempos, de que no existe un algoritmo eficiente (con complejidad polinomial) que pueda ejecutarse en una computadora convencional, que resuelva el problema en el que el cifrado homomórfico sustenta su seguridad. Sin embargo, los algoritmos que resuelven el problema matemático, y que son de complejidad exponencial, sí podrían ejecutarse en tiempo polinomial en una computadora cuántica. Por ello, los métodos criptográficos homomórficos, sobre los que se construyen los PPDM, están amenazados en un escenario postcuántico.

4. PPDM basado en esquemas criptográficos

En esta sección se describen PPDM representativos en el estado del arte, basados en cifrado homomórfico, el cual es uno de los enfoques más utilizados para preservar la privacidad en las tareas de minería de datos, por ejemplo, las tareas de agrupamiento o clasificación.

4.1. Cifrado homomórfico con el esquema de Liu

El esquema de Liu [229] es un esquema de cifrado simétrico totalmente homomórfico conformado por tres algoritmos: 1) generación de llaves (`KeyGen`), 2) cifrado (`Encrypt`), y 3) escifrado (`Decrypt`). Las características de cada uno de estos algoritmos se describen a continuación:

- **`KeyGen`**: Crea una llave simétrica SK como una tupla de m componentes, cada una de tres elementos: $SK_m = [(k_1, s_1, t_1), ..., (k_m, s_m, t_m)]$, donde k_i, s_i y t_i son valores aleatorios en $\mathbb{R}$. SK_m debe cumplir condiciones asociadas con el valor de m y a las componentes k_i, s_i y t_i [229]. Como cualquier otro cifrador simétrico, la seguridad radica en la dificultad para realizar un ataque por fuerza bruta a SK_m, la cual debe ser lo suficientemente grande.
- **`Encrypt`**: Se recibe como entrada un texto en claro ($v \in \mathbb{R}$) y la llave secreta SK_m. Como resultado, regresa el texto cifrado de v que consiste en una tupla de m componentes $E_v = [e_1, e_2, \cdots, e_m]$, donde $e_i \in \mathbb{R}$ obtenido como una combinación algebraica de los elementos de SK_m, el texto en claro v, y valores aleatorios generados durante el proceso. El pseudocódigo para implementar cifrado de Liu se muestra en el Algoritmo 4.

Algorithm 4 Cifrado con el esquema de Liu

procedure ENCRYPT(v, SK_m)
  Uniformly generate m large random numbers in $\mathbb{R}$, $\{r_1, \ldots, r_m\}$
  Declare E_v as a set of m elements in $\mathbb{R}$, $E_v = [e_1, e_2, \cdots, e_m]$
  $e_1 = k_1 \times t_1 \times v + s_1 \times r_m + k_1 \times (r_1 - r_{m-1})$
  for $i = 2$ to $m - 1$ **do**
    $e_i = k_i \times t_i \times v + s_i \times r_m + k_i \times (r_i - r_{m-1})$
  end for
  $e_m = (k_m + s_m + t_m) \times r_m$
  Exit with E_v
end procedure

- **`Decrypt`**: Recibe como entrada el texto cifrado E_v y la llave simétrica SK_m, ambos de m componentes. Como resultado, se regresa el texto en claro original $v \in \mathbb{R}$. El pseudocódigo para implementar el descifrado de Liu se muestra en el Algoritmo 5.

Como se mencionó anteriormente, el esquema de Liu es totalmente homomórfico. Permite realizar operaciones sobre datos cifrados bajo los operadores de suma y multiplicación. Además, se definen dos operaciones adicionales útiles en algunos algoritmos de minería de datos: la multiplicación por un escalar y la resta. Estas operaciones se explican a continuación:

- **Suma**: Sean $E_1 = [e_{1_1}, \cdots e_{1_m}]$ y $E_2 = [e_{2_1}, \cdots e_{2_m}]$, los textos cifrados de v_1 y v_2 respectivamente de acuerdo con el Algoritmo 4, usando la clave SK_m.

  Se define la suma de E_1 y E_2 como se muestra en la Ecuación 3, dando como resultado E_3. La propiedad homomórfica satisface que `Decrypt`(E_3, SK_m) = $v_1 + v_2$.

$$E_3 = E_1 \oplus E_2 = [e_{1_1} + e_{2_1}, \cdots e_{1_m} + e_{2_m}] \tag{3}$$

Algorithm 5 Liu scheme decryption algorithm

1: **procedure** DECRYPT(E_v, $SK(m)$)

2: $t = \sum_{i=1}^{m-1} t_i$

3: $s = \frac{e_m}{(k_m + s_m + t_m)}$

4: $v = \frac{\sum_{i=1}^{m-1} t_i(e_i - s \times s_i)/k_i}{t}$

5: **Exit** with *v*

6: **end procedure**

- **Multiplicación**: Sean $E_1 = [e_{1_1}, \cdots e_{1_m}]$ y $E_2 = [e_{2_1}, \cdots e_{2_m}]$, los cifrados de v_1 y v_2, respectivamente, de acuerdo con el Algoritmo 4, usando la clave SK_m.

  Se define la multiplicación de E_1 y E_2 como E_3, de acuerdo con la Ecuación 4. La propiedad homomórfica satisface que `Decrypt`(E_3, SK_m) = $v_1 \times v_2$.

$$E_3 = E_1 \otimes E_2 = [e_{1_1} \times e_{2_1}, \cdots, e_{1_1} \times e_{2_m}, \cdots, e_{1_m} \times e_{2_1}, \cdots, e_{1_m} \times e_{2_m}] \tag{4}$$

- **Multiplicación por un escalar**: Sea $E_1 = [e_1, \cdots e_m]$ el resultado de cifrar v_1, de acuerdo con el Algoritmo 4, usando la clave SK_m. Sea c un escalar.

  La multiplicación de c con v_1 se define como se muestra en la Ecuación 5, de tal forma que `Decrypt`(E_3, SK_m) = $c \times v_1$.

$$E_c = c \odot E_1 = [c \times e_1, \cdots c \times e_m] \tag{5}$$

- **Resta**: A partir de la multiplicación por un escalar y de la suma, es posible definir la resta del texto cifrado E_2 a E_1. Sea $E_1 = [e_{1_1}, \cdots, e_{1_m}]$ y $E_2 = [e_{2_1}, \cdots, e_{2_m}]$, los resultados de cifrar v_1 y v_2, respectivamente, de acuerdo con el Algoritmo 4, usando la clave SK_m. La resta se define como se muestra en la Ecuación 6. La propiedad homomórfica hace cumplir que `Decrypt`(E_3, SK_m) = $v_1 - v_2$.

$$E_3 = E_1 \ominus E_2 = E_1 \oplus ((-1) \odot E_2) \tag{6}$$

4.2. Esquema homomórfico de Paillier

El esquema de Paillier es un esquema homomórfico parcial que permite dos tipos de operaciones: suma y multiplicación por un escalar.

Este esquema se define sobre el espacio de mensajes de enteros positivos $\mathcal{M} = (Z_N^+)$, los valores a cifrar van de 0 a $N - 1$, donde N representa el módulo RSA [230]. Los algoritmos que componen este criptosistema se describen a continuación:

- **`KeyGen`**: Este algoritmo permite generar tanto la clave pública, como la clave privada.

  Los pasos para generar las claves son los siguientes:

  1. Seleccionar dos números primos aleatorios distintos: p y q, de k bits de longitud. Cabe resaltar que para garantizar seguridad con este esquema, se recomienda que k sea de al menos 2048 bits.

2. Se calcula n y λ tal como se ve en la ecuación 7

$$\begin{aligned} n &= p \times q \\ \lambda &= mcm(p-1, q-1) \end{aligned} \tag{7}$$

3. Seleccionar un entero g tal que $g \in Z^*_{n^2}$
4. Se valida que n divide a g utilizando la ecuación 8

$$\begin{aligned} \mu &= (L(g^\lambda mod\ n^2))^{-1}\ mod\ n \\ L(u) &= \frac{u-1}{n} \end{aligned} \tag{8}$$

La clave pública es (n, g) y la clave privada es (λ, μ)

- **`Encrypt`**: Este algoritmo permite hacer el cifrado de un texto en claro.
  1. Sea v el texto a cifrar, tal que $v \in \mathbb{Z}_n$,
  2. Se elige un número aleatorio r donde $0 < r > n$, tal que $r \in \mathbb{Z}^*_n$
  3. El mensaje cifrado c se calcula con la ecuación 9

$$c = g^m \cdot r^n (mod\ n^2) \tag{9}$$

- **`Decrypt`**: Este algoritmo permite hacer el descifrado de un texto cifrado.
  1. Dado un texto cifrado c, tal que $c \in \mathbb{Z}_{n^2}$
  2. El texto descifrado se calcula con la ecuación 10

$$v = L(c^\lambda\ mod\ n^2) \cdot \mu\ mod\ n \tag{10}$$

El rendimiento (eficiencia) del esquema Paillier y sus propiedades matemáticas HE se basan en el parámetro de seguridad seleccionado k.

El esquema PHE de Paillier tiene una característica homomórfica aditiva que asigna la suma (+) de texto plano a la multiplicación cifrada ($\otimes$) tal como se ve en la ecuación 11

$$e_1 \otimes e_2\ (mod\ N^2) = v_1 + v_2 \tag{11}$$

Además, Paillier tiene propiedades homomórficas adicionales que permiten la multiplicación por un escalar. Esta multiplicación se puede utilizar para realizar la resta de textos cifrados (ecuación 12), como en el caso del esquema de Liu.

$$c \circledast e = e^c\ (mod\ N^2) = c \times v \tag{12}$$

4.3. FDH-OPE

El esquema de Cifrado de Preservación de Orden, de Ocultación de Frecuencia y Distribución (FDH-OPE) [17] es una fusión de dos esquemas OPE existentes: 1) el esquema de preservación de orden no lineal [231] y el esquema de Zheli *et. al* [232]. El primero oculta la frecuencia de los datos en los textos cifrados generados, y el segundo oculta la distribución de los datos. Para ello se requiere conocer la distribución de los textos en claro y los intervalos del texto en claro.

El esquema FDH-OPE utiliza dos funciones: 1) `KeyGen` para generar las claves de cifrado y 2) `Encrypt` para cifrar valores de texto en claro (usando las claves de cifrado generadas). FDH-OPE no cuenta con una función de descifrado, ya que solo se requiere el cifrado para establecer una relación de orden de los textos cifrados sin comprometer la seguridad (revelar) de los textos en claro.

- **KeyGen**: El espacio de textos en claro se establece como el intervalo semiabierto $\mathcal{M}$ = [l, h). El espacio de textos cifrados se establece como el intervalo semiabierto $\mathcal{C}$ = [l', h'), siendo qué $\mid \mathcal{M} \mid \ll \mid \mathcal{C} \mid$. En FDH-OPE, el cifrado de un elemento v en $\mathcal{M}$ es su mapeo a un elemento v' en $\mathcal{C}$, de manera no determinista. Este proceso de extrapolación se realiza de la siguiente forma: se divide aleatoriamente $\mathcal{M}$ en t sub-intervalos $\mathcal{M} = \{m_1 \cup ... \cup m_t\}$ ($m_i \cap m_j =, \forall i \neq j$), siendo t un número aleatorio. Si se considera que $D = \{v_1, v_2, ..., v_n\}$ es el conjunto de datos a cifrar y que para todo v_i, $v_i \in [l, h)$, entonces v_i pertenece a algún m_i. Se calcula la densidad de datos $dens_i$ para cada sub-intervalo m_i, es decir, la cantidad de valores en D que se encuentran en m_i. Este valor permite dividir $\mathcal{C}$ en el mismo número de t sub-intervalos $\mathcal{C} = \{c_1 \cup c_2 \cup ... \cup c_t\}$, tal que si $dens_i > dens_j$ entonces $\mid c_i \mid > \mid c_j \mid$. Lo anterior permite mantener la relación de orden entre los elementos v_i, v_j en $\mathcal{M}$ y sus correspondientes versiones cifradas en $\mathcal{C}$. Así, el proceso de cifrado se convierte en una extrapolación de un elemento en algún intervalo m_i hacia uno en c_i. Cada m_i está definido como un intervalo semiabierto $[l_i, h_i)$. De manera análoga, cada intervalo $c_i = [l'_i, h'_i)$. Note que $l = l_1$, $h_i = l_{i+1}$, $h_t = h$.

Algorithm 6 Generación de llaves FDH-OPE

1: **procedure** KEYGEN(D)
2: $[l, h) = crearIntervalo(D)$
3: $t = rand(4, 8)$
4: $m[] = Split(l, h, t)$ ▷ t intervalos semi cerrados $[m[i].l_i, m[i].h_i)$
5: $n = |D|$
6: $[l', h') = crearIntervalo(l, h)$ ▷ $|[l, h)|$ mucho menor que $|[l', h')|$
7: $dens[] = computeDensity(D, m, t, n)$ ▷ t densidades, una para cada intervalo en M
8: $c[]$ = Split(l', h', t,dens) ▷ t intervalos semi cerrados $[c[i].l'_i, c[i].h'_i)$
9: **Exit** with $\{m, c\}$
10: **end procedure**

El intervalo $\mathcal{M}$ = [l', h') y los sub-intervalos m_i, c_i constituyen la llave privada del esquema FDH-OPE. El algoritmo 6 implementa la operación de generación de llaves del esquema FDH-OPE, teniendo como entrada el conjunto de datos D. Como salida, se regresan el espacio de mensajes, siendo l, h, l',h' las llaves secretas. Primero (línea 2), a partir de los datos de entrada se determina el intervalo [l, h). Debe ser un espacio de valores donde queden contenidos todos los valores en D. Por simplicidad, solo se considera un espacio de valores positivos, por lo que se obtiene el máximo valor absoluto en D. Un valor negativo en D se procesará como positivo y el signo se restaurará al final del procesamiento, sin que resulte una afectación en la extrapolación y en el orden de los elementos. Más detalle de esto se dará en la operación de cifrado. Ya definido $\mathcal{M}$, se procede a determinar sus t sub-intervalos $m_i = [l_i, h_i)$ (líneas 3 y 4), de manera no uniforme, es decir, para cualquiera dos sub-intervalos m_i, m_j sus tamaños no son necesariamente iguales. A continuación se determina el intervalo $\mathcal{C}$ = [l', h') hacia donde se realizará la extrapolación. Se espera que $\mathcal{C}$ sea mucho mayor que $\mathcal{M}$. La división de $\mathcal{C}$ en t sub-intervalos considera la densidad de D en $\mathcal{M}$, misma que se calcula en la línea 7. $dens_i$ es el número de valores en D que caen en el intervalo m_i. Así, la longitud s_i de c_i será proporcional a $dens_i$ respecto al total de valores en D, es decir $s_i = \frac{h' \times dens_i}{|D|}$. Por lo anterior, resulta que $l' = l'_1$, $h'_i = s_i = l'_{i+1}$, $h'_t = h'$.

- **Encrypt**: El proceso de cifrado FDH-OPE se muestra en el Algoritmo 7. El cifrado es no determinista, por lo que se usa un valor aleatorio para garantizar que si $v_1 = v_2$, entonces $\texttt{encrypt}(v_1) \neq \texttt{encrypt}(v_2)$. Primero, se identifica el intervalo i (línea 2) al que $v \in D$ pertenece en $\mathcal{M}$ (m_i) y, por tanto, el intervalo al que se debe extrapolar en $\mathcal{M}$ (c_i). La interpolación se calcula usando la ecuación general de la recta (línea 5), siendo $\theta_i = \frac{h'_i - l'_i}{h_i - l_i}$ la pendiente (línea 3). Es en este punto donde se calcula la componente aleatoria δ_i para cifrado determinista. Para ello, se considera la sensibilidad ($sens$), que se refiere a la diferencia mínima $\mid v_1 - v_2 \mid$ entre cualquiera dos valores en claro v_1 y v_2 en D. Lo anterior previene que si $v_1 < v_2$, debido a δ_i resulte que $\texttt{encrypt}(v_1) \geq \texttt{encrypt}(v_2)$, lo cual afectaría la preservación de orden. Como resultado final, el algoritmo regresa el texto cifrado v', tomando en cuenta su signo de entrada.

Algorithm 7 Cifrado FDH-OPE

procedure ENCRYPT(*v*,*sens*)
  $i \leftarrow$ IntervalID(|v|)
  θ_i = $\dfrac{c[i].h'_i - c[i].l'_i}{m[i].h_i - m[i].l_i}$
  δ = Random(0, $sens \times \theta_i$)
  v' = $c[i].l'_i + \theta_i \times$ (|v| $- m[i].l_i$) + δ_i
  If $v < 0$ **then**
    $v' = v' \times -1$
  end if
  Exit with v'
end procedure

4.4. PPDM para la tarea de agrupamiento

Algunos ejemplos de algoritmos de minería de datos con preservación de privacidad son los que se presentan en las subsecciones siguientes.

4.4.1. S*k*-means

Como ya se ha comentado, k-means ejecutado por un tercero no confiable, violaría la privacidad de los datos si estos fueran de naturaleza sensible o estuvieran protegidos por alguna regulación.

En la literatura se ha propuesto una versión de k-means segura llamada Sk-means [233], que en lugar de operar sobre el conjunto de datos D en claro, opera sobre D cifrado. De manera formal, se puede definir el conjunto de datos como $D = \{r_1, r_2, \cdots, r_n\}$, donde cada registro r_x representa un conjunto de valores que corresponden a un conjunto de atributos $A = \{r_{x,1}, \cdots r_{x,a}\}$, donde cada elemento de $r_{x,y}$ pertenece a $\mathbb{R}$. El cifrado de D se puede definir como $D' = \{E_1, E_2, \cdots, E_n\}$ donde cada E_i representa el cifrado de r_i.

Sk-means funciona de manera similar a k-means, pero realiza el agrupamiento sobre los elementos en D' que son los textos cifrados resultado del cifrado de Liu ($E_i = [e_{i_1}, \cdots, e_{i_m}]$, ver Sección 4.1), haciendo uso de las propiedades del cifrado homomórfico [234]). La tarea de agrupamiento con preservación de privacidad mediante Sk-means involucra distintas tareas: unas dependerán del PD y otras del PS. El proceso se inicia de parte del PD, quien se encarga de hacer el cifrado de los datos, así como el cálculo de una matriz de distancias U. Ambos elementos son enviados al PS, quien ejecutará la tarea de agrupamiento.

$$U = \begin{bmatrix} (u_{1,1,1}, \cdots, u_{1,1,a}) & (u_{1,2,1}, \cdots, u_{1,2,a}) & \cdots & (u_{1,k,1}, \cdots, u_{1,k,a}) \\ \vdots & \vdots & \ddots & \vdots \\ (u_{n,1,1}, \cdots, u_{n,1,a}) & (u_{n,2,1}, \cdots, u_{n,2,a}) & \cdots & (u_{n,k,1}, \cdots, u_{n,k,a}) \end{bmatrix} \tag{13}$$

La matriz U, mostrada en la Eq. 13 contiene n renglones y k columnas, donde $u_{x,y,z} = r_{x,z} - cent_{y,z}$ representa la distancia del registro r_x al centroide y, en relación con cada atributo z. Inicialmente, los k centroides son los primeros k registros de D, por lo que en la primera iteración $u_{x,y,z} = r_{x,z} - r_{y,z}$. Por simplicidad, $u_{x,y}$ representa la distancia entre los registros x y y, en claro, en la primera iteración del algoritmo. Para las futuras iteraciones, $u_{x,y}$ representa la distancia del registro y al centroide $cent_y$. Desde la perspectiva del PS, el centroide se calcula con datos cifrados, desde la perspectiva del PD, ese centroide debe estar en claro. Por ello, se requiere una interacción entre el PS y el propietario para descifrar el centroide y actualizar la matriz U. Aprovechando las propiedades homomórficas, se obtiene la diferencia (desplazamiento) entre el centroide en la iteración anterior y la siguiente (ambas cifradas) y esa diferencia es la que el proveedor envía al proveedor para descifrar y que permite actualizar a la matriz U en el lado del PS.

El algoritmo 8 especifica la preparación del conjunto de datos D desde el lado del propietario. En primer lugar, se genera la clave secreta (SK) y se cifra cada registro en D, que está conformado por a atributos. Esto permite obtener

D'. Posteriormente, se crea una matriz vacía U (explicada anteriormente). Se itera sobre cada par de registros en D, calculando la diferencia entre los valores correspondientes a cada atributo, guardando esta distancia en la matriz U, lo que permite un análisis posterior de los datos externalizados sin revelar información sensible de los datos originales.

Algorithm 8 Preparación de los datos desde el lado del PD (SKMEANS)

procedure OUTSOURCEDATA(D, a, k, m)
  SK_m = Liu scheme secret key ▷ Seccion 4.1
  $D' = \emptyset$
  for all $r \in D$ **do**
    D' = $D' \cup$ `Encrypt`(r, SK_m) ▷ Cifrado de Liu (Algoritmo 4)
  end for
  U = Empty UDM of dimensions $|D| \times |D| \times a$
  for $x = 1$ to $|D|$ **do**
    for $y = 1$ to k **do**
      for $z = 1$ to a **do**
        $u_{x,y,z} = r_{x,z} - r_{y,z}(u_{x,y,z} \in U)$
      end for
    end for
  end for
  Exit with D' and **U**
end procedure

A partir del cálculo de D' y U, el PS ejecutará el algoritmo de agrupamiento. En la iteración i, los registros $E_j \in D'$ más cercanos al centroide $cent_y^{(i)}$ (de acuerdo con U) se agruparán en el clúster C_y. Para $i = 1$, $cent_y^{(1)} = \{E_1, E_2, ..., E_k\}$. En la iteración $i + 1$, los centroides se actualizarán como $cent_y^{(i+1)} = average(C_y)$, siendo $average(C_y)$ el promedio de todos los registros E_j en C_y en ese momento. Este promedio es posible calcularlo usando los operadores homomórficos del esquema de Liu. El resultado será un nuevo centroide de a componentes, cada uno asociado al promedio de todos los atributos cifrados en cada registro cifrado E_j.

En la siguiente iteración, el proceso se repite, por lo que al tener nuevos centroides, la matriz U se debe recalcular para reflejar la nueva distancia de r_x a $cent_y^{(i+1)}$. Esta actualización de U se puede realizar de la siguiente forma: 1) calcular el desplazamiento $S'_y = cent_y^{(i)} \ominus cent_y^{(i+1)} = (s'_{y,1}, s'_{y,2}, ..., s'_{y,a})$ y 2) sumar este desplazamiento a todos los vectores en la columna y en U (columna asociada al centroide y). Sin embargo, dado que S'_y son valores cifrados y los valores en U son texto en claro, será necesario primero descifrar S'_y en $S_y = (s_{y,1}, s_{y,2}, ..., s_{y,a})$ y después realizar la suma, como se muestra en la Eq. 14. La actualización, considerando todos los desplazamientos $S'_y \in S'$, $1 \leq y \leq k$, se muestra en el Algoritmo 9.

$$update(U, S_y) = \begin{bmatrix} (u_{1,y,1} + s_{y,1}, \cdots, u_{1,y,a} + s_{y,a}) \\ \vdots \\ (u_{n,y,1} + s_{y,1}, \cdots, u_{n,y,a} + s_{y,a}) \end{bmatrix} \tag{14}$$

El PS calcula S'_y, luego se calcula el descifrado S_y del lado del PD y se envía al PS. Una vez recibida S_y se almacena en S y se procede a ejecutar la siguiente iteración $i + 1$ de la tarea de agrupamiento. El proceso termina cuando $S'_y = 0$, para todos los centroides $cent_y^{(i+1)}$, es decir, no hay cambios en relación con los centroides en la iteración previa.

Algorithm 9 Proceso de actualización de U, en el lado del PS

1: **procedure** UPDATEU(U, S')
2: $S = \emptyset$
3: **for** $S_y \in S'$ **do**
4: S_y = `Decrypt`(S'_y) ▷ Descifrado de Liu (Algoritmo 5)
5: $S = S \cup S_y$
6: **end for**
7: **U** = $U + S$
8: **Exit** with **U** and S
9: **end procedure**

El algoritmo Sk-means ejecutado en el lado del proveedor se muestra en el Algoritmo 11, el cual recibe como entrada D', U y k. Como salida, el algoritmo regresa el conjunto de agrupaciones finales $C = \{C_1, C_2, ..., C_k\}$, es decir, grupos que contienen a subconjuntos disjuntos de registros cifrados E_j en D'. Este modelo se envía al PD, quien entonces los puede descifrar para obtener el modelo de agrupamiento en claro.

El algoritmo toma los primeros k elementos de D' y los utiliza como los centroides en la primera iteración (línea 7). Se define como $Cent^{(i)} = \{cent_1^{(i)}, \cdots, cent_k^{(i)}\}$ al conjunto de centroides en la i-ésima iteración (línea 9). En este momento, la matriz U tiene toda la información para saber la distancia de todos los registros restantes a los centroides iniciales. Por ello, los registros restantes $k+1$ a n se consideran en la primera agrupación con la llamada a la función `doClustering` (línea 12) (Algoritmo 10), en la que cada registro cifrado E_x en D' ($1 \leq x \leq n$) se coloca en el cluster C_y si la distancia de E_x a $cent_y^{(i)}$ (Eq. 15) es la más corta (información tomada de U).

$$dis\left(U, E_x, cent_y^{(i)}\right) = \sum_{z=1}^{z=a} |u_{x,y,z}| \tag{15}$$

Algorithm 10 Proceso de agrupamiento

1: **procedure** DOCLUSTERING(*numReg*, U, C, D', $Cent^{(i)}$)
2: $n = |D'|$
3: $k = |C|$
4: **for** x= numReg to n **do**
5: id = *-1*
6: **for** y=1 to k **do** ▷ Test the min distance of E_x to all clusters
7: $d = dis(U, E_x, cent_y^{(i)})$ where $E_x \in D'$ and $cent_y^{(i)} \in Cent^{(i)}$
8: id = cluster identifier with lowest d value so far
9: **end for**
10: $C_{id} = C_{id} \cup E_x$ ▷ Add E_x to the selected cluster
11: **end for**
12: **Exit** with C
13: **end procedure**

Algorithm 11 Algoritmo de agrupación segura (Sk-means)

1: **procedure** SK-MEANS(*D'*, **U**, *k*)
2: Assign the first *k* records in D' to C (one per cluster)
3: *i* = 1
4: $C = \emptyset$ ▷ set of *k* empty clusters
5: $Cent^{(i)} = \emptyset$ ▷ set of *k* empty centroids
6: **for** $y = 1$ to k **do** ▷ centroids and clusters are the first k records
7: $cent_y^{(i)}$ = $E_y \in D'$
8: $C_y = E_y$
9: $Cent^{(i)} = Cent^{(i)} \cup \{cent_y^{(i)}\}$
10: $C = C \cup \{C_y\}$
11: **end for**
12: `doClustering`(*k*+1, **U**, C, D', $Cent^{(i)}$) ▷ Algoritmo 10
13: $Cent^{(i+1)} = \emptyset$
14: $S' = \emptyset$
15: **for** $y = 1$ to k **do**
16: $cent_y^{(i+1)}$ = `average`(C_y) ▷ Algoritmo 12
17: $Cent^{(i+1)} = Cent^{(i+1)} \cup \{cent_y^{(i+1)}\}$
18: S'_y = $cent_y^{(i)} \ominus cent_y^{(i+1)}$
19: $S' = S' \cup \{S'_y\}$
20: **end for**
21: **U**,S = `UpdateU`(S') ▷ Algoritmo 9
22: **while** S $\neq 0$ **do**
23: $C = \emptyset$
24: C = `doClustering`(1, **U**, C, D', $Cent^{(i+1)}$)
25: $i = i + 1$
26: $Cent^{(i+1)} = \emptyset$
27: $S' = \emptyset$
28: **for** $y = 1$ to k **do**
29: $cent_y^{(i+1)}$ = `average` (C_y)
30: $Cent^{(i+1)} = Cent^{(i+1)} \cup \{cent_y^{(i+1)}\}$
31: S'_y = $cent_y^{(i)} \ominus cent_y^{(i+1)}$
32: $S' = S' \cup \{S'_y\}$
33: **end for**
34: **U**,S = `UpdateU`(S')
35: **end while**
36: **Exit** with C
37: **end procedure**

Una vez que todos los registros se han asignado a un clúster $C_y \in C$, los nuevos centroides $Cent^{(i+1)}$ se calculan mediante el promedio de los registros en cada C_y, haciendo uso de la suma y la multiplicación homomórfica (líneas 15 - 19). Esto se hace mediante la llamada a la función `average`(C_y) implementada a través del Algoritmo 12. Cada elemento E_j en C_y es el cifrado de r_i en D. Siendo r_i un vector de a componentes (una para cada atributo, ver definición en la sección 4.4.1), E_j también tiene a componentes, cada una siendo el cifrado de cada atributo en r_i.

Algorithm 12 Proceso de recálculo de centroides

```
1:  procedure AVERAGE(C_y)
2:      avrg' set of a elements initialised by 0
3:      for E_j ∈ C_y do                          ▷ Cada registro cifrado en el clúster
4:          for q = 1 to a do
5:              avrg'[q] = avrg'[q] ⊕ E_j[q]      ▷ Suma acumulada de cada componente en E_j
6:          end for
7:          for q = 1 to a do
8:              avrg'[q] = avrg'[q] ⊛ 1/|C_y|
9:          end for
10:     end for
11:     Exit with avrg'
12: end procedure
```

Con estos nuevos centroides $Cent^{(i+1)}$, se calcula el desplazamiento y se actualiza la matriz de distancias llamando a la función `UpdateU` (Algoritmo 9). Posteriormente, se inicia un ciclo que se repite hasta que la matriz de desplazamiento S contenga solo ceros (líneas 23 - 32). En cada iteración, se realiza el agrupamiento con los centroides actuales, se actualizan los centroides cuando el agrupamiento termina, se calcula el desplazamiento y se actualiza la matriz U.

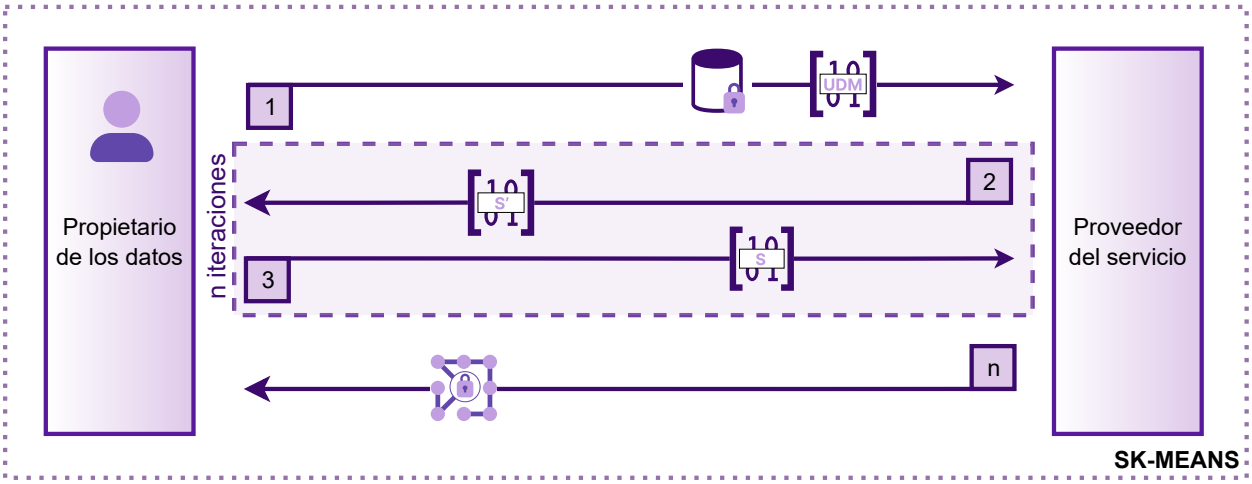

Figura 16. Interacción entre el propietario de los datos (PD) y el proveedor del servicio de minería de datos en la nube (PS) para el algoritmo Algoritmo s*k*-means.

La ejecución coordinada entre el PD y el PS, dará como resultado la ejecución de la tarea de agrupamiento con preservación de la privacidad de los datos. En la Figura 16 se describen de manera gráfica las interacciones entre estas dos entidades (PD y PS) para el caso de Sk-means. En el paso uno se requiere que el PD haga disponible sus datos seguros D' junto con la matriz de distancias U (matriz de distancias actualizables). Dicha matriz solo refleja la distancia entre cada registro del conjunto de datos con cada centroide, lo cual no revela información de los datos. En este paso, el PS recibe estos dos elementos e inicia el proceso de agrupamiento siguiendo el mismo enfoque que en k-means. PS mantiene una constante comunicación con PD para lograr la actualización de U. Una vez calculado el modelo final de los datos, este es devuelto al PD para su uso.

4.4.2. Dbs*k*-means

El acceso a los valores de distancia de los registros en claro en D y a los centroides en las iteraciones de Sk-means en el lado del PS, puede resultar en una posible filtración de información sobre el contenido de D, comprometiendo la privacidad. En [233] se propuso dbs$k$-means, una versión de Sk-means que, en lugar de usar la matriz de distancias U en el lado del PS, usa EU (Eq. 16), la versión cifrada de U. EU contiene las distancias cifradas de los registros de D a los centroides (Eq. 17). Este cifrado de U se hace con el esquema FDH-OPE descrito en la sección 4.3. EU es prácticamente igual a U, salvo que sus componentes $eu_{x,y,z}$ son el cifrado de $u_{x,y,z}$, siendo este último la distancia del registro r_x al centroide $cent_y$, respecto al atributo z. EU se genera en dos pasos: (i) cálculo de U, y (ii) cifrado de U.

$$EU = \begin{bmatrix} (eu_{1,1,1}, ..., eu_{1,1,a}) & \cdots & (eu_{1,k,1}, ..., eu_{1,k,a}) \\ \vdots & \ddots & \vdots \\ (eu_{n,,1,1}, ..., eu_{n,1,a}) & \cdots & (eu_{n,k,1}, ..., eu_{n,k,a}) \end{bmatrix} \tag{16}$$

$$eu_{x,y,z} = \text{FDH-OPE}(u_{x,y,z}) \tag{17}$$

La distancia del registro $r_x \in D$ al centroide $cent_y^{(i)}$ se determina utilizando EU de acuerdo con la ecuación 18.

$$dis(EU, r_x, cent_y^{(i)}) = \sum_{z=1}^{z=a} \mid eu_{x,y,z} \mid \tag{18}$$

Al igual que para sk-means, el proceso se inicia del lado del propietario. El flujo se presenta en el Algoritmo 13, el cual tiene como propósito la preparación de los datos para su externalización. El primer paso es hacer uso de la llave secreta SK, generada con el esquema de Liu, para cifrar cada registro del conjunto de datos D, generando un nuevo conjunto D' compuesto por estos datos cifrados. Luego, se calcula la matriz de distancias EU (previamente explicada), haciendo uso del conjunto de datos original y el parámetro a (que indica la cantidad de atributos en D), Posteriormente, cada elemento de EU se cifra utilizando el esquema FDH-OPE. Finalmente, el algoritmo devuelve el conjunto de datos cifrados D y la matriz de distancias EU.

Algorithm 13 Preparación de los datos desde el lado del PD (DBSKMEANS)

1: **procedure** OUTSOURCEDATA(D, a, m)
2: SK_m = Liu scheme secret key ▷ Seccion 4.1
3: $D' = \emptyset$
4: **for** all $r \in D$ **do**
5: $D' = D' \cup$ `Encrypt`(r, SK_m) ▷ Cifrado de Liu (Algoritmo 4)
6: **end for**
7: EU = `CalculateUDM` (D, a)
8: **for** $eu \in EU$ **do**
9: $EU = EU \cup$ `Encrypt`(eu) ▷ Cifrado FDH-OPE (Algoritmo 7)
10: **end for**
11: **Exit** with D' and EU
12: **end procedure**

La tarea de agrupamiento desde el lado del PS haciendo uso de dbs$k$-means se realiza prácticamente igual que el Algoritmo 11. La diferencia radica en el uso de EU en lugar de U, y, por tanto, en una forma diferente de realizar la actualización en cada iteración del algoritmo debido a la actualización de los centroides. La actualización de EU en lugar de U (líneas 21 y 34 del Algoritmo 11) se realiza usando el Algoritmo 14.

Algorithm 14 Proceso de actualización de U, en el lado del PS

procedure UPDATEEU(EU, S')
  $S = \emptyset$
  for $S'_y \in S'$ **do**
    S''_y = `Decrypt`(S'_y) ▷ Descifrado de Liu (Algoritmo 5)
    S_y = `Encrypt`(S''_y) ▷ Cifrado FDH-OPE (Algoritmo 7)
    $S = S \cup S_y$
  end for
  EU = $EU + S$
  Exit with EU and S
end procedure

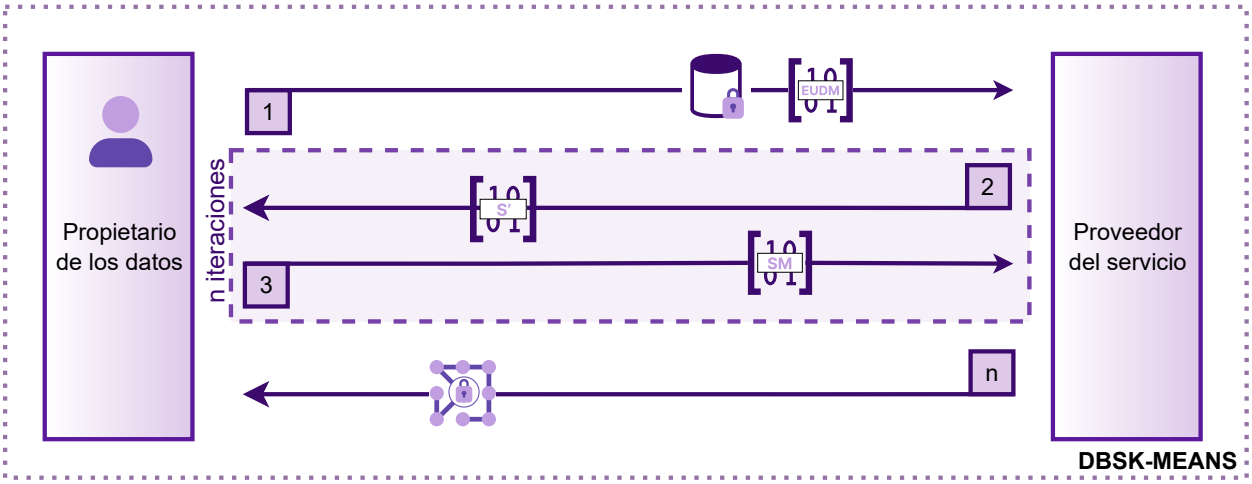

Figura 17. Interacción entre el propietario de los datos (PD) y el proveedor del servicio de minería de datos (PS) en la nube para el algoritmo dbs*k*-means.

En la Figura 17 se describen de manera gráfica las interacciones del PD con el PS de minería de datos para el caso de Dbs$k$-means.

4.4.3. Dbsnnc

El algoritmo nnc, como se explicó en la sección 2.1.1 usa el principio del vecino más cercano para realizar el agrupamiento. Se puede agrupar un número predefinido de muestras más cercanas en distancia a un elemento particular del conjunto de datos o la cantidad de muestras puede variar según la densidad local de los datos (aprendizaje del vecino basado en el radio).

Para garantizar la privacidad del algoritmo nnc, Almutari *et. al.* [17] propusieron el algoritmo dbsnnc, el cual usa también la matriz de distancias cifradas EU descrita en la sección previa con una ligera variante. Esta matriz (llamada ED) es de dimensión $n \times n$ y permite mantener las distancias a nivel de registro, no a nivel de atributo como en dbs$k$-means. La forma de una matriz ED se muestra en la ecuación 19, la cual se genera de acuerdo con la ecuación 20. A diferencia de las matrices de distancias mencionadas en los PPDM anteriores y por el propio funcionamiento del algoritmo dbsnnc, la matriz ED no requiere actualización. Es importante mencionar que ED es simétrica con respecto

a la diagonal principal, por lo que solo es necesario calcular los valores para el triángulo 2D inferior (o superior) de la matriz.

$$ED = \begin{bmatrix} ed_{1,1} & ed_{1,2} & \cdots & ed_{1,n} \\ \vdots & \cdots & \ddots & \vdots \\ ed_{n,1} & ed_{n,2} & \cdots & ed_{n,n} \end{bmatrix} \tag{19}$$

$$ed_{x,y} = \text{FDH-OPE}\left(\sum_{i=0}^{i=a}(\mid r_{x,i} - r_{y,i} \mid)\right) \tag{20}$$

Así, la distancia entre cualquier par de registros $\{r_x, r_y\} \in D$ está dada directamente por el elemento $ed_{x,y} \in ED$.

El proceso se inicia del lado del PD, con el algoritmo 15, este se encarga de hacer el cifrado del conjunto de datos D y generar la matriz de distancias ED. El proceso inicia obteniendo la clave secreta SK del esquema de Liu, para luego proceder a cifrar cada registro del conjunto de datos D mediante el cifrado de Liu. Los registros cifrados se almacenan en un nuevo conjunto D'. A continuación, se calcula una matriz de distancias ED utilizando el conjunto de datos original D y el parámetro a. Posteriormente, cada elemento de ED se cifra utilizando el cifrado FDH-OPE. Finalmente, el algoritmo regresa el conjunto de datos cifrados D' y la matriz de distancias ED.

Algorithm 15 Preparación de los datos desde el lado del PD (DBSNNC)

1: **procedure** OUTSOURCEDATA(D, a, m)
2: SK_m = Liu scheme secret key ▷ Seccion 4.1
3: $D' = \emptyset$
4: ED = Empty ED of dimension $|D| \times |D|$
5: **for** all $r \in D$ **do**
6: $D' = D' \cup \texttt{Encrypt}(r, SK_m)$ ▷ Cifrado de Liu (Algoritmo 4)
7: **end for**
8: DM = `CalculateDM` (D, a)
9: **for** $dm \in DM$ **do**
10: $ED = ED \cup \texttt{Encrypt}(dm)$ ▷ Cifrado FDH-OPE (Algoritmo 7)
11: **end for**
12: **Exit** with D' and ED
13: **end procedure**

El algoritmo 16 describe el funcionamiento de dbsnnc desde el lado del PS. Recibe como entrada: 1) el conjunto de datos cifrado D' (cifrados con el esquema de Liu, al igual que en los algoritmos sk-means y dbs$k$-means), 2) la matriz ED, y 3) σ, que representa un umbral dependiente de la aplicación, para determinar que tan cercano o no es un registro r_x de otro registro r_y.

El algoritmo maneja un conjunto de grupos $C = \{C_1, C_2, ...C_k\}$ que se van formando progresivamente. El primer grupo se forma con el primer elemento $E_1 \in D'$. El ciclo principal del algoritmo recorre todos los elementos E_y, donde $y \geq 2$ y lo compara con cada elemento E_x de cada grupo en C. Esto lo hace usando la matriz ED. E_y se agregará al grupo C_k sí $E_x \in C_k$, $ed_{x,y}$ es el mínimo y $ed_{x,y} < \mu$. Si la condición anterior no se cumple, entonces se crea un nuevo grupo en C, $C_2 = \{E_y\}$. El algoritmo se repetirá hasta que todos los registros estén asignados a un grupo.

En la Figura 18 se describen de manera gráfica las interacciones del PD con el PS de minería de datos para el caso de dbsnnc. En el paso uno se requiere que el PD haga disponible sus datos (previamente cifrados) junto con la matriz ED (previamente cifrada). En este paso el PS recibe los datos e inicia el proceso de agrupamiento haciendo uso de la matriz ED para posteriormente, en el paso dos, regresar al PD el modelo de agrupamiento final, para que el propietario lo descifre y haga uso de él.

Como se puede observar en este diagrama, al hacer uso de la matriz ED, no se requiere de la interacción entre el PD y el PS, como en el caso de sk-means y dbs$k$-means.

Algorithm 16 Algoritmo de agrupación segura (DBSNNC)

procedure DBSNNC(D', ED, μ)
  C = Set of empty clusters
  $C_1 = \{E_1\}$
  $C = C \cup \{C_1\}$
  $k = 1$
  for $y = 2$ to $|D'|$ **do**
    Find E_x in some cluster $C_m \in C$ where the $ed_{x,y}$ is the smallest
    if $ed_{x,y} < \mu$ **then**
      $C_m = C_m \cup \{E_y\}$
    else
      $k = k + 1$
      $C_k = \{E_y\}$
      $C = C \cup \{C_k\}$
    end if
  end for
  Exit with C
end procedure

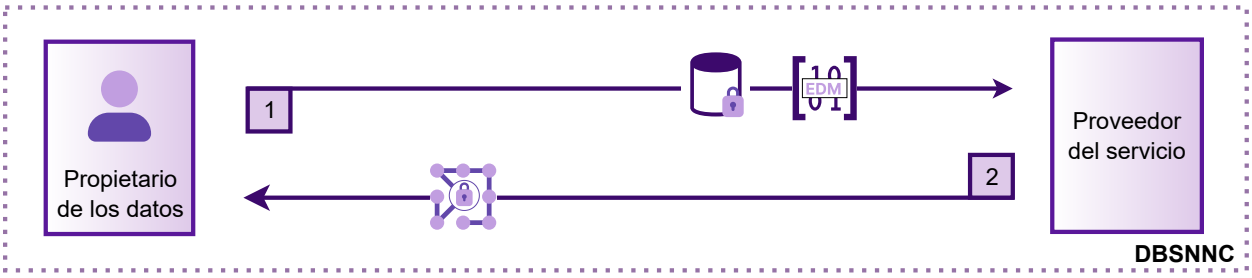

Figura 18. Interacción entre el propietario de los datos (PD) y el proveedor del servicio de minería de datos en la nube para el algoritmo dbsnnc.

4.5. PPDM para la tarea de clasificación

Sk-nn [19] es un algoritmo de clasificación con preservación de privacidad basado en el algoritmo k-nn descrito en la sección 2.1.2.

El algoritmo Sk-nn se divide en dos etapas:

1. **Etapa de entrenamiento**: Sk-nn es un algoritmo de clasificación no paramétrico, que predice la clase de una nueva instancia en función de la proximidad de esta con datos disponibles, asumiendo que es posible encontrar similitud entre dichos datos. En esta etapa se almacenan los datos del PD, cifrados con una clase previa asignada, que servirán como modelo de entrenamiento de las nuevas instancias.
2. **Etapa de predicción:** Esta etapa ocurre cuando se desea predecir las clases de nuevas instancias. El algoritmo calcula la distancia de cada instancia con respecto a todos los datos en el modelo original y asigna la clase con base en la mejor similitud.

En la etapa de predicción, sk-nn opera sobre un conjunto de datos T' que es la versión cifrada (cifrados con el

esquema de Liu, sección 4.1) del conjunto de entrenamiento en claro $T = \{[vc_1, c_1], [vc_2, c_2], ..., [vc_n, c_n]\}$. Las clases de T se mantienen en claro en T', esto es, $T' = \{[E_1, c_1], [E_2, c_2], \cdots, [Ec_n, c_n]\}$, donde cada E_i es el cifrado de vc_i.

El proceso se inicia del lado del PD con el algoritmo 17, en este se describe el proceso para cifrar un conjunto de datos T que representa el modelo. Se inicia el proceso creando la llave simétrica para el cifrado de los datos. Posteriormente, se crea un conjunto vacío T' en el que se almacenaran los datos cifrados. Haciendo uso de la llave secreta y el algoritmo de cifrado de Liu (`Encrypt`, algoritmo 4) se cifra cada registro y se agrega a T'. Finalmente, este nuevo conjunto T' se entrega como salida del algoritmo, el cual será enviado al PS.

Algorithm 17 Preparación del modelo T' desde el lado del PD

1: **procedure** OUTSOURCEDATA(T, m)
2: SK_m = Liu scheme secret key ▷ Sección 4.1
3: T' = $\emptyset$
4: **for** all $[vc_i, c_i] \in T$ **do**
5: $E_i = \texttt{Encrypt}(vc_i, SK_m)$ ▷ Cifrado de Liu (Algoritmo 4)
6: T' = $T' \cup \{[E_i, c_i]\}$
7: **end for**
8: **Exit** with T'
9: **end procedure**

El pseudocódigo que implementa sk-nn y se ejecuta del lado del PS se muestra en el Algoritmo 18. Recibe como entrada el modelo de los datos T', y la instancia a clasificar, cifrada E_x. Inicialmente, se crea una lista vacía L_δ para almacenar las distancias calculadas entre la nueva instancia E_x y cada instancia del modelo $E_i \in T'$. Se itera a través de cada instancia $E_i \in T'$ y se calcula la distancia de Manhattan entre E_i y E_x, definida mediante la Eq. 21. Esta distancia, al involucrar elementos cifrados, se calcula haciendo uso de las propiedades homomórficas del esquema de Liu.

$$manhattan\_distance(E_i, E_x) = (e_{i,1} \ominus e_{x,1}) \oplus (e_{i,2} \ominus e_{x,2}) \oplus \cdots (e_{i,n} \ominus e_{x,n}) \tag{21}$$

Estas distancias se agregan a la lista L_δ. Luego, se establece comunicación con el PD a través del algoritmo `DecryptAndMinDistance`, en el cual se realiza el descifrado del las distancias (Algoritmo 5) y se obtiene el índice de la instancia con la menor distancia. Finalmente, se ubica en L_δ la clase predicha a partir de la etiqueta de clase c_index para E_i.

Algorithm 18 Sk-NN, algoritmo de clasificación k-nn con preservación de privaciadad.

1: **procedure** SKNN(T',E_x)
2: $L_\delta = []$
3: **for** each $E_i \in T'$ **do**
4: E_δ = manhattan_distance(E_i, E_x) ▷ Ecuacion 21)
5: L_δ.append(E_δ)
6: **end for**
7: $index$ = `DecryptAndMinDistance`(L_δ) ▷ Descifrado de Liu (Agoritmo 5)
8: $c_x = c_{index}$
9: **Exit** with c_x
10: **end procedure**

En la Figura 19 se describen de manera gráfica las interacciones entre el PD y el PS para el algoritmo de sk-nn. Como se mencionó anteriormente, este proceso se divide en dos etapas: entrenamiento y predicción. En el paso uno, se aplica la etapa de entrenamiento, en esta, el PD hace disponibles sus datos (previamente cifrados) al PS. Este lo

almacena para usarlo en la etapa siguiente. En el paso dos, se inicia la etapa de predicción, en esta, el propietario envía el conjunto de datos a clasificar (previamente cifrado). Una vez recibido, se inicia el proceso de clasificación de los datos para posteriormente, en el siguiente paso (paso 3), regresar al PD un vector con las etiquetas correspondientes a los registros clasificados.

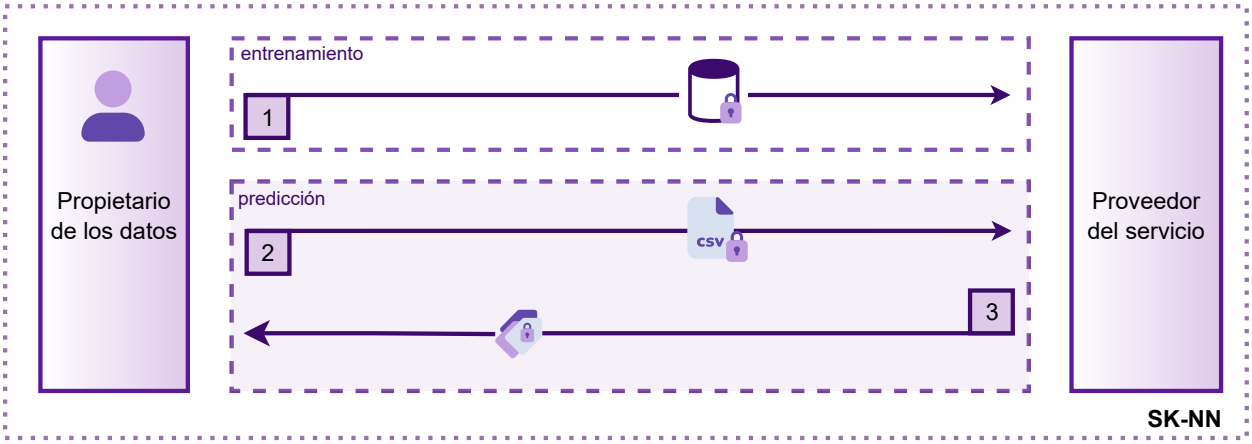

Figura 19. Interacción entre el propietario de los datos (PD) y el proveedor del servicio (PS) durante la ejecución de s*k*-nn.

5. Arquitectura PPDMaaS propuesta

Bajo un enfoque DMaaS es posible que una o varias entidades hagan disponibles sus datos a un PS para su análisis mediante un algoritmo de minería de datos, como la agrupación o la clasificación. En este caso, cuando los datos sean externalizados para su procesamiento, las garantías de privacidad por parte del propietario ya no se pueden mantener [12]. Aunque existen algoritmos de minería de datos con preservación de privacidad, como los PPDM revisados en la sección previa, se necesita una plataforma que habilite el despliegue del modelo PPDMaaS y resuelva, entre otros aspectos, los siguientes desafíos:

1. *Escalabilidad*: Esto es, que pueda soportar cargas de datos altas (en el contexto de Big Data), sin pérdida de eficiencia y usabilidad.
2. *Parametrización*: Que permita la interacción entre el PD y el PS, implicada por la operación de los mismos PPDM. Además, que la arquitectura y modelo de cómputo no sea fuertemente dependiente del PPDM usado, es decir, que no se tengan implementaciones a la medida para cada caso (sk-means, sk-nn, etc.).
3. *Flexibilidad*: que permita un despliegue simple y en cualquier infraestructura de nube.
4. *Eficiencia*: Que pueda procesar grandes cargas de trabajo implicadas por el volumen de los datos y por los niveles de seguridad empleados. Generalmente, un mayor nivel de seguridad implica más costo computacional, que aunado a una gran cantidad de datos puede resultar en un proceso DMaaS efectivo pero ineficiente y, por tanto, poco usable. La eficiencia es la característica más importante en el diseño de la plataforma PPDMaaS.

Una plataforma con las características previas no existe, y su diseño y construcción es lo que se detalla en esta sección. El diseño conceptual de la plataforma PPDMaaS es la propuesta en este trabajo y se presenta en la Figura 20,la cual está guiada por tres escenarios de trabajo:

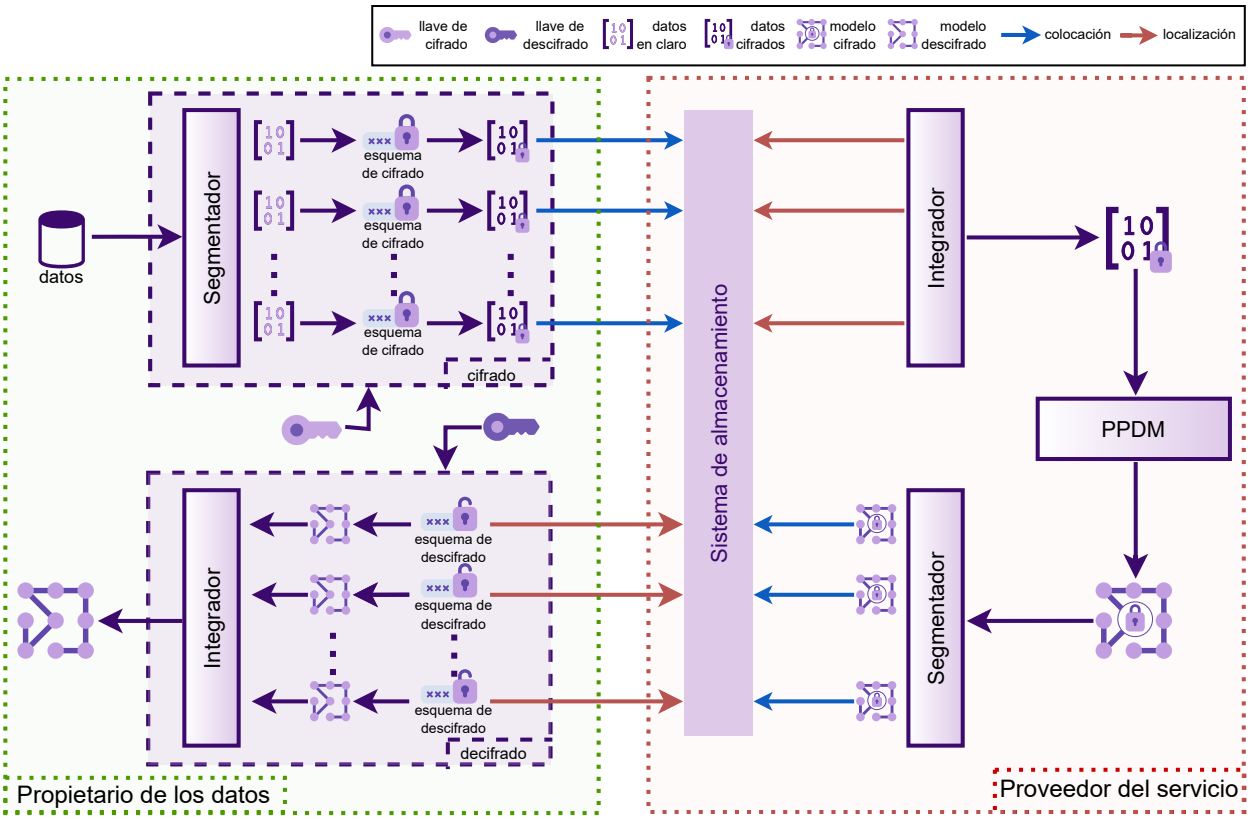

Figura 20. Representación conceptual de la solución propuesta.

1. Del lado del PD se realiza la tarea de cifrado de los datos. Debido a que las tareas de cifrado y descifrado son computacionalmente costosas, se han aplicado técnicas de paralelismo que permiten la eficiencia en la realización de ambas tareas (los detalles de esta técnica se explican en la subsección 5.3).

   El tamaño de los resultados obtenidos de la tarea de cifrado (segmentos de cifrado) generalmente será más grande que los datos en claro. En un contexto de Big Data, este volumen debe ser manejado de manera eficiente (por ejemplo, mediante segmentación de datos) por lo que se considera un sistema de almacenamiento externo, desde donde el PS pueda colocar y localizar sus datos. Dicha segmentación durante el proceso de generación de datos implica una tarea de integración durante el proceso de uso de esos datos.

2. Del lado del PS se hace la localización de segmentos de cifrado, la integración de dichos segmentos, la ejecución del PPDM de interés (agrupamiento o clasificación), la segmentación del modelo generado y finalmente la colocación de dichos segmentos en el sistema de almacenamiento para su posterior uso por parte del usuario final.

3. Del lado del PD (o del usuario final) se hace la localización de los segmentos del modelo, el descifrado de los segmentos y la integración para formar un modelo de minería de datos (agrupación o clasificación) en claro.

Así, la arquitectura PPDMaaS puede visualizarse con base en dos contextos de ejecución (ver Figura 21):

1. **Contexto local (confiable):** Este se ubica del lado del PD. Dentro de este contexto se realiza el proceso de cifrado y descifrado de los datos de manera segura, sin involucrar procesos externos que puedan comprometer los datos. Los datos seguros generados son la salida de este contexto. Dentro de este se incluyen dos actores: usuario y cliente. El usuario es el PD, este hace uso de clientes encargados de cifrar sus datos. Cabe resaltar

que puede existir n cantidad de usuarios y para cada uno de ellos es necesario desplegar un componente *client*. Este *client* es el que se comunica con el PS para realizar las interacciones implicadas por los algoritmos PPDM (descifrado en tiempo de ejecución del PPDM) a través del intercambio de metadatos.

2. **Contexto de servicio (no confiable):** Es el que se ubica del lado del PS, el cual opera en infraestructura en la nube. En este contexto se gestionan los algoritmos PPDM de interés (como el de agrupamiento o el de clasificación). Entre las características de este contexto se encuentran que es un sistema virtualizado que permite ofrecer confidencialidad y privacidad de los datos empleando esquemas de cifrado homomórfico, además de ofrecer alta disponibilidad, empleando replicación de servicios bajo el modelo de procesamiento *manager - worker*, explicado con más detalle en la sección siguiente.

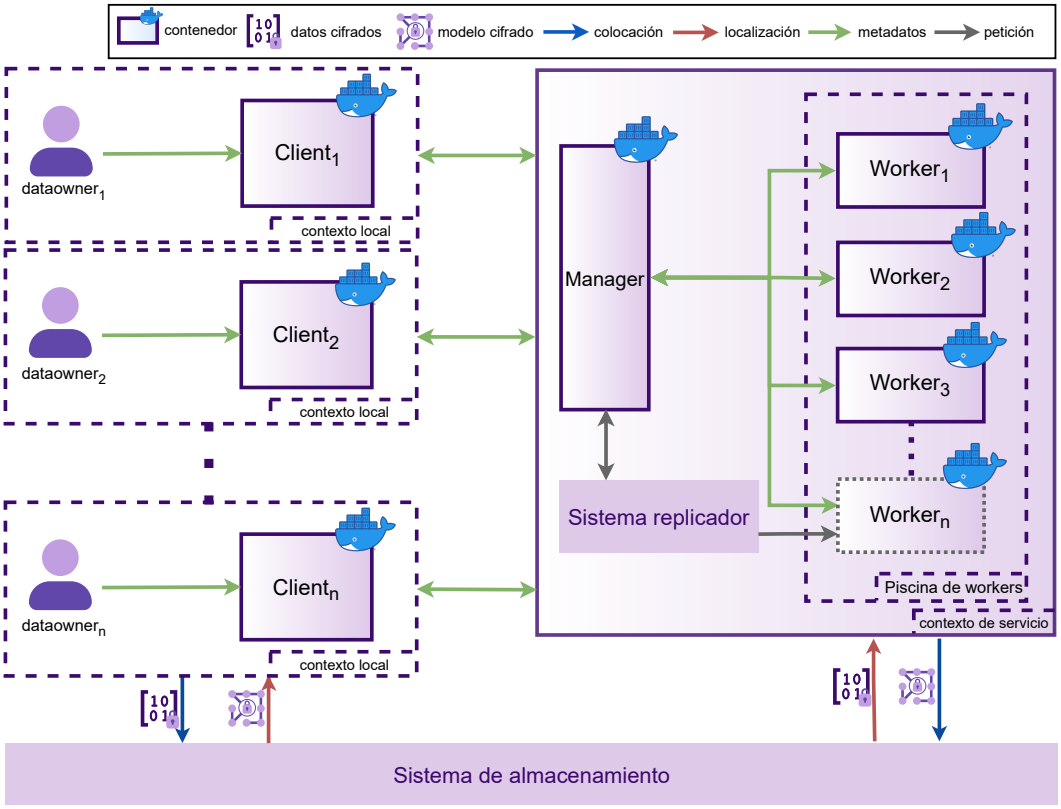

Figura 21. Representación conceptual que refleja la interacción general de múltiples usuarios con el PS y con un sistema de almacenamiento para el intercambio de datos.

Ambos contextos están relacionados por un sistema de almacenamiento externo [235] que permite la comunicación de datos masivos del PD al PS y viceversa. A partir de la Figura 21, se definen los 5 componentes principales para la arquitectura de la plataforma PPDMaaS propuesta: *client, manager, worker, sistema replicador* y *sistema de almacenamiento*.

5.1. Arquitectura de componentes

Los componentes *client, manager* y *worker* tienen una arquitectura de 3 capas (Sección 2.3.1): 1) Acceso: permite hacer la gestión de identidad y acceso. En esta capa se realiza la tarea de autenticación de usuarios a través de

credenciales. Solo en caso de ser válidas, permite el acceso a la siguiente capa. 2) Procesamiento: lleva a cabo una tarea específica, como procesos de balanceo, cifrado o segmentación. 3) Datos: permite realizar la persistencia de los datos o metadatos generados en las capas superiores/anteriores.

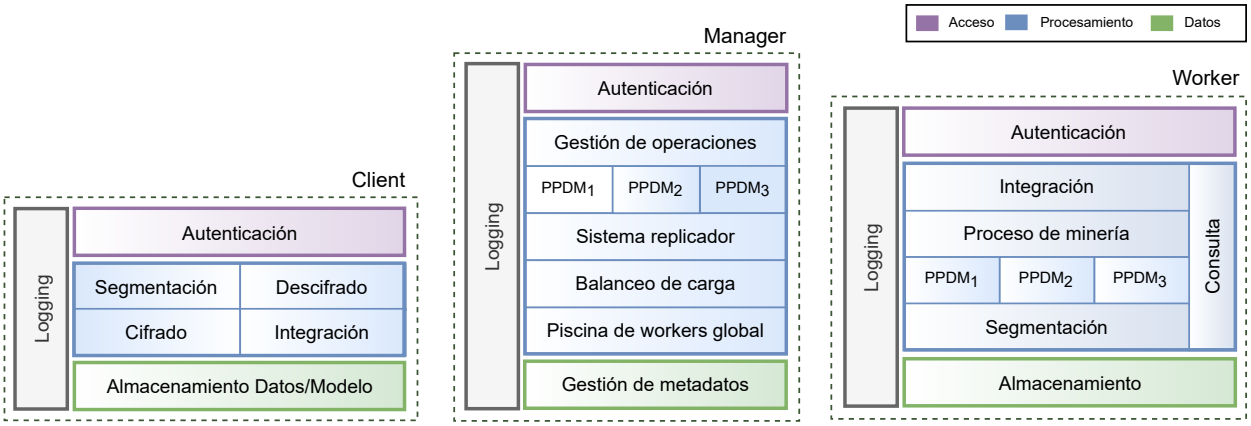

Figura 22. Arquitectura de los componentes de software.

- *Client*: Se ejecuta del lado del PD, quien cuenta con los datos que se van a externalizar. Este componente le permite preparar los datos (cifrarlos) previo a su externalización, y depende directamente del proceso de cifrado/descifrado que se esté usando (el esquema de Liu, por ejemplo, descrito en la Sección 4.1). La capa de acceso inicia con la autenticación, que permite el acceso a los recursos del componente. En la capa de procesamiento se observan dos posibles caminos, por un lado, la segmentación del conjunto de datos original y el cifrado de cada segmento, y por el otro, el descifrado y la integración del modelo. En la capa de datos se hace el almacenamiento de los datos y del modelo, para que los componentes siguientes puedan acceder a ellos.
- *Manager*: Se ejecuta del lado del PS, quien ofrece el servicio de minería de datos. La capa de acceso se inicia con la autenticación. La capa de procesamiento realiza la gestión de las operaciones enviadas por el *client*, y a través de los metadatos enviados se lleva a cabo el proceso de balanceo. Después, se hace uso de un sistema replicador, el cual es el encargado de desplegar tantos *workers* según sea necesario. Se encarga también del balanceo carga, que permite seleccionar el *worker* disponible y ubicarlo en la piscina de *workers* global. La capa de datos permite hacer la gestión de los metadatos.
- *Worker*: Representa el componente que contiene el catálogo con los PPDM disponibles. La capa de acceso inicia con la autenticación. La capa de procesamiento puede realizar el proceso de minería o simplemente una consulta de un proceso de minería realizado anteriormente. En el caso de realizar el proceso de minería, inicia con la integración de los diferentes segmentos que conforman el conjunto de datos. Posteriormente, realiza el proceso con el algoritmo elegido, sería posible elegir entre diversos algoritmos PPDM, por ejemplo, sk-means (explicado en la Subsección 4.4.1). Como producto de este proceso se obtiene un modelo, el cual pasa por un proceso de segmentación. En la capa de datos se hace el almacenamiento para que otros procesos puedan acceder a los datos.

Cabe resaltar que todos los componentes cuentan con un módulo de $logging$ transversal necesario para almacenar información del sistema que es generada en cada una de las capas de la arquitectura, por ejemplo, tiempos de respuesta, tiempos de servicio, entre otros. Estos registros se utilizan para el análisis del rendimiento del sistema.

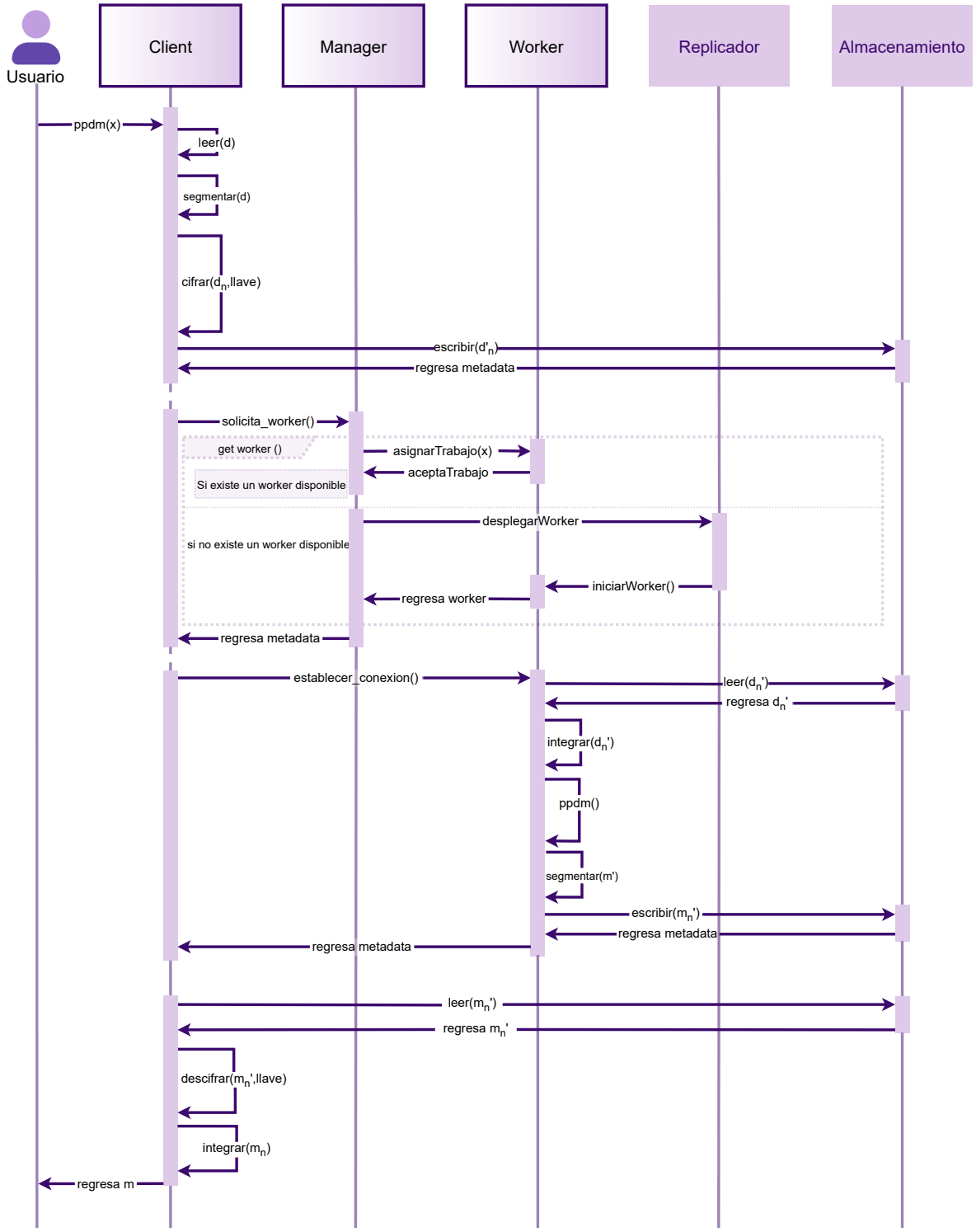

Figura 23. Diagrama de interacciones entre componentes de la plataforma.

5.2. Modelo de comunicación

La interacción coordinada de estos componentes se muestra en el diagrama de interacción en la Figura 23. El proceso inicia con el usuario o PD, que cuenta con el conjunto de datos de interés a analizar y con su llave para cifrarlos. Estos dos elementos se pasan al *client*, que es el encargado de cifrar los datos y ponerlos disponibles al PS a través del *sistema de almacenamiento*. Para ello, el *client* tiene que preparar los datos: leer el conjunto de datos original (d), segmentarlo en n partes y posteriormente cifrar cada segmento con la llave simétrica del propietario, finalmente escribir cada segmento cifrado (d'_n) en el *sistema de almacenamiento* y obtener los metadatos asociados.

Ya con los datos cifrados, el siguiente paso es establecer la conexión entre los componentes *client* y *manager* (bajo el modelo de comunicación *cliente-servidor*), que representa la conexión de los dos contextos, el local (confiable) del lado del PD (*client*), y el de servicio (no confiable) del lado del PS (*manager*).

La ejecución de un PPDM en el lado del PS la realiza un *worker*, que es creado, administrado y asociado al *client* a través del *manager* quien opera bajo un esquema de balanceo de carga, esto es, si el *worker* existe, se recupera, si no, se crea a través de una solicitud del *manager* al *sistema replicador* (también bajo el modelo *cliente-servidor*). El *manager* asocia los componentes *worker* y *client* quienes se comunica de manera directa a través de metadatos.

Una vez establecida la comunicación directa entre *client* y *worker*, el proceso de ejecución del PPDM de interés comienza. El *worker* extrae los segmentos cifrados mediante el *sistema de almacenamiento* y los integra en un solo conjunto que se hace disponible al algoritmo PPDM. Cuando el PPDM termina, el modelo resultante se segmenta y cada segmento del modelo se hace disponible al *client* a través del *almacenamiento*. Finalmente, el *client* obtiene el modelo completo en claro al extraerlo del *sistema de almacenamiento* los segmentos, descifrarlos, e integrarlos.

5.3. Patrones de procesamiento

La arquitectura PPDMaaS considera dos patrones de procesamiento como parte de su diseño: *Divide and conquer* y *Manager-worker* (Sección 2.3.2).

- **Divide and conquer**

  Se diseñó un patrón *divide and conquer* distribuido para agilizar el cifrado y descifrado de los datos, que abarca, por tanto, a los componentes *client* (en el lado del PD) y *worker* (en el lado del PS).

  Este patrón se implementa a nivel de registros, lo que permite dividir el proceso en tareas más pequeñas y distribuirlas entre los diferentes componentes involucrados (*client* y *worker*).

  Al distribuir los registros para realizar el cifrado de manera paralela se logra una mejora significativa en el tiempo de procesamiento de datos. Además, dicho patrón permite una gestión coherente y eficiente de los recursos de la plataforma en cuanto al transporte y almacenamiento, tanto de los datos, como de los modelos generados a partir del PPDM aplicado.

  En ese sentido, en la Figura 24 se observa de manera gráfica la aplicación de este patrón para el proceso de cifrado de los datos. Inicia del lado del *client*, cuando este recibe el conjunto de datos en claro y la llave de cifrado de parte del PD. El patrón realiza la segmentación del conjunto de datos en n segmentos en función del total de recursos disponibles para realizar el cifrado. Posteriormente, cada segmento cifrado se colocará en el sistema de *almacenamiento* de manera paralela. En el componente *worker* los segmentos cifrados se localizan en el sistema de *almacenamiento* y se integran para realizar el proceso PPDM.

  Para el caso del descifrado de los datos, en la Figura 25 se observa de manera gráfica el patrón *divide and conquer*. En este caso, el proceso inicia del lado del *worker*, que al contar con el modelo PPDM cifrado, lo segmenta y lo coloca en el sistema de almacenamiento. El *client* localizará estos segmentos en el sistema de almacenamiento para descifrarlos, integrarlos y así obtener el modelo final en claro.

  Uno de los principales beneficios de este enfoque es la reducción del tamaño de los datos transportados entre los componentes, lo que a su vez alivia la carga en la red al permitir el transporte de paquetes más pequeños en lugar de paquetes de gran tamaño. Además, la paralelización del proceso de cifrado y descifrado permite una mejor utilización de los recursos computacionales disponibles y reduce significativamente el tiempo necesario para llevar a cabo estas operaciones.

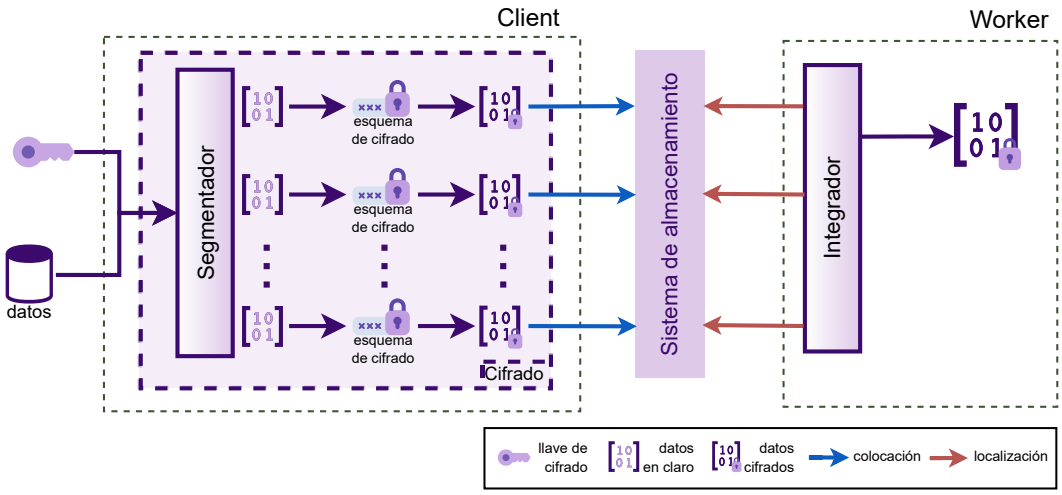

Figura 24. Diagrama conceptual del patrón *divide and conquer* para el proceso de cifrado.

En resumen, la implementación de este patrón de procesamiento *divide and conquer* distribuido proporciona una solución eficaz y eficiente para el cifrado y descifrado de datos en la plataforma PPDMaaS propuesta. Al mejorar el transporte, almacenamiento y procesamiento de los datos, se alcanza una mejora significativa en el rendimiento general del sistema, junto con una reducción en la carga de la red y el tiempo de procesamiento.

- **Manager-worker**

  El patrón de procesamiento *manager-worker* desempeña un papel fundamental en la arquitectura propuesta para la plataforma, ya que proporciona una solución eficiente para gestionar la carga de trabajo y distribuir las peticiones entre los *workers* disponibles. Este enfoque permite mejorar la eficiencia del sistema al reducir los encolamientos y garantizar una distribución uniforme de las operaciones realizadas para los PPDM.

  En conjunto con el algoritmo de balanceo de carga *Two Choices* [236], el patrón *manager-worker* asegura que la carga de trabajo se distribuya de manera equitativa entre los trabajadores disponibles, lo que contribuye a mejorar tanto la utilización de los recursos, como la capacidad de respuesta del sistema. Este enfoque a nivel de peticiones permite gestionar un mayor nivel de concurrencia, lo que significa que la plataforma puede soportar múltiples usuarios realizando peticiones simultáneas sin comprometer su rendimiento.

  La capacidad de la plataforma para integrar nuevos *workers* según sea necesario garantiza su escalabilidad en entornos dinámicos. A medida que se añaden más *workers*, la plataforma puede gestionar un mayor nivel de concurrencia, lo que se traduce en una mayor cantidad de usuarios conectados y realizando operaciones de PPDM. Esto contribuye a reducir los tiempos de respuesta percibidos por los usuarios y mejora su experiencia de uso.

  En escenarios de alta concurrencia, el sistema es capaz de gestionar dinámicamente la cantidad de *workers*, añadiendo nuevos *workers* o eliminándolos según sea necesario para garantizar un mejor uso de los recursos computacionales disponibles. Esta capacidad de adaptación garantiza que el sistema pueda mantener tiempos de respuesta aceptables incluso en momentos de alta demanda.

  En resumen, el patrón de procesamiento *manager-worker*, junto con un algoritmo de balanceo de carga eficiente, proporciona una solución robusta y escalable para gestionar la carga de trabajo en la plataforma propuesta. Este enfoque permite mejorar la eficiencia del sistema, reducir los tiempos de respuesta percibidos por los clientes y garantizar una distribución equitativa de las operaciones realizadas por los componentes de PPDM.

  La Figura 26 muestra un diagrama conceptual de este patrón, destacando el papel del *manager* en la gestión de la carga entre los *workers* disponibles y su comunicación con el *sistema replicador* para gestionar nuevos *workers* según sea necesario.

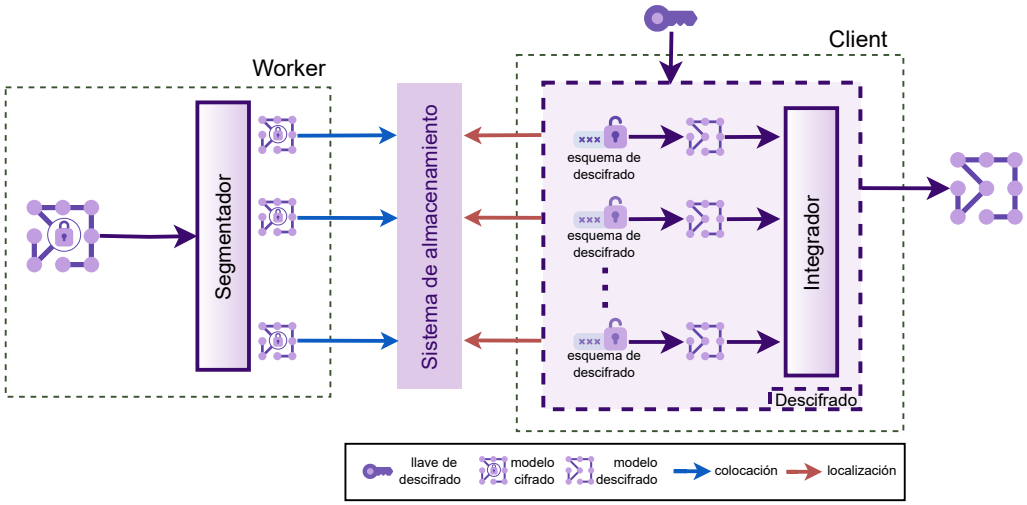

Figura 25. Diagrama conceptual del patrón *divide and conquer* para el proceso de descifrado.

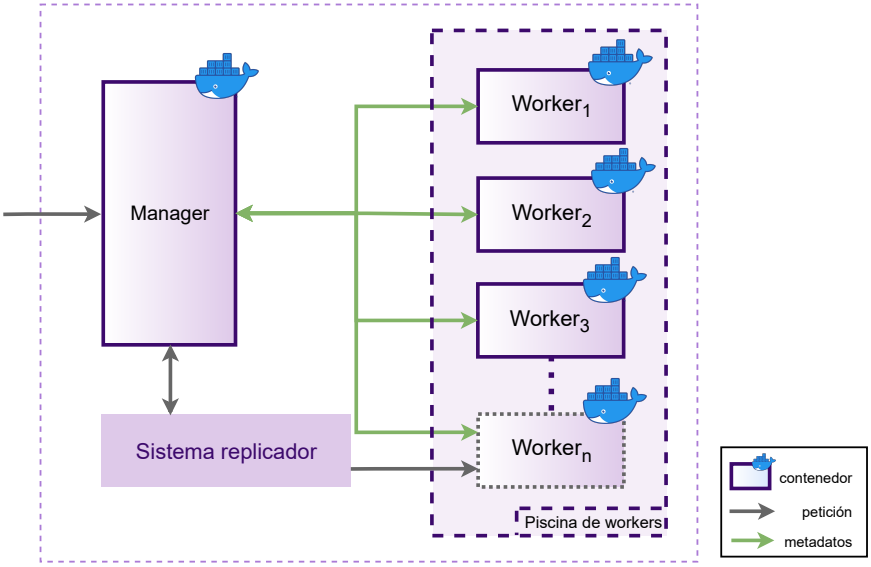

Figura 26. Diagrama conceptual del patrón *manager-worker*.

6. Experimentación y resultados

Se desarrolló un prototipo funcional en Python 3.8 basado en el diseño de la arquitectura de la plataforma PPDMaaS descrita en la sección anterior (Figura 22). Este prototipo, como prueba de concepto, permite validar y evaluar la plataforma en diversos casos de uso, incluyendo diferentes tareas de minería de datos, variados algoritmos PPDM, distintas cargas de trabajo y configuraciones de PPDM adaptadas a distintos niveles de seguridad.

En esta sección se presentan los detalles de implementación del prototipo y de los experimentos realizados. Se describen los conjunto de datos usado y las trazas, así como las métricas empleadas en la evaluación. Finalmente, se discuten los resultados obtenidos.

6.1. Escenarios de evaluación

La experimentación para validar la arquitectura PPDMaaS consiste en tres etapas:

- **Etapa 1 - Validación de la no perdida de utilidad en los datos:** En esta etapa, el objetivo es demostrar que no existe pérdida de utilidad en los datos al hacer uso de algoritmos PPDM, esto es, desde la perspectiva de la tarea de minería de datos, los resultados deberían ser los mismos usando datos en claro o los correspondientes datos cifrados. Para ello, se hace uso de métricas que permiten medir la precisión de los resultados de agrupamiento o de clasificación, según corresponda.
- **Etapa 2 - Evaluación de los esquemas de cifrado:** El cifrado de datos es una de las operaciones más costosas computacionalmente. Por eso, como se mencionó anteriormente, se han implementado técnicas de paralelismo para reducir los tiempos de respuesta. En esta etapa, el objetivo es cuantificar la reducción en el tiempo de respuesta para esta tarea utilizando el esquema de Liu (ver Sección 4.1) y el esquema FDH-OPE (ver Sección 4.3).
- **Etapa 3 - Rendimiento del sistema:** La ejecución de un PPDM implica costos adicionales debido a varias razones. Estos incluyen la sobrecarga asociada con el cifrado de los datos en el lado del PD, el uso de datos cifrados (que son más grandes que los datos en claro) en las operaciones de minería de datos, y las operaciones homomórficas realizadas en estos datos cifrados. Además, hay una interacción implícita entre el PS y el PD en los PPDM. Por lo tanto, en esta etapa, el objetivo es evaluar el rendimiento del sistema teniendo en cuenta todas estas tareas adicionales. Para lograr esto, se realiza una comparación directa entre el tiempo de respuesta de un algoritmo de minería de datos convencional y su contraparte PPDM.

6.2. Infraestructura

Para el despliegue del prototipo funcional descrito en las secciones anteriores se utilizó la siguiente infraestructura:

- *Hardware:* Durante la etapa de experimentación, se emplearon 12 equipos cuyas características se detallan en la Tabla 3. En ella se incluye información sobre el sistema operativo, la cantidad de procesadores disponibles, la memoria RAM instalada, el espacio en disco, el rol de cada equipo y el contexto para el que se utilizó cada uno. Estos equipos corresponden a servidores instalados en el Cinvestav Unidad Tamaulipas utilizados para emular al PD, al PS y al sistema de almacenamiento.

  La tabla presenta una variedad de roles, cada uno representando un componente clave en la plataforma PPDMaaS propuesta. En total, se identifican cinco componentes esenciales:

  - *Cliente* es el encargado de las tareas de cifrado de datos, y descifrado del modelo.
  - *Dataowner* diseñado para simular el comportamiento de un usuario real. Esto se logra mediante la lectura de una traza y su interpretación línea por línea.
  - *Manager* es el responsable de facilitar la comunicación entre el componente *client* y *worker*, además de gestionar el despliegue de los componentes *worker* de manera dinámica utilizando el sistema *replicador*.

Contexto	Equipo (SO)	Cores	RAM	Disco	Rol
Local	Ubuntu 22.04	16	62 GB	3145 GB	Client
Local	Ubuntu 22.04	6	12 GB	500 GB	Dataowner
Servicio	Ubuntu 22.04	4	32 GB	4243 GB	Manager
Servicio	Ubuntu 22.04	6	12 GB	917 GB	Worker
Servicio	Ubuntu 22.04	6	12 GB	1400 GB	Worker
Servicio	Ubuntu 22.04	6	12 GB	1500 GB	Worker
Servicio	Ubuntu 22.04	6	12 GB	1500 GB	Worker
Servicio	Ubuntu 22.04	12	64 GB	3600 GB	Worker
Servicio	Ubuntu 22.04	6	32 GB	1900 GB	Worker
Servicio	Ubuntu 22.04	6	24 GB	1490 GB	Worker
Almacenamiento	Ubuntu 22.04	12	64 GB	5000 GB	Nodo de almacenamiento
Almacenamiento	Ubuntu 22.04	24	256 GB	2000 GB	Nodo de almacenamiento
Cifrado	Ubuntu 22.04	48	256 GB	1000 GB	Esquemas de cifrado

Cuadro 3. Características de la infraestructura utilizada.

- *Worker* es el responsable de realizar las operaciones de agrupamiento o clasificación según la selección del usuario. Los equipos con este rol permiten formar la piscina de *workers*, los cuales son desplegados a través del componente *manager*.
- *Nodo de almacenamiento* es responsable de gestionar las operaciones de colocación y localización de los datos. Para efectos de experimentación, estos nodos se han ubicado en equipos diferentes a los que albergan los *workers*. Sin embargo, es posible colocarlos en el mismo equipo; en ese caso, las operaciones de colocación y localización se realizarían de manera local.

Es importante destacar que el último equipo listado en la tabla se dedica exclusivamente a evaluar la tarea de cifrado, utilizando los dos esquemas especificados: el Esquema de Liu y el Esquema FDH-OPE, ambos aplicando la paralelización.

- *Software:* El lenguaje de programación usado fue Python 3.8, sin recurrir a bibliotecas específicas. Se codificó el esquema de Liu, al igual que los algoritmos de PPDM, sk-means, dbs$k$-means, dbsnnc y sk-nn. Se utilizó Docker para el despliegue de los componentes dentro de contenedores virtuales.

Previo a la realización de los experimentos de evaluación, se realizaron pruebas de caja negra para validar todos los componentes de forma individual.

6.3. Descripción de los conjuntos de datos

Los datos utilizados para la etapa de experimentación se dividieron en tres tipos, de acuerdo a las etapas de evacuación descritas anteriormente, las características de cada uno de ellos se describen a continuación:

- **Etapa 1:** Se utilizó un total de 16 conjuntos de datos obtenidos del repositorio UCI [1]. En la Tabla 4 se describen estos conjuntos de datos, donde se incluyen las siguientes características:
  - **Conjunto de datos:** Indica el nombre del conjunto de datos.
  - **Registros:** Indica el número de registros totales en el conjunto de datos.
  - **Atributos:** Indica el número de atributos en el conjunto de datos.
  - **k:** Indica la cantidad de grupos presentes en ese conjunto de datos.

Conjunto de datos	Registros	Atributos	k	Entrenamiento	Prediccion
Audit Data	776	26	2	620	156
Blood Transfusion	748	4	2	598	150
Breast Cancer Coimbra	116	9	2	92	24
Breast Cancer Wisconsin	699	9	2	559	140
Breast Tissue	106	9	6	84	22
Divorce	170	54	2	136	34
Ecoli	336	7	8	268	68
Fertility	100	9	2	80	20
Glass	214	9	6	171	43
Haberman	306	3	2	244	62
Iris	150	4	3	120	30
Mammographic	961	5	2	768	193
Seeds	210	7	3	168	42
Somerville Happiness	143	6	2	114	29
Speaker Accent	329	12	6	263	66
Wholesale	440	7	2	352	88

Cuadro 4. Características de los conjuntos de datos utilizados para la etapa 1.

  - **Entrenamiento:** Indica la cantidad de registros utilizados para la etapa de entrenamiento. Este número se utiliza cuando se hace uso de un algoritmo de clasificación.
  - **Predicción:** Indica la cantidad de registros utilizados para la etapa de predicción. Este número se utiliza cuando se hace uso de un algoritmo de clasificación.

- **Etapa 2:** Se emplearon dos conjuntos de datos distintos para esta etapa, ambos generados mediante un script en $Python$ versión 3.9 utilizando la biblioteca $scipy$. El primero, usado en el esquema de Liu, cuenta con un promedio de 1,000,000 registros y 100 atributos, ocupando un tamaño de 762.93 MB y estructurado como registros $\times$ atributos. El segundo conjunto de datos, utilizado para el esquema FDH-OPE, cuenta con un promedio de 20,000 registros y 4 atributos, con un tamaño total de 11.92 GB y organizado como registros $\times$ registros $\times$ atributos.
- **Etapa 3:** Se generaron 50 conjuntos de datos mediante un script en $Python$ versión 3.9, utilizando la biblioteca $scipy$, que ofrece un módulo de estadísticas para facilitar la creación de muestras desde distintas distribuciones. Cada conjunto está compuesto por aproximadamente 1000 registros, 10 atributos y 5 grupos, generados siguiendo una distribución normal.

6.4. Configuraciones del sistema

El proceso de configuración del sistema se realiza mediante dos tablas que contienen los posibles parámetros a modificar según el tipo de algoritmo utilizado. Es importante destacar que los parámetros a configurar dependen del algoritmo a evaluar, y no todos los algoritmos requieren de todos los parámetros. Esta diferenciación permite una configuración precisa y adaptada a las necesidades específicas de cada algoritmo, asegurando así la validez y consistencia de los resultados obtenidos en los experimentos.

Esta configuración se presenta a manera de tabla, en este caso se ha dividido en dos, una dedicada a los algoritmos de agrupamiento (Tabla 5) y la otra dedicada a los algoritmos de clasificación (Tabla 6), ya que cada tipo de algoritmo requiere diferentes parámetros para su correcta configuración.

En la tabla 5 se incluyen los parámetros adecuados para la ejecución de algoritmos de agrupamiento, y varían según el tipo de algoritmo y conjunto de datos a utilizar. Estos parámetros se describen a continuación:

- **experiment_id:** Este parámetro identifica de manera única una operación específica dentro del sistema. Utiliza un formato que combina varios aspectos en una sola cadena, como el algoritmo utilizado, el conjunto de datos empleado, el número de agrupaciones (k), y el número máximo de iteraciones (max_iteration). Por ejemplo, el experiment_id "E-KMEANS-AUDITDATA-K2-MI100" indica una operación de agrupamiento con el algoritmo k-means, utilizando el conjunto de datos "audit_data", con 2 agrupaciones y un máximo de 100 iteraciones.
- **algoritmo:** Este parámetro indica el algoritmo que se utilizará en la operación. Puede ser un algoritmo PPDM, como sk-means, dbs$k$-means o dbsnnc, o un algoritmo de minería de datos estándar, como k-means o nnc.
- **dataset_id:** Representa el identificador único del conjunto de datos utilizado en la operación. Este identificador corresponde a un conjunto de datos específico que se ha predefinido y se encuentra disponible en el sistema. La tabla 4 proporciona detalles sobre los conjuntos de datos disponibles.
- **level:** Este parámetro es necesario solo cuando se trabaja con un PPDM, ya que permite definir el tamaño en bits de la llave secreta utilizada en el cifrado de los datos del propietario. Para los experimentos realizados, se ha mantenido constante en 128 bits, que se considera un nivel de seguridad aceptable actualmente.
- **k:** Indica la cantidad de agrupaciones esperadas para un conjunto de datos particular. Este valor depende directamente de las características del dataset utilizado. No es necesario para todos los algoritmos de la plataforma, por lo que cuando no es relevante, se asigna un valor nulo.
- **threshold:** Representa el umbral esperado para ciertos algoritmos, como nnc y dbsnnc. Este parámetro determina la sensibilidad del algoritmo a la hora de realizar clasificaciones o agrupaciones. Cuando no es necesario, se asigna un valor nulo.
- **max_iteration:** Indica el número máximo de iteraciones que un algoritmo puede realizar antes de detenerse. Es necesario para algoritmos que pueden entrar en ciclos, como los basados en k-means.
- **interarribo:** Define el tiempo entre la llegada de una petición y la siguiente a la plataforma. Este parámetro es importante para simular escenarios realistas de llegada de peticiones al sistema y evaluar su rendimiento bajo diferentes cargas de trabajo. En esta experimentación, los intervalos entre arribos empleados siguen una distribución exponencial.

experiment_id	algoritmo	dataset_id	k	level	max_iteration	threshold	interarribo
E-KMEANS-AUDITDATA-K2-MI100	KMEANS	audit_data	2	-	100	-	1.015738
E-KMEANS-DIVORCE-K2-MI50	KMEANS	divorce	2	-	50	-	0.767572
E-KMEANS-ECOLI-K8-MI100	KMEANS	ecoli	8	-	100	-	2.455259
E-SKMEANS-AUDITDATA-K2-L128-MI100	SKMEANS	audit_data	2	128	100	-	2.007676
E-SKMEANS-DIVORCE-K2-L128-MI50	SKMEANS	divorce	2	128	50	-	1.267885
E-SKMEANS-ECOLI-K8-L128-MI100	SKMEANS	ecoli	8	128	100	-	0.381355
E-DBSKMEANS-AUDITDATA-K2-L128-MI100	DBSKMEANS	audit_data	2	128	100	-	2.724852
E-DBSKMEANS-DIVORCE-K2-L128-MI50	DBSKMEANS	divorce	2	128	50	-	2.209975
E-DBSKMEANS-ECOLI-K8-L128-MI100	DBSKMEANS	ecoli	8	128	100	-	5.959165
E-NNC-AUDITDATA-TRESHOLD05	NNC	audit_data	-	-	-	0.5	2.831309
E-NNC-DIVORCE-THRESHOLD07	NNC	divorce	-	-	-	0.7	0.858099
E-NNC-ECOLI-THRESHOLD03	NNC	ecoli	-	-	-	0.3	1.114741
E-DBSNNC-AUDITDATA-L128-TRESHOLD05	DBSNNC	audit_data	-	128	-	0.5	0.413974
E-DBSNNC-DIVORCE-L128-THRESHOLD07	DBSNNC	divorce	-	128	-	0.7	0.183545
E-DBSNNC-ECOLI-L128-THRESHOLD03	DBSNNC	ecoli	-	128	-	0.3	1.319486

Cuadro 5. Parámetros para la configuración de los algoritmos de agrupamiento

Por otro lado, la tabla 6 presenta los parámetros específicos para los algoritmos de clasificación y se utilizan para definir el entorno de experimentación, incluyendo el conjunto de datos y los modelos a evaluar. Aquí explicamos cada uno de ellos con mayor detalle:

- **experiment_id:** Este parámetro proporciona una identificación única para cada operación realizada en el sistema. Se compone de varios elementos combinados en una sola cadena, lo que permite una identificación clara y concisa de la operación en cuestión. Por ejemplo, el experiment_id "E-SKNN-AUDITDATAMODEL-AUDITDATA-L128" indica una operación utilizando el algoritmo k-nn para un modelo de datos llamado "auditdatamodel", utilizando el conjunto de datos "auditdata", con un nivel de seguridad de 128 bits.
- **algoritmo:** Indica el algoritmo de clasificación que se utilizará en la operación. Puede ser un PPDM, como sk-nn, o un algoritmo de minería estándar, como k-nn.
- **model_id:** Este parámetro representa el ID único del modelo de datos que se utilizará en la operación. El modelo de datos contiene datos etiquetados y se utiliza para la predicción de etiquetas en el conjunto de datos indicado en el siguiente parámetro.
- **dataset_id:** Es el ID único del conjunto de datos al que se le predecirán las etiquetas utilizando el modelo de datos mencionado anteriormente. Este conjunto de datos puede tener diferentes características que se detallan en la tabla 4.
- **level:** Este parámetro determina el tamaño de la llave secreta en bits utilizada por el PD para cifrar y descifrar los datos. Para los experimentos realizados, se ha mantenido constante en 128 bits.
- **interarribo:** Define el tiempo entre la llegada de una petición y la siguiente al sistema. Un tiempo de interarribo más corto indica una mayor frecuencia de llegada de peticiones al sistema, mientras que un tiempo más largo indica una frecuencia más baja. Para efectos de esta experimentación, los interarribos utilizados tienen una distribución exponencial.

Estos parámetros son cruciales para configurar y adaptar el sistema a las necesidades específicas de cada experimento, garantizando la coherencia y la reproducibilidad de los resultados obtenidos.

experiment_id	algoritmo	model_id	dataset_id	level	intearribo
E-KNN-AUDITDATAMODEL-AUDITDATA	KNN	audit_data_model	audit_data	-	1.015738
E-KNN-DIVORCEMODEL-DIVORCE	KNN	divorce_model	divorce	-	0.767572
E-KNN-ECOLIMODEL-ECOLI	KNN	ecoli_model	ecoli	-	2.455259
E-SKNN-AUDITDATAMODEL-AUDITDATA-L128	SKNN	audit_data_model	audit_data	128	2.007676
E-SKNN-DIVORCEMODEL-DIVORCE-L128	SKNN	divorce_model	divorce	128	1.267885
E-SKNN-ECOLIMODEL-ECOLI-L128	SKNN	ecoli_model	ecoli	128	0.381355

Cuadro 6. Parametros de configuracion para los algoritmos de clasificacion.

6.5. Métricas

Para evaluar los algoritmos implementados se aplicaron diferentes métricas dependiendo del tipo de tarea realizada, y de la etapa en la que se encuentran.

- **Etapa 1:** En esta etapa se hace el cálculo de la precisión de los algoritmos aplicados. Las métricas elegidas para esta etapa dependen de la tarea de minería de datos elegida:

  - **Agrupamiento:** En esta etapa de experimentación, se emplearán métricas fundamentales para medir la calidad de los agrupamientos obtenidos. Dos de las métricas más destacadas que guiarán nuestra evaluación son el Coeficiente de Silhouette y el Índice de Davies-Bouldin.

    El Coeficiente de Silhouette proporciona una medida intuitiva de cuán bien definidos están los clústeres encontrados por el algoritmo. Esta métrica asigna a cada punto de datos un valor que refleja la similitud

del clúster respecto a los clústeres vecinos, permitiendo evaluar la coherencia interna de los agrupamientos. En la ecuación 22 se muestra la fórmula para hacer el cálculo de este coeficiente:

$$Sil(i) = \frac{b(i) - a(i)}{max\{b(i), a(i)\}} \tag{22}$$

donde:

- $Sil(i)$ es el coeficiente de Silhouette para el punto de datos i
- $a(i)$ es la distancia promedio entre el punto de datos i y los otros puntos en el mismo grupo (cohesión).
- $b(i)$ es la distancia promedio entre el punto de datos i y los puntos en el grupo más cercano diferente al que pertenece i (separación).

El coeficiente de Silhouette varía entre -1 y 1. Un valor cercano a 1 indica que el punto está bien clasificado, mientras que un valor cercano a -1 indica que el punto podría haberse clasificado mejor en el grupo opuesto. Un valor cercano a 0 indica que el punto está cerca del límite entre dos grupos. En general, cuanto mayor sea el valor del coeficiente de Silhouette, mejor será la calidad de la agrupación.

Por otro lado, el Índice de Davies-Bouldin se centra en la dispersión entre los clústeres, evaluando la relación entre la distancia intra-cluster y la distancia inter-cluster. Este índice ofrece una perspectiva valiosa sobre la separación de los clústeres, contribuyendo así a la identificación de agrupamientos más definidos y significativos.

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{j \neq i} \left(\frac{\text{Dispersión intra-cluster de } C_i + \text{Dispersión intra-cluster de } C_j}{\text{Distancia entre centroides de } C_i \text{ y } C_j} \right) \tag{23}$$

donde:

- n es el número total de clusters.
- C_i y C_j son clusters diferentes.
- *Dispersion intra-cluster* es una medida de qué tan compacto es un cluster.
- *Distancia entre centroides* es la distancia euclidiana entre los centroides de dos clusters.

La combinación de estas métricas permitirá una evaluación exhaustiva de la calidad de los agrupamientos generados por el algoritmo, brindando una visión general de su desempeño en términos de coherencia interna y separación entre clústeres.

- **Clasificación:** En esta fase de experimentación, se incluyen métricas clave que proporcionan una evaluación de la predicción de los algoritmos implementados. Las métricas principales que guían la evaluación son: precisión, exactitud y recall.

  La exactitud, por su parte, cuantifica la proporción de instancias correctamente clasificadas respecto al total de instancias. Esta métrica ofrece una visión general de la efectividad global del modelo, considerando tanto los aciertos como los errores en la clasificación.

  La precisión es una medida que evalúa la proporción de instancias correctamente clasificadas entre todas las instancias predichas como positivas. Esta métrica proporciona información sobre la fiabilidad del modelo en la identificación precisa de casos positivos, contribuyendo así a la comprensión de la calidad general de las predicciones.

  El recall también conocido como sensibilidad o tasa de verdaderos positivos, mide la proporción de instancias positivas que fueron correctamente identificadas por el modelo en relación con el total de instancias positivas reales. Es especialmente útil cuando es crucial minimizar los falsos negativos, es decir, cuando la omisión de casos positivos es costosa.

  La aplicación conjunta de estas métricas permitirá una evaluación exhaustiva del rendimiento de los algoritmos de clasificación. La precisión y la exactitud ofrecerán perspectivas sobre la capacidad de clasificación general, mientras que el recall proporcionará información valiosa sobre la capacidad del modelo para capturar correctamente casos positivos.

- **Etapa 2:** Durante esta etapa, se evalúa el tiempo total que el sistema requiere para cifrar los datos de manera paralela. Para calcular este tiempo, utilizamos la fórmula:

$$t_e = t_s + \max(t_{es}) \tag{24}$$

Aquí,t_s representa el tiempo necesario para segmentar el conjunto de datos original, mientras que t_{es} indica el tiempo que toma cifrar un segmento. Se selecciona el tiempo máximo de cifrado entre todos los segmentos para asegurar que se contempla el escenario más lento en el proceso paralelo.

- **Etapa 3:** En esta etapa, se calcula el rendimiento del sistema midiendo el tiempo total que toma procesar una solicitud, desde el cifrado de datos hasta el descifrado del modelo obtenido. El tiempo de respuesta se calcula usando la fórmula:

$$t_{respuesta} = ts_{cliente(cifrado)} + ts_{manager} + ts_{worker} + ts_{cliente(descifrado)} + \lambda \tag{25}$$

Donde $ts_{cliente(cifrado)}$ es el tiempo que el cliente toma para cifrar los datos, $ts_{manager}$ es el tiempo que el manager necesita para balancear la carga entre los trabajadores disponibles, ts_{worker} es el tiempo de procesamiento del trabajador asignado a la tarea de minería de datos, $ts_{cliente(descifrado)}$ es el tiempo que el cliente tarda en descifrar el modelo resultante, y λ es la latencia entre los componentes del sistema.

Los tiempos individuales se calculan según las fórmulas detalladas a continuación:

- Tiempo de cifrado del cliente:

$$ts_{cliente(cifrado)} = t_{segmentacionD} + \max(t_{cifrado} + t_{almacenamiento}) \tag{26}$$

Aquí, $t_{segmentacionD}$ es el tiempo de segmentación de los datos originales, $t_{cifrado}$ el tiempo de cifrado por segmento, y $t_{almacenamiento}$ el tiempo de almacenamiento de cada segmento cifrado.

- Tiempo de gestión del manager:

$$ts_{manager} = t_{gestion} + t_{balanceo} + t_{replicacion} + t_{gestionmetadatos} \tag{27}$$

Donde $t_{gestion}$ es el tiempo de manejo de operaciones, $t_{balanceo}$ el tiempo para balancear las peticiones recibidas, $t_{replicacion}$ el tiempo que tarda el componente en hacer la replicación un nodo, y $t_{gestionmetadatos}$ el tiempo de manejo de metadatos.

- Tiempo de procesamiento del trabajador:

$$ts_{worker} = t_{localizacion} + t_{integracionD'} + t_{procesamientoPPDM} + t_{segmentacionM} + t_{almacenamiento} \tag{28}$$

Donde $t_{localizacion}$ es el tiempo que tarda el componente en hacer la extracción de los segmentos desde el sistema de almacenamiento, $t_{integracionD'}$ representa el tiempo que tarda el componente en hacer la integración de los segmentos del conjunto de datos cifrado, $t_{procesamientoPPDM}$ es el tiempo que tarda en procesar el algoritmo PPDM seleccionado, $t_{segmentacionM}$ indica el tiempo tardado en hacer la segmentación del modelo final y $t_{almacenamiento}$ es el tiempo que se necesita para hacer la escritura del segmento en el sistema de almacenamiento.

- Tiempo de descifrado del cliente:

$$ts_{cliente(descifrado)} = t_{localizacion} + t_{integracionM} + t_{descifrado} + t_{almacenamiento} \tag{29}$$

Aquí, $t_{localizacion}$ y $t_{integracionM}$ se refieren a la extracción e integración de los segmentos del modelo respectivamente, $t_{descifrado}$ es el tiempo de descifrado del modelo, y $t_{almacenamiento}$ el tiempo para almacenar el modelo descifrado.

7. Resultados

En esta sección se muestran los resultados obtenidos para las etapas planteadas anteriormente. Se realizaron 31 pruebas para cada una de las etapas y se aplicaron las métricas mencionadas en la sección anterior. A continuación se describen los resultados obtenidos para cada una de las etapas:

7.1. Etapa 1 - Algoritmos de agrupamiento

En esta subsección se describen los resultados obtenidos al aplicar las métricas que permiten medir la precisión entre los algoritmos de agrupamiento aplicados.

En la Figura 27 se muestran los resultados del *coeficiente de Silhouette* para los conjuntos de datos correspondientes a la etapa 1 para el algoritmo de agrupamiento k-means y dos versiones seguras del mismo algoritmo, sk-means y dbs$k$-means. En la Figura 28 se muestran los resultados del *coeficiente de Silhouette* para los conjuntos de datos correspondientes a la etapa 1 para el algoritmo de agrupamiento nnc y su versión segura dbsnnc. Para ambas gráficas, en el eje x se muestran los conjuntos de datos utilizados, y en el eje y se muestran los resultados del *coeficiente de Silhouette* obtenido para cada algoritmo.

Este *coeficiente de Silhouette* se evalúa entre -1 y 1, donde el valor 1 indica una buena agrupación de los datos.

Como podemos ver en las dos gráficas, tanto en la versión estándar como en las versiones seguras de los algoritmos, se observa un comportamiento similar en términos de los resultados del *coeficiente de Silhouette* para los diferentes conjuntos de datos evaluados. Este patrón sugiere que las versiones seguras de los algoritmos mantienen un desempeño consistente en comparación con las versiones estándar, lo que indica que no existe perdida de utilidad en los datos al hacer uso de las versiones seguras.

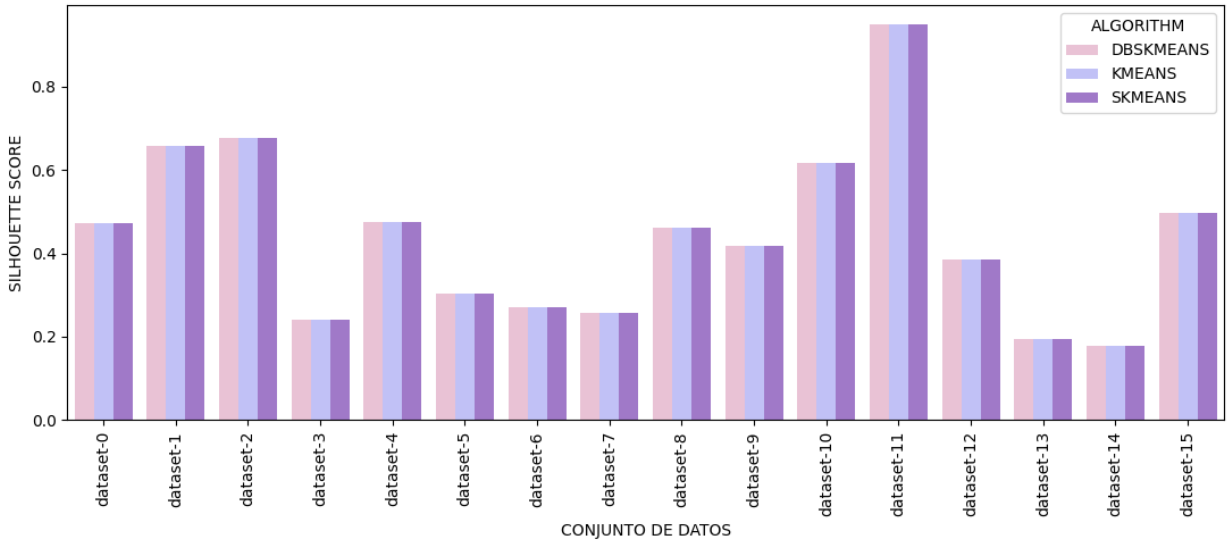

Figura 27. Gráfica - coeficiente de Silhouette para los algoritmos *k*-means, s*k*-means y dbs*k*-means.

En la Figura 29 se muestra el *índice de Davies-Bouldin* obtenido, en la evaluación de los conjuntos de datos correspondientes a la etapa 1 para los algoritmos de agrupamiento k-means y dos versiones seguras del mismo algoritmo, sk-means y dbs$k$-means. En la Figura 30 se muestra el *índice de Davies-Bouldin* obtenido para el algoritmo nnc y su versión segura dbsnnc. En ambas graficas, en el eje x se detallan los diferentes conjuntos de datos utilizados, mientras que en el eje y se representan los valores del *índice de Davies-Bouldin* para cada algoritmo.

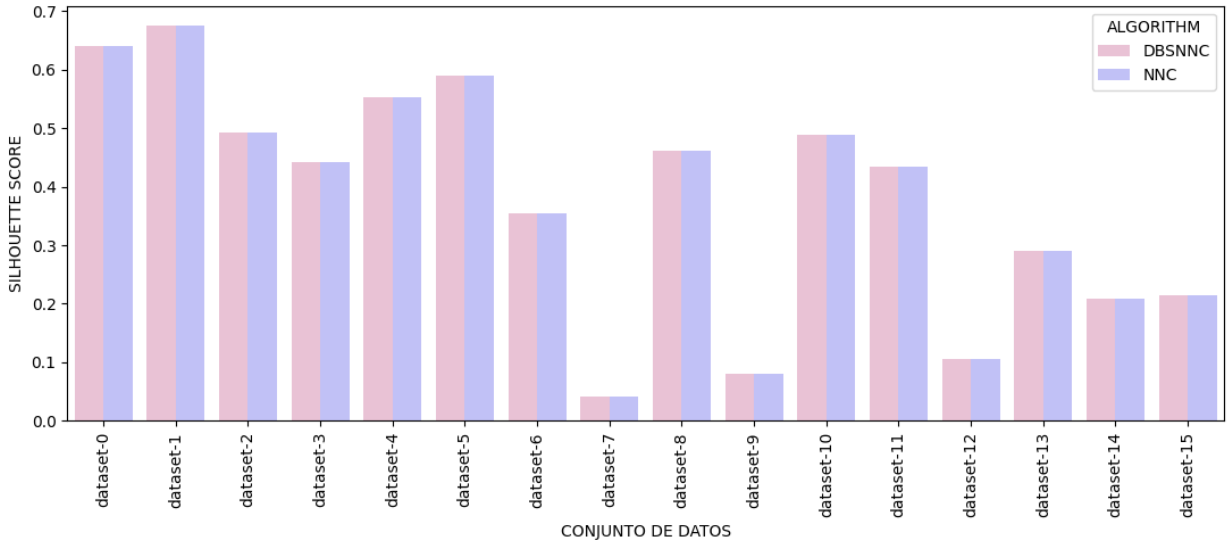

Figura 28. Gráfica del coeficiente de Silhouette para los algoritmos nnc y dbsnnc.

En estas gráficas, al igual que las gráficas obtenidas para el *coeficiente de Silhouette*, se observa un comportamiento similar en términos de los resultados del *índice de Davies-Bouldin* en los distintos conjuntos de datos, tanto en la versión estándar como en las versiones seguras de los algoritmos.

Aquí, podemos observar que el *índice de Davies-Bouldin* en ambas gráficas se mantiene, en la mayoría de los casos, en valores cercanos a 0, lo cual indica un buen agrupamiento. Además, se nota que los resultados son consistentes en ambas gráficas para cada PPDM y su respectiva versión estándar. Por lo tanto, al igual que con la métrica anterior, podemos concluir que no hay pérdida de utilidad en los datos al usar el algoritmo seguro.

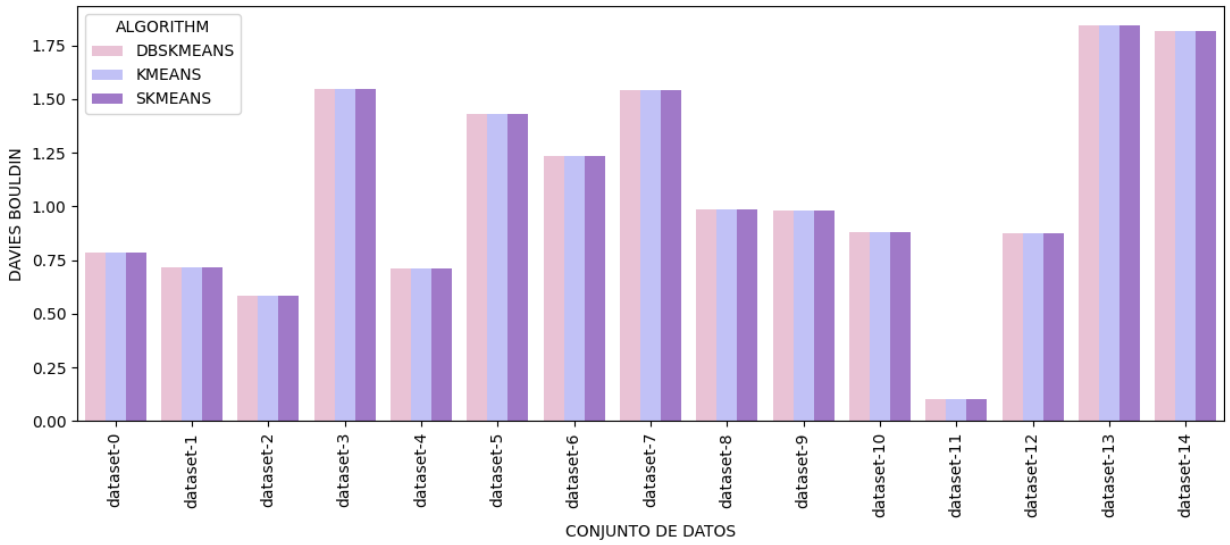

Figura 29. Gráfica del Indice de Davies-Bouldin para los algoritmos k-means, sk-means y dbs$k$-means.

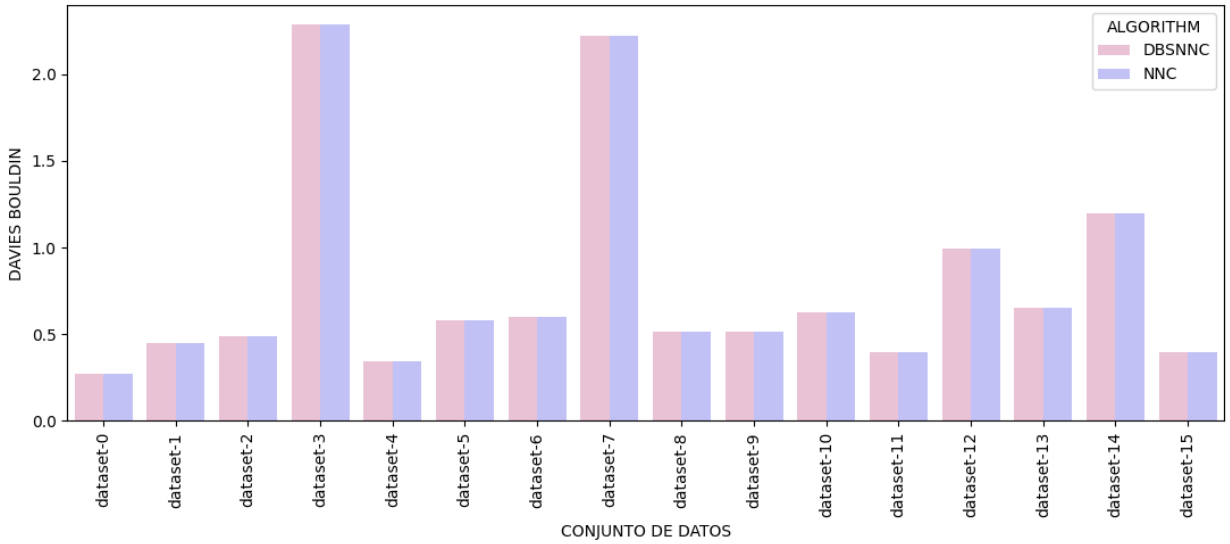

Figura 30. Gráfica del Indice de Davies-Bouldin para los algoritmos nnc y dbsnnc.

7.2. Etapa 1 - Algoritmos de clasificación

En esta subsección se describen los resultados obtenidos al aplicar las métricas que permiten medir la precisión entre los algoritmos de clasificación aplicados. Dichas métricas han sido explicadas en la sección anterior.

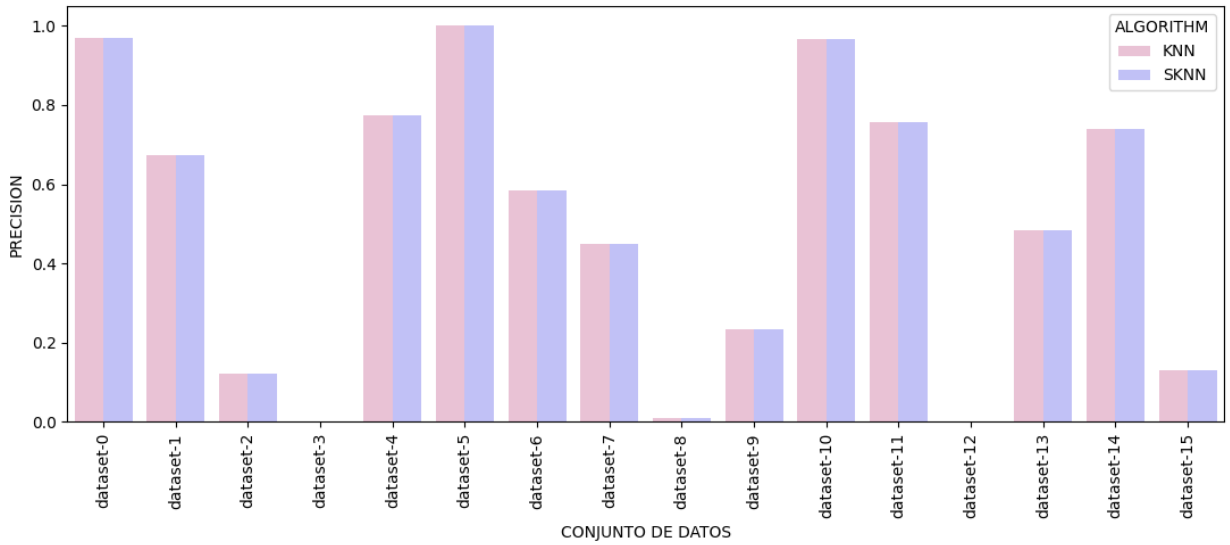

Figura 31. Gráfica de precisión para los algoritmos k-nn y sk-nn.

En la Figura 31 se muestran los resultados obtenidos para la métrica de precisión, en la evaluación de los conjuntos de datos correspondientes en la etapa 1 para los algoritmos de clasificación. En el eje x de dicha gráfica se muestran los conjuntos de datos utilizados, y en el eje y se muestran los resultados para la precisión obtenida en cada algoritmo. En esta gráfica se evaluaron los algoritmos k-nn y sk-nn. En esta gráfica se puede observar que los dos algoritmos tienen

una precisión similar para cada conjunto de datos, por lo que no existe perdida de utilidad en los datos al usar datos cifrados, como es el caso del algoritmo seguro.

En la Figura 32 se pueden observar los resultados obtenidos para la exactitud de los dos algoritmos evaluados. Al igual que en la gráfica anterior, en el eje x se muestran los conjuntos de datos utilizados, y en el eje y los resultados obtenidos. En esta gráfica podemos ver que los dos algoritmos tienen una exactitud similar para ambos algoritmos, por lo que al igual que la gráfica anterior, se puede concluir que usar un algoritmo seguro no afecta la exactitud del modelo obtenido con un algoritmo seguro.

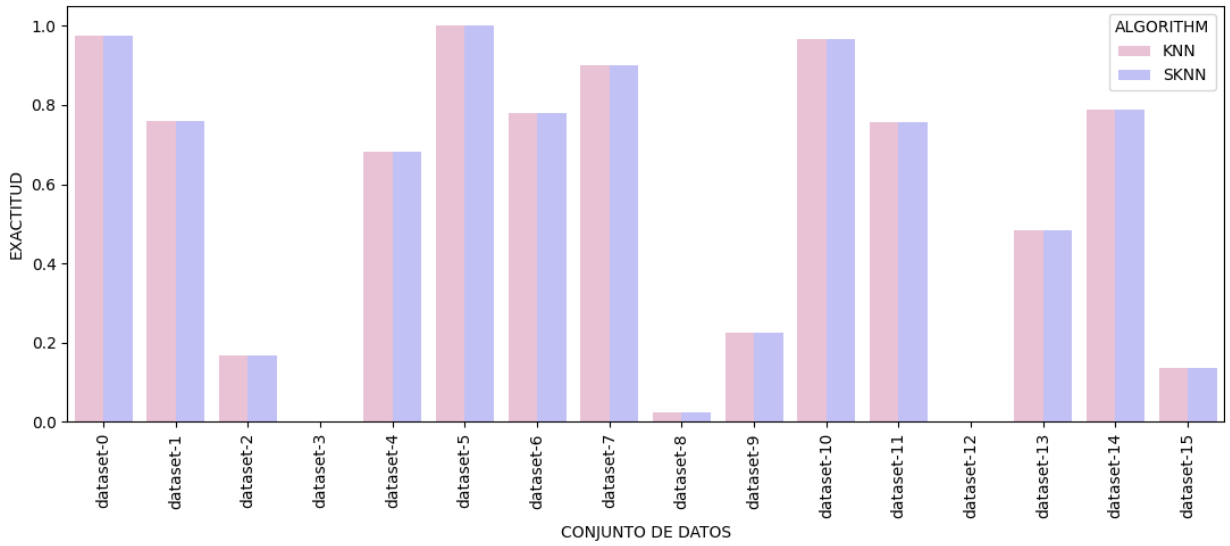

Figura 32. Gráfica de exactitud para los algoritmos *k*-nn y s*k*-nn.

En la Figura 33 se pueden ver los resultados obtenidos para el recall de los dos algoritmos evaluados. En el eje x se muestran los conjuntos de datos utilizados y en el eje y los resultados obtenidos para el recall. En esta gráfica se puede observar como los dos algoritmos evaluados presentan el mismo recall, por lo que, al igual que las gráficas anteriores, se confirma que el uso de un algoritmo PPDM no afecta los resultados finales obtenidos.

7.3. Etapa 2 - Tiempo de cifrado (del lado del PD)

Como se mencionó anteriormente, se implementó una técnica de paralelismo que permite agilizar la tarea de cifrado, la cual se ha demostrado que es una de las tareas más lentas. Se evaluaron dos de los algoritmos de cifrado implementados, el esquema de Liu y el esquema FDH-OPE.

En la Figura 34 se muestra el tiempo de respuesta para la tarea de cifrado de los datos con el esquema de Liu, evaluando los tres niveles de seguridad aceptados actualmente (128, 192 y 256 bits). En el eje x se muestra la cantidad de *workers* utilizados, y en el eje y el tiempo de cifrado en segundos. Para obtenerla se evaluó un conjunto de datos con 1,000,000 registros y 100 atributos, con un total de 31 repeticiones. Se aumentó la cantidad de *workers* disponibles entre 1, 2, 4, 8, 16 y 32.

En esta gráfica se puede observar como al aumentar la cantidad de *workers* disponibles, disminuye el tiempo de cifrado del conjunto de datos de manera exponencial. Cabe resaltar que los conjuntos de datos a cifrar tienen la forma $r \times a$, donde r representa la cantidad de registros y a la cantidad de atributos.

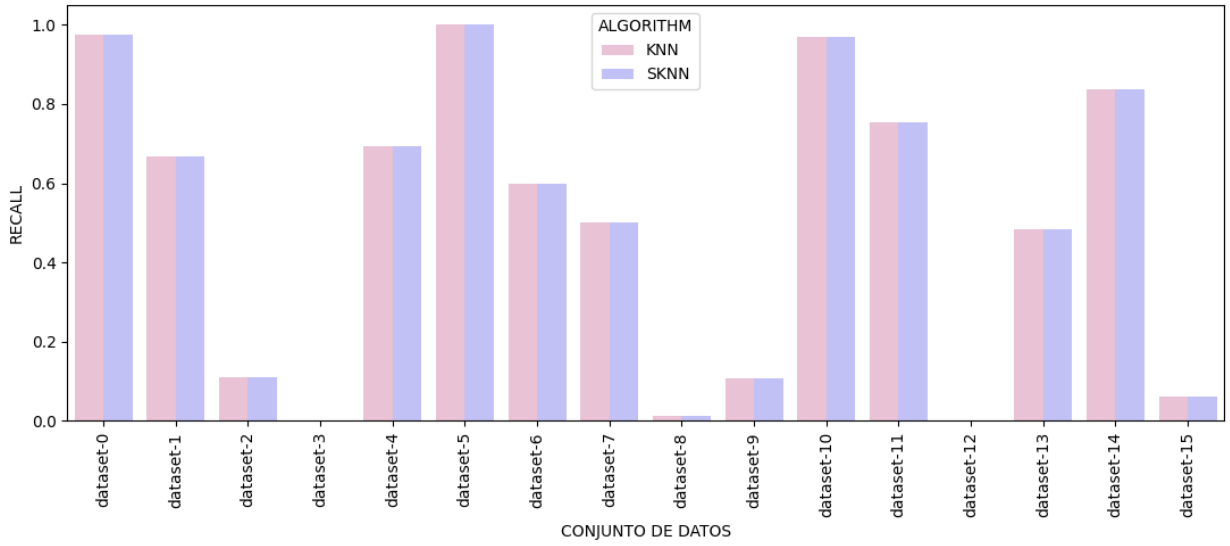

Figura 33. Gráfica de recall para los algoritmos k-nn y sk-nn.

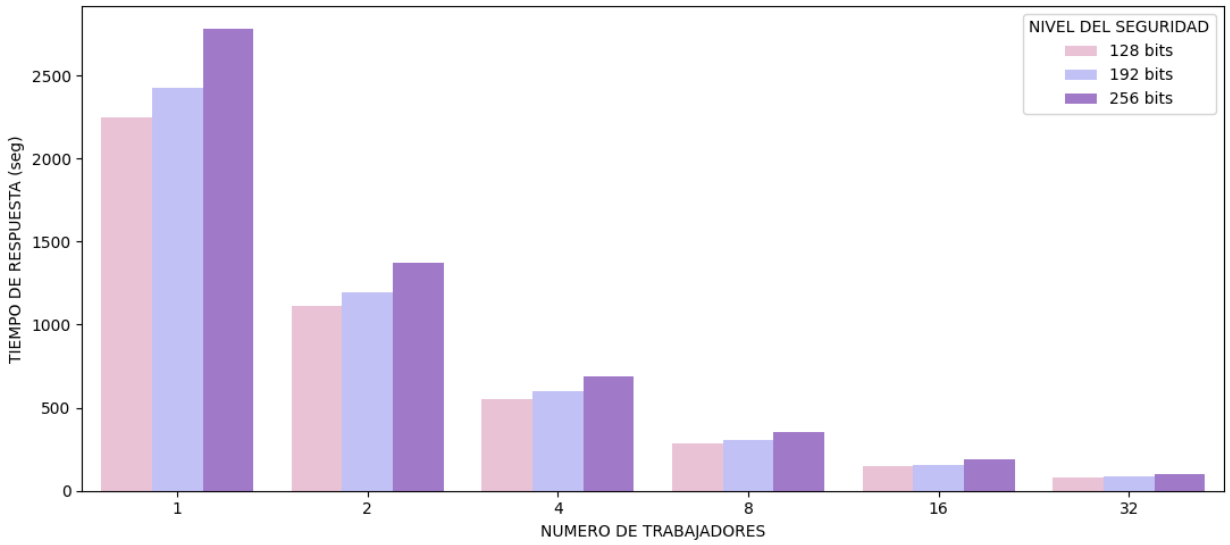

Figura 34. Tiempo de respuesta para la tarea de cifrado de los datos con el esquema de Liu.

En la Figura 35 se muestra el tiempo de respuesta del sistema para la tarea de cifrado de una matriz de distancias con el esquema FDH-OPE. En el eje x se muestra la cantidad de *workers* disponibles, y en el eje y el tiempo de respuesta para ambas tareas. Para obtenerla se evaluó un conjunto de datos con 20,000 registros y 4 atributos, con un total de 31 repeticiones. En esta gráfica se puede observar que al aumentar la cantidad de *workers* disponibles disminuye el tiempo de cifrado de la matriz U.

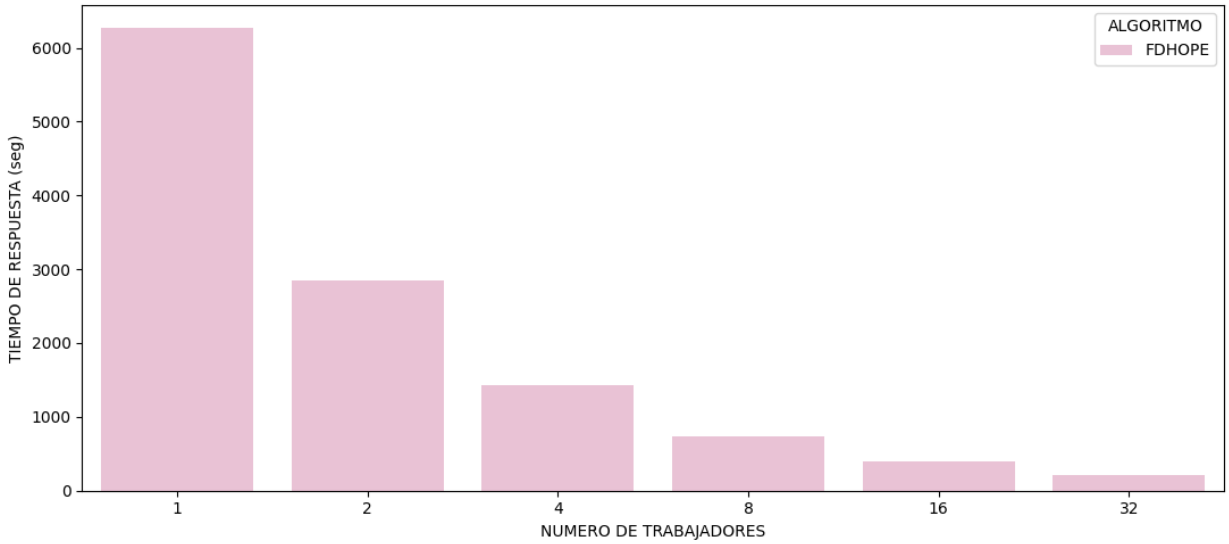

Figura 35. Tiempo de respuesta para la generación de la matriz U y para el cifrado de dicha matriz.

7.4. Etapa 3 - Tiempo de respuesta del sistema para el PPDM en general

En esta etapa se evaluaron 50 conjuntos de datos, con un total de 31 repeticiones para cada uno. Además, se evaluaron para un nivel de seguridad de 128 bits.

En la Figura 36 se muestra el tiempo de respuesta del sistema para la tarea de agrupamiento k-means y sus versiones seguras sk-means y dbs$k$-means, aumentando la cantidad de *workers* disponibles para el sistema.

Esta es una gráfica agrupada, donde en el eje x se presentan los diferentes algoritmos y en el eje y los tiempos de respuesta del sistema en segundos. En la parte superior, las agrupaciones están hechas por la cantidad de *workers*, aumentando de 1, 2, 4 y 8.

Se observa un comportamiento exponencial decreciente al aumentar la cantidad de *workers* disponibles, lo que indica una mejora en los tiempos de respuesta cuando se cuenta con más *workers*. Los tiempos están agrupados de acuerdo a los diferentes componentes de la plataforma, cada barra incluye tres tiempos: tiempo de servicio para el cliente, tiempo de servicio para el manager y tiempo de servicio para el *worker*.

En la gráfica, el tiempo del cliente es el más alto, especialmente para dbs$k$-means, donde este tiempo incluye el cifrado del dataset original con el esquema de Liu, la generación de la matriz de distancias U y el cifrado de dicha matriz con el esquema FDH-OPE. Para sk-means, el tiempo incluye el cifrado del dataset original con el esquema de Liu y la generación de la matriz de distancias U. En el caso de k-means, no se incluyen estos tiempos adicionales, por lo que su barra es la que menos tiempo consume.

El tiempo del manager no es significativo, ya que su principal función es balancear las peticiones que recibe, por lo que no se aprecia en la gráfica. El tiempo del *worker* incluye las tareas de procesamiento del PPDM indicado.

En la Figura 37 se muestra el tiempo de respuesta del sistema para el algoritmo de agrupamiento nnc y su versión segura dbsnnc. Esta gráfica muestra el tiempo de respuesta del sistema para los algoritmos de agrupamiento nnc y dbsnnc. La gráfica está agrupada de modo que en el eje x se encuentran los diferentes algoritmos y en el eje y los tiempos de respuesta del sistema en segundos. Las agrupaciones en la parte superior de la gráfica están determinadas por la cantidad de *workers*, que varían de 1, 2, 4 a 8 *workers*. Se observa un comportamiento exponencial decreciente en los tiempos de respuesta a medida que se incrementa la cantidad de *workers* disponibles, mostrando la eficiencia de la paralelización.

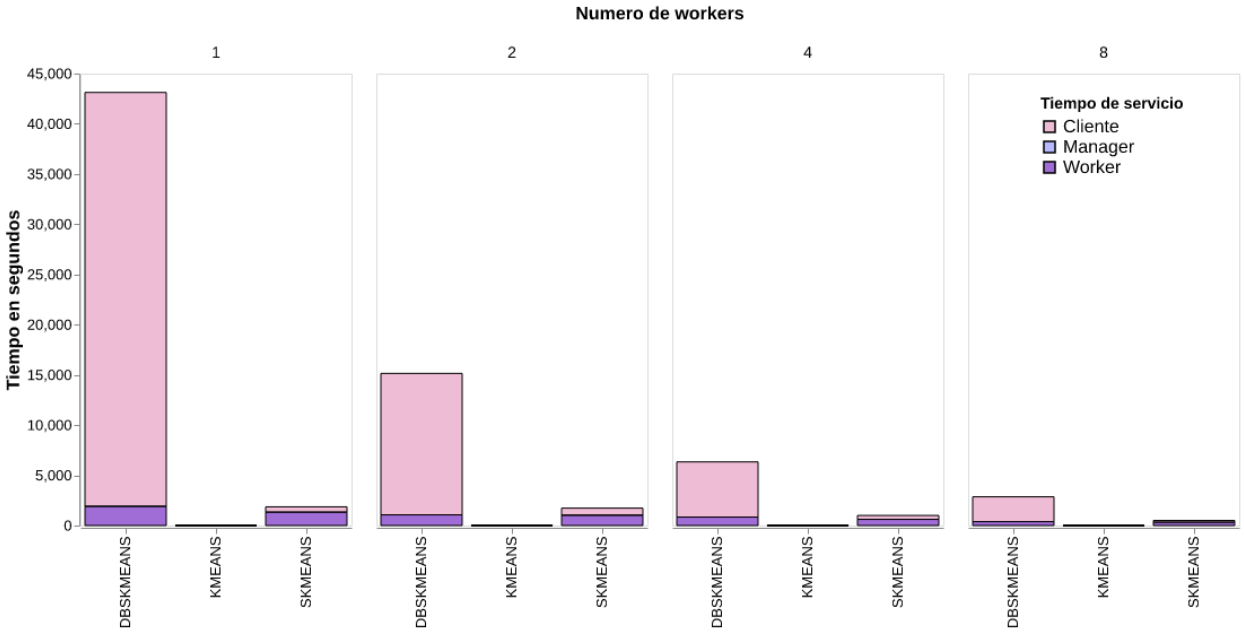

Figura 36. Tiempo de respuesta del sistema para los algoritmos de agrupamiento k-means, sk-means y dbs$k$-means.

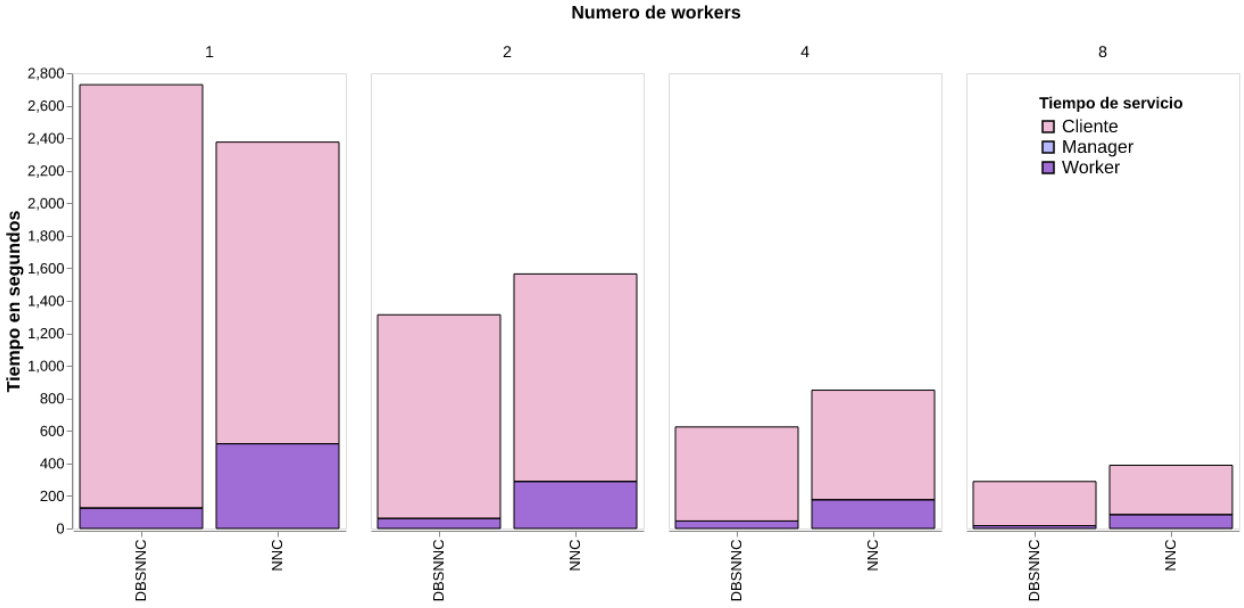

Figura 37. Tiempo de respuesta del sistema para los algoritmos de agrupamiento nnc y dbsnnc.

Cada barra en la gráfica está desglosada en tres componentes: tiempo de servicio para el cliente, tiempo de servicio

para el manager y tiempo de servicio para el *worker*. El tiempo del cliente es el más significativo en la gráfica. Para dbsnnc, este tiempo incluye el cifrado del dataset original con el esquema de Liu y la generación de la matriz de distancias ED necesaria para el algoritmo. En el caso de nnc, el tiempo del cliente incluye la generación de la matriz de distancias ED, pero no involucra cifrado de datos, lo que lo hace más rápido que dbsnnc. El tiempo del manager es insignificante en comparación con el tiempo de los otros componentes, ya que su principal función es balancear las peticiones recibidas, lo que no consume mucho tiempo. Por esta razón, el tiempo del manager no es claramente visible en la gráfica. El tiempo del *worker*, que sí es visible en la gráfica, incluye las tareas de procesamiento del PPDM indicado para cada algoritmo, lo que contribuye a la variabilidad observada en los tiempos de respuesta.

En la Figura 38 se muestra el tiempo de respuesta del sistema para el algoritmo de clasificación k-nn y su versión segura sk-nn. La gráfica está organizada de manera agrupada, con el eje x mostrando los diferentes algoritmos y el eje y representando los tiempos de respuesta del sistema en segundos. Las agrupaciones en la parte superior de la gráfica indican la cantidad de *workers*, que incrementan de 1, 2, 4 a 8 *workers*. Se observa un comportamiento exponencialmente decreciente en los tiempos de respuesta a medida que aumenta el número de *workers* disponibles. Esto sugiere que la paralelización mejora significativamente el rendimiento del sistema.

Un algoritmo de clasificación se divide en dos fases: entrenamiento y predicción. En la gráfica, los tiempos están agrupados y cada barra incluye cuatro tiempos: el tiempo de entrenamiento (realizado en el lado del cliente) y los tiempos de predicción que comprenden tres componentes: tiempo de servicio para el cliente, tiempo de servicio para el manager y tiempo de servicio para el *worker*. El tiempo de entrenamiento es relativamente pequeño para ambos algoritmos, ya simplemente toman el modelo de los datos y lo almacenan en el sistema de almacenamiento. El tiempo de predicción del lado del cliente, para sk-nn incluye el cifrado del modelo y del conjunto de datos a etiquetar utilizando el esquema de Liu. Para k-nn, el tiempo de predicción del cliente es menor porque solo implica almacenar el conjunto de datos a etiquetar en el sistema de almacenamiento sin realizar cifrado adicional. En el caso del tiempo del manager, al igual que en las gráficas anteriores, no es significativo, porque su función principal es balancear las peticiones recibidas, lo cual no requiere mucho tiempo de procesamiento. Por esta razón, el tiempo del manager no es claramente visible en la gráfica. En el caso del tiempo del *worker*, para k-nn, incluye la lectura de ambos datasets (el modelo y el conjunto de datos a etiquetar), la integración de estos datasets y posteriormente el procesamiento. Para k-nn, el tiempo del *worker* incluye la lectura y el procesamiento del conjunto de datos a etiquetar.

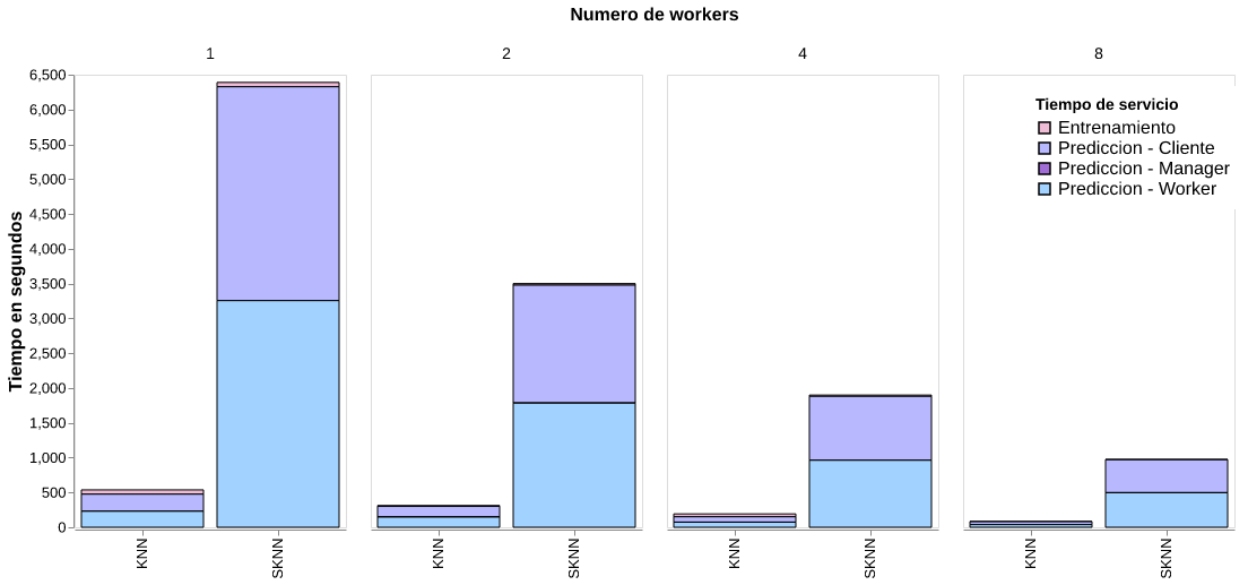

Figura 38. Tiempo de respuesta del sistema para los algoritmos de clasificación *k*-nn y s*k*-nn.

De manera general, la reducción exponencial en los tiempos de respuesta para los diferentes algoritmos evaluados, al aumentar la cantidad de *workers* disponibles, demuestra la eficiencia de la distribución de carga y la paralelización en el

procesamiento de datos.

El tiempo de respuesta de un algoritmo seguro es significativamente más alto debido al cifrado adicional requerido por el esquema de Liu o el esquema FDH-OPE (cuando se requiere), mientras que una versión estándar presenta tiempos más bajos al no requerir este procesamiento adicional. La constante carga de trabajo del manager reafirma su rol de balanceador de peticiones, sin impactar significativamente los tiempos totales.

Estas gráficas proporciona una clara visión del rendimiento de los algoritmos estándar y sus respectivas versiones seguras en función del número de *workers* y resalta las diferencias en los tiempos de procesamiento debido a las características específicas de cada algoritmo, especialmente en cuanto al cifrado y manejo de datos.

7.5. Discusión de los resultados

Tras analizar los resultados obtenidos durante la etapa de experimentación con los diversos algoritmos implementados, podemos notar que, si bien los tiempos de respuesta del sistema pueden incrementarse, la precisión de los resultados finales se mantiene intacta.

Es notable que los algoritmos de Minería de Datos con Preservación de Privacidad (PPDM) ofrecen una precisión de etiquetado similar a las versiones estándar. Esto sugiere que salvaguardar la privacidad de los datos no compromete significativamente los resultados de agrupamiento y clasificación.

Este aumento en el tiempo de procesamiento puede ser considerado como una inversión en privacidad y confidencialidad, dependiendo del contexto y los requisitos del escenario. En ese sentido, la plataforma cuenta con técnicas de balanceo de carga y paralelismo para disminuir el tiempo de procesamiento del lado del PS, tal como se mostró en las gráficas anteriores. Es fundamental destacar que los resultados obtenidos tanto para la clasificación como el agrupamiento están ligados a la calidad y naturaleza de los datos utilizados.

En esta evaluación, el objetivo no radica en juzgar los resultados de la agrupación en sí, sino en determinar la eficacia de los algoritmos cuando se aplican a datos cifrados en lugar de datos en claro. Las gráficas previamente presentadas indican que los algoritmos producen resultados comparables en las métricas presentadas, *coeficiente de Silhouette* e *índice de Davies-Bouldin* para los algoritmos de agrupamiento, y precisión, exactitud y recall para los algoritmos de clasificación, lo que evidencia que no hay una pérdida de utilidad de los datos al emplear un algoritmo de minería segura.

Desde la perspectiva de DMaaS, este hallazgo es de relevancia, ya que asegura al PD que los resultados obtenidos mediante modelos de minería de datos en un entorno no confiable, como la nube, son consistentes con aquellos obtenidos al utilizar datos en un entorno controlado.

8. Conclusiones

En este reporte técnico se ha presentado el diseño, construcción, validación y evaluación de una plataforma para Minería de Datos con Preservación de Privacidad como Servicio (PPDMaaS). La plataforma se ha construido para hacer frente al problema de privacidad y seguridad en la tarea de minería de datos bajo un escenario de Big Data y donde los algoritmos de minería de datos son ejecutados por un tercero no confiable.

Se realizó un diseño que simplifica y favorece su uso desde la perspectiva del propietario de datos. Además, su construcción permite flexibilidad para usar distintos algoritmos de minería de datos con preservación de privacidad. En este trabajo, la plataforma se evaluó usando cinco algoritmos de agrupamiento: k-means, sk-means, dbs$k$-means, nnc y dbsnnc, así como dos algoritmos de clasificación: k-nn y sk-nn. Todos ellos con la propiedad de preservación de privacidad y con la posibilidad de usar distintos niveles de seguridad, igual o mayor a 128 bits. Esta flexibilidad es uno de los aspectos más significativos de la plataforma PPDMaaS desarrollada para adaptarse y considerar otros PPDM, actuales o futuros, sin requerir modificaciones significativas en su arquitectura.

La eficiencia en la plataforma se consiguió incorporando, desde la fase de diseño, patrones de paralelismo y esquemas de balanceo de carga, que permitan hacer frente a los altos volúmenes de datos de entrada en las tareas del propietario de los datos (PD) y de los datos cifrados a procesar del lado del proveedor del servicio (PS) y del usuario final.

Los resultados revelan que incluir privacidad y confidencialidad en los datos no compromete la calidad de los resultados finales de la minería de datos. Sin embargo, se observa un incremento en el tiempo de ejecución debido a la

capa de seguridad basada en cifrado homomórfico. En algunos algoritmos, tales como sk-means o dbs$k-$means, este incremento es mayor, lo que se atribuye a la necesidad de establecer una comunicación continua entre el PS y el PD, lo que introduce una latencia adicional en el intercambio de mensajes. Esta sobrecarga en las tareas de cómputo se han mitigado con los esquemas eficaces de paralelismo y balanceo de carga considerados.

La plataforma PPDMaaS propuesta ofrece un marco flexible y escalable para la ejecución de tareas de minería de datos con preservación de privacidad, destacando su capacidad para mejorar continuamente su capacidad y rendimiento debido a su construcción modular y elástica.

Referencias

[1] "Machine learning repository uci." `https://archive.ics.uci.edu/ml/datasets.php`, 2007 (accessed October 25, 2022).

[2] R. H. Hariri, E. M. Fredericks, and K. M. Bowers, "Uncertainty in big data analytics: Survey, opportunities, and challenges," *Journal of Big Data*, vol. 6, no. 1, pp. 1–16, 2019.

[3] B. Vuleta, "How much data is created every day?," tech. rep., Seeds Scientific, 2021.

[4] K. H. Tan, G. Ji, C. P. Lim, and M.-L. Tseng, "Using big data to make better decisions in the digital economy," *International Journal of Production Research*, vol. 55, no. 17, pp. 4998–5000, 2017.

[5] Y. Zhang, T. Huang, and E. F. Bompard, "Big data analytics in smart grids: A review," *Energy Informatics*, vol. 1, no. 1, pp. 1–24, 2018.

[6] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," *Journal of Big Data*, vol. 2, no. 1, pp. 1–32, 2015.

[7] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "Iot privacy and security: Challenges and solutions," *Applied Sciences*, vol. 10, no. 12, p. 4102, 2020.

[8] D. Arunachalam, N. Kumar, and J. P. Kawalek, "Understanding big data analytics capabilities in supply chain management: Unravelling the issues, challenges and implications for practice," *Transportation Research Part E: Logistics and Transportation Review*, vol. 114, pp. 416–436, 2018.

[9] T. Chen, J. Chen, and B. Zhou, "A system for parallel data mining service on cloud," in *Second International Conference on Cloud and Green Computing*, pp. 329–330, IEEE, IEEE, 2012.

[10] A. Sunyaev, "Cloud computing," in *Internet Computing*, pp. 195–236, Springer, 2020.

[11] N. Almutairi, F. Coenen, and K. Dures, "Secure third-party data clustering using securecl, data and multi-user order preserving encryption," *Expert Systems*, p. e12581, 2020.

[12] N. Naveen and K. Thippeswamy, "Safeguarding solitude for outsourced data mining," *International Journal of Computer Science & Communication*, pp. 76–79, 2020.

[13] D. Zhang, "Big data security and privacy protection," in *8th International Conference on Management and Computer Science (ICMCS)*, vol. 77, pp. 275–278, Atlantis Press, Atlantis Press, 2018.

[14] D. T. Larose, *Data mining methods & models*. John Wiley & Sons, 2006.

[15] V. R. Falmari and M. Brindha, "Privacy preserving cloud based secure digital locker using paillier based difference function and chaos based cryptosystem," *Journal of Information Security and Applications*, vol. 53, p. 102513, 2020.

[16] Y. Lindell and B. Pinkas, "Privacy preserving data mining.," *Journal of Cryptology*, vol. 15, no. 3, pp. 177–206, 2002.

[17] N. M. Almutairi, *Privacy Preserving Third Party Data Mining Using Cryptography*. The University of Liverpool (United Kingdom), 2020.

[18] A. M. Qadir and N. Varol, "A review paper on cryptography," in *2019 7th international symposium on digital forensics and security (ISDFS)*, pp. 1–6, IEEE, IEEE, 2019.

[19] D. Zhao, X. Hu, S. Xiong, J. Tian, J. Xiang, J. Zhou, and H. Li, "K-means clustering and knn classification based on negative databases," *Applied soft computing*, vol. 110, p. 107732, 2021.

[20] J. Han, *Data Mining*, pp. 595–598. Boston, MA: Springer US, 2009.

[21] J. Han, J. Pei, and H. Tong, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2022.

[22] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data-ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2019.

[23] X. Jing, Z. Yan, and W. Pedrycz, "Security data collection and data analytics in the internet: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 586–618, 2018.

[24] J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2020.

[25] V. Kotu and B. Deshpande, *Data Science: Concepts and Practice*. Morgan Kaufmann, 2018.

[26] S. Yang and J. K. Kim, "Statistical data integration in survey sampling: A review," *Japanese Journal of Statistics and Data Science*, vol. 3, no. 2, pp. 625–650, 2020.

[27] X. L. Dong and T. Rekatsinas, "Data integration and machine learning: A natural synergy," in *International Conference on Management of Data*, pp. 1645–1650, ACM, ACM, 2018.

[28] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Data processing and text mining technologies on electronic medical records: A review," *Journal of Healthcare Engineering*, vol. 2018, pp. 1–18, 2018.

[29] A. Pika, M. T. Wynn, S. Budiono, A. H. ter Hofstede, W. M. van der Aalst, and H. A. Reijers, "Towards privacy-preserving process mining in healthcare," in *International Conference on Business Process Management*, pp. 483–495, Springer, Springer, 2019.

[30] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020.

[31] A. Dogan and D. Birant, "Machine learning and data mining in manufacturing," *Expert Systems with Applications*, vol. 166, p. 114060, 2021.

[32] C. Romero and S. Ventura, "Educational data mining and learning analytics: An updated survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 3, p. e1355, 2020.

[33] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[34] S. S. Alaoui, Y. Farhaoui, and B. Aksasse, "Classification algorithms in data mining," *International Journal of Tomography & Simulation*, vol. 31, pp. 34–44, 2018.

[35] A. V. Novikov, "Pyclustering: Data mining library," *Journal of Open Source Software*, vol. 4, no. 36, p. 1230, 2019.

[36] M. Z. Hossain, M. N. Akhtar, R. B. Ahmad, and M. Rahman, "A dynamic k-means clustering for data mining," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 2, pp. 521–526, 2019.

[37] N. Yogeesh, "Mathematical approach to representation of locations using k-means clustering algorithm," *International Journal of Mathematics And its Applications*, vol. 9, no. 1, pp. 127–136, 2021.

[38] B. Y. Gravesteijn, D. Nieboer, A. Ercole, H. F. Lingsma, D. Nelson, B. Van Calster, E. W. Steyerberg, C. Åkerlund, K. Amrein, N. Andelic, *et al.*, "Machine learning algorithms performed no better than regression models for prognostication in traumatic brain injury," *Journal of Clinical Epidemiology*, vol. 122, pp. 95–107, 2020.

[39] S. Ray, "A quick review of machine learning algorithms," in *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 35–39, IEEE, IEEE, 2019.

[40] S. M. Ghafari and C. Tjortjis, "A survey on association rules mining using heuristics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1307, 2019.

[41] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[42] M. Syakur, B. Khotimah, E. Rochman, and B. D. Satoto, "Integration k-means clustering method and elbow method for identification of the best customer profile cluster," in *IOP Conference Series: Materials Science and Engineering*, vol. 336, p. 012017, IOP Publishing, IOP Publishing, 2018.

[43] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[44] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (L. M. L. Cam and J. Neyman, eds.), vol. 1, pp. 281–297, University of California Press, 1967.

[45] C. Yuan and H. Yang, "Research on k-value selection method of k-means clustering algorithm," *J*, vol. 2, no. 2, pp. 226–235, 2019.

[46] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020.

[47] I. Ba'abbad, T. Althubiti, A. Alharbi, K. Alfarsi, and S. Rasheed, "A short review of classification algorithms accuracy for data prediction in data mining applications," *Journal of Data Analysis and Information Processing*, vol. 9, no. 3, pp. 162–174, 2021.

[48] F. Kazerouni, A. Bayani, F. Asadi, L. Saeidi, N. Parvizi, and Z. Mansoori, "Type 2 diabetes mellitus prediction using data mining algorithms based on the long-noncoding rnas expression: A comparison of four data mining approaches," *BMC Bioinformatics*, vol. 21, pp. 1–13, 2020.

[49] P. Chandra Kanth and M. S. Anbarasi, "A generic framework for data analysis in privacy-preserving data mining," in *Computational Intelligence in Data Mining* (H. S. Behera, J. Nayak, B. Naik, and D. Pelusi, eds.), (Singapore), pp. 653–661, Springer Singapore, 2020.

[50] V. U. Rani and M. S. Rao, "Privguard: Sensitivity guided anonymization based ppdm with automatic selection of sensitive attributes," in *7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 1832–1837, IEEE, IEEE, 2021.

[51] R. Ratra, P. Gulia, and N. S. Gill, "Evaluation of re-identification risk using anonymization and differential privacy in healthcare," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, 2022.

[52] K. Arava and S. Lingamgunta, "Adaptive k-anonymity approach for privacy preserving in cloud," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 2425–2432, 2020.

[53] A. Kiran and N. Shirisha, "K-anonymization approach for privacy preservation using data perturbation techniques in data mining," *Materials Today: Proceedings*, vol. 64, pp. 578–584, 2022.

[54] C. Mauger, G. L. Mahec, and G. Dequen, "Multi-criteria optimization using l-diversity and t-closeness for k-anonymization," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM and CBT*, pp. 73–88, Springer, Springer, 2020.

[55] R. Ratra and P. Gulia, "Privacy preserving data mining: Techniques and algorithms," *International Journal of Engineering Trends and Technology*, vol. 68, no. 11, pp. 56–62, 2020.

[56] Y. Canbay, S. Sagiroglu, and Y. Vural, "A new utility-aware anonymization model for privacy preserving data publishing," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 10, 2022.

[57] F. S. Nininahazwe, "Studying l-diversity and k-anonymity over datasets with sensitive fields," in *Artificial Intelligence and Security: 5th International Conference*, pp. 63–73, Springer, Springer, 2019.

[58] S. Shimona, "Survey on privacy preservation technique," in *International Conference on Inventive Computation Technologies (ICICT)*, pp. 64–68, IEEE, IEEE, 2020.

[59] M. Binjubeir, A. A. Ahmed, M. A. B. Ismail, A. S. Sadiq, and M. K. Khan, "Comprehensive survey on big data privacy protection," *IEEE Access*, vol. 8, pp. 20067–20079, 2019.

[60] N. Kousika and K. Premalatha, "An improved privacy-preserving data mining technique using singular value decomposition with three-dimensional rotation data perturbation," *The Journal of Supercomputing*, vol. 77, pp. 10003–10011, 2021.

[61] S. Belkoura, M. Zanin, and A. LaTorre, "Fostering interpretability of data mining models through data perturbation," *Expert Systems with Applications*, vol. 137, pp. 191–201, 2019.

[62] W. Haoxiang, S. Smys, *et al.*, "Big data analysis and perturbation using data mining algorithm," *Journal of Soft Computing Paradigm (JSCP)*, vol. 3, no. 01, pp. 19–28, 2021.

[63] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving distributed machine learning with federated learning," *Computer Communications*, vol. 171, pp. 112–125, 2021.

[64] M. A. P. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, "Efficient privacy preservation of big data for accurate data mining," *Information Sciences*, vol. 527, pp. 420–443, 2020.

[65] A. A. Khaliq, A. Anjum, A. B. Ajmal, J. L. Webber, A. Mehbodniya, and S. Khan, "A secure and privacy preserved parking recommender system using elliptic curve cryptography and local differential privacy," *IEEE Access*, vol. 10, pp. 56410–56426, 2022.

[66] H.-Y. Tran and J. Hu, "Privacy-preserving big data analytics: A comprehensive survey," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 207–218, 2019.

[67] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, jul 2018.

[68] H. Osman, M. A. Maarof, and M. M. Siraj, "Hybrid solution for privacy-preserving data mining on the cloud computing," in *Emerging Trends in Intelligent Computing and Informatics* (F. Saeed, F. Mohammed, and N. Gazem, eds.), (Cham), pp. 748–758, Springer International Publishing, 2020.

[69] M. Richards, *Software Architecture Patterns*. O'Reilly Media, second ed., 7 2022.

[70] M. Zichichi, S. Ferretti, and G. D'angelo, "A framework based on distributed ledger technologies for data management and services in intelligent transportation systems," *IEEE Access*, vol. 8, pp. 100384–100402, 2020.

[71] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.

[72] H. Yokoyama, "Machine learning system architectural pattern for improving operational stability," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 267–274, IEEE, IEEE, 2019.

[73] P. Siano, G. De Marco, A. Rolán, and V. Loia, "A survey and evaluation of the potentials of distributed ledger technology for peer-to-peer transactive energy exchanges in local energy markets," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3454–3466, 2019.

[74] A. Sunyaev and A. Sunyaev, "Cloud computing," *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, pp. 195–236, 2020.

[75] A. T. Raj, G. Nalinipriya, N. Kanagavalli, and K. Maheswari, "A dynamic object oriented approach for context aware services in distributed pervasive healthcare environment," in *2020 7th International Conference on Smart Structures and Systems (ICSSS)*, pp. 1–6, IEEE, IEEE, 2020.

[76] A. Ali-Eldin, J. Westin, B. Wang, P. Sharma, and P. Shenoy, "Spotweb: Running latency-sensitive distributed web services on transient cloud servers," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 1–12, ACM, 2019.

[77] S. Sadeghiram, H. Ma, and G. Chen, "Multi-objective distributed web service composition—a link-dominance driven evolutionary approach," *Future Generation Computer Systems*, 2023.

[78] S. Malakuti, J. Schmitt, M. Platenius-Mohr, S. Grüner, R. Gitzel, and P. Bihani, "A four-layer architecture pattern for constructing and managing digital twins," in *Software Architecture: 13th European Conference, ECSA 2019, Paris, France, September 9–13, 2019, Proceedings 13*, pp. 231–246, Springer, Springer, 2019.

[79] N. Ford, M. Richards, P. Sadalage, and Z. Dehghani, *Software Architecture: The Hard Parts Modern Trade-Off Analyses for Distributed Architectures*. O'Reilly Media, second ed., 2022.

[80] R. Laigner, M. Kalinowski, P. Diniz, L. Barros, C. Cassino, M. Lemos, D. Arruda, S. Lifschitz, and Y. Zhou, "From a monolithic big data system to a microservices event-driven architecture," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 213–220, IEEE, IEEE, 2020.

[81] D. Yang, X. Gao, L. Kong, Y. Pang, and B. Zhou, "An event-driven convolutional neural architecture for non-intrusive load monitoring of residential appliance," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 173–182, 2020.

[82] V. Mosin, D. Durisic, and M. Staron, "Applicability of machine learning architectural patterns in vehicle architecture: A case study.," in *ECSA (Companion)*, Springer, Springer, 2021.

[83] D. Carrizales, D. D. Sánchez-Gallegos, H. Reyes, J. González-Compeán, M. Morales-Sandoval, J. Carretero, and A. Galaviz-Mosqueda, "A data preparation approach for cloud storage based on containerized parallel patterns," in *Internet and Distributed Computing Systems: 12th International Conference, IDCS 2019, Naples, Italy, October 10–12, 2019, Proceedings 12*, pp. 478–490, Springer, Springer, 2019.

[84] J. L. Ortega-Arjona, "Applying architectural patterns for parallel programming: Solving a matrix multiplication," in *26th European Conference on Pattern Languages of Programs*, pp. 1–7, Springer, Springer, 2021.

[85] J.-P. Abegg, Q. Bramas, T. Brugière, and N. Thomas, "Supra, a distributed publish/subscribe protocol with blockchain as a conflict resolver," in *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pp. 41–42, IEEE, IEEE, 2021.

[86] J.-P. Abegg, Q. Bramas, T. Brugière, and T. Noel, "Distributed publish/subscribe protocol with minimum number of encryption," in *23rd International Conference on Distributed Computing and Networking*, pp. 117–123, ACM, ACM, 2022.

[87] H. Ganesan, M. Longsworth, and G. Sutmann, "Parallel hybrid monte carlo/molecular statics for simulation of solute segregation in solids," in *Journal of Physics: Conference Series*, vol. 1740, p. 012001, IOP Publishing, IOP Publishing, 2021.

[88] B. Tola, Y. Jiang, and B. E. Helvik, "On the resilience of the nfv-mano: An availability model of a cloud-native architecture," in *2020 16th International Conference on the Design of Reliable Communication Networks DRCN 2020*, pp. 1–7, IEEE, IEEE, 2020.

[89] R. Zhang, C. Zhang, Z. Cao, W. Song, P. S. Tan, J. Zhang, B. Wen, and J. Dauwels, "Learning to solve multiple-tsp with time window and rejections via deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[90] P. Burkhardt and G. Jékely, "Evolution of synapses and neurotransmitter systems: the divide-and-conquer model for early neural cell-type evolution," *Current Opinion in Neurobiology*, vol. 71, pp. 127–138, 2021.

[91] L. Wang, J. Zhang, O. Wang, Z. Lin, and H. Lu, "Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 541–550, IEEE, IEEE, 2020.

[92] Y. Liu and D. Wang, "Divide and conquer: A deep casa approach to talker-independent monaural speaker separation," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 27, no. 12, pp. 2092–2102, 2019.

[93] D. L. Freire, R. Z. Frantz, F. Roos-Frantz, and S. Sawicki, "Survey on the run-time systems of enterprise application integration platforms focusing on performance," *Software: Practice and Experience*, vol. 49, no. 3, pp. 341–360, 2019.

[94] B. M. Santos, A. de Souza Landi, I. G.-R. de Guzmán, M. Piattini, and V. V. de Camargo, "Towards a reference architecture for adm-based modernization tools," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pp. 114–123, ACM, ACM, 2019.

[95] T. Malaska and S. Babu, *Rebuilding reliable data pipelines through modern tools*. O'Reilly Media, first ed., 2019.

[96] J. Densmore, *Data pipelines pocket reference*. O'Reilly Media, 2021.

[97] M. Ham, J. Moon, G. Lim, J. Jung, H. Ahn, W. Song, S. Woo, P. Kapoor, D. Chae, G. Jang, *et al.*, "Nnstreamer: Efficient and agile development of on-device ai systems," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 198–207, IEEE, IEEE, 2021.

[98] S. Purohit and A. Bhar, "A novel approach to deep learning applications using fork-join neural network architecture," *International Research Journal of Engineering and Technology*, 2021.

[99] H. Nishikawa, K. Shimada, I. Taniguchi, and H. Tomiyama, "Scheduling of moldable fork-join tasks with inter-and intra-task communications," in *Proceedings of the 23th International Workshop on Software and Compilers for Embedded Systems*, pp. 7–12, ACM, ACM, 2020.

[100] A. I. Baranchikov, P. A. Baranchikov, and N. S. Fokina, "Fork-join method of decomposing monolithic software products," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–5, IEEE, IEEE, 2021.

[101] H. Nishikawa, K. Shimada, I. Taniguchi, and H. Tomiyama, "Energy-aware scheduling of malleable fork-join tasks under a deadline constraint on heterogeneous multicores," *ACM SIGBED Review*, vol. 16, no. 3, pp. 57–62, 2019.

[102] C. Seguin, O. Sporns, A. Zalesky, F. Calamante, *et al.*, "Network communication models narrow the gap between the modular organization of structural and functional brain networks," *NeuroImage*, vol. 257, p. 119323, 2022.

[103] W. Li, S. Wang, W. Xie, K. Yu, and C. Feng, "Large scale medical image online three-dimensional reconstruction based on webgl using four tier client server architecture," *Information Visualization*, vol. 22, no. 2, pp. 100–114, 2023.

[104] D. Berbecaru and A. Lioy, "Towards simplifying pki implementation: Client-server based validation of public key certificates," *arXiv preprint arXiv:1910.06641*, 2019.

[105] A. Braeken, "Public key versus symmetric key cryptography in client–server authentication protocols," *International Journal of Information Security*, vol. 21, no. 1, pp. 103–114, 2022.

[106] S. Ali, R. Alauldeen, and A. Ruaa, "What is client-server system: Architecture, issues and challenge of client-server system," *HBRP Publication*, pp. 1–6, 2020.

[107] A. Alferaidi, K. Yadav, Y. Alharbi, W. Viriyasitavat, S. Kautish, and G. Dhiman, "Federated learning algorithms to optimize the client and cost selections," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–12, 2022.

[108] A. G. Reddy, A. K. Das, V. Odelu, A. Ahmad, and J. S. Shin, "A privacy preserving three-factor authenticated key agreement protocol for client–server environment," *Journal of ambient intelligence and humanized computing*, vol. 10, pp. 661–680, 2019.

[109] H. Niaei, A. Masoumi, A. R. Jafari, M. Marzband, S. H. Hosseini, and A. Mahmoudi, "Smart peer-to-peer and transactive energy sharing architecture considering incentive-based demand response programming under joint uncertainty and line outage contingency," *Journal of Cleaner Production*, vol. 363, p. 132403, 2022.

[110] V. Iannino, C. Mocci, and V. Colla, "A hybrid peer-to-peer architecture for agent-based steel manufacturing processes," *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 528–533, 2021.

[111] G. Fu, Y. Zhang, and G. Yu, "A fair comparison of message queuing systems," *IEEE Access*, vol. 9, pp. 421–432, 2020.

[112] I. Ungurean and N. C. Gaitan, "A software architecture for the industrial internet of things—a conceptual model," *Sensors*, vol. 20, no. 19, p. 5603, 2020.

[113] A. Diro, H. Reda, N. Chilamkurti, A. Mahmood, N. Zaman, and Y. Nam, "Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication," *IEEE Access*, vol. 8, pp. 60539–60551, 2020.

[114] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–35, 2019.

[115] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp. 149–158, 2020.

[116] F. Garcia-Carballeira, A. Calderon, and J. Carretero, "Enhancing the power of two choices load balancing algorithm using round robin policy," *Cluster Computing*, vol. 24, no. 2, pp. 611–624, 2021.

[117] J. Anselmi, "Combining size-based load balancing with round-robin for scalable low latency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 886–896, 2019.

[118] S. T. Milan, L. Rajabion, H. Ranjbar, and N. J. Navimipour, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments," *Computers & Operations Research*, vol. 110, pp. 159–187, 2019.

[119] N. Bansal and O. N. Feldheim, "The power of two choices in graphical allocation," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 52–63, ACM, 2022.

[120] M. Wang, W. Zhao, K. Cheng, Z. Wu, and J. Liu, "Homomorphic encryption based privacy preservation scheme for dbscan clustering," *Electronics*, vol. 11, no. 7, p. 1046, 2022.

[121] W.-J. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu, "Pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption; pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 123–137, IEEE, 2021.

[122] Y. Fan, J. Bai, X. Lei, W. Lin, Q. Hu, G. Wu, J. Guo, and G. Tan, "Ppmck: Privacy-preserving multi-party computing for k-means clustering," *Journal of Parallel and Distributed Computing*, vol. 154, pp. 54–63, 8 2021.

[123] G. Smilarubavathy, R. Nidhya, N. V. Abiramy, and A. D. Kumar, "Paillier homomorphic encryption with k-means clustering algorithm (phekc) for data mining security in cloud," *Lecture Notes in Networks and Systems*, vol. 145, pp. 941–948, 2021.

[124] W. Wu, M. Xian, U. Parampalli, and B. Lu, "Efficient privacy-preserving frequent itemset query over semantically secure encrypted cloud database," *World Wide Web*, vol. 24, pp. 607–629, 2021.

[125] A. Brandão, R. Mendes, and J. P. Vilela, "Efficient privacy preserving distributed k-means for non-iid data," in *Advances in Intelligent Data Analysis XIX*, pp. 439–451, Springer International Publishing, 2021.

[126] C. Gong, Z. Dong, A. Gani, H. Qi, and H. Qi, "Quantum k-means algorithm based on trusted server in quantum cloud computing," *Quantum Information Processing*, vol. 20, 2021.

[127] G. Dilip, "An efficient privacy preserving on high-order heterogeneous data using fuzzy k-prototype clustering," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 5191–5203, 2021.

[128] I. Chandrakar and V. R. Hulipalled, "Privacy preserving big data mining using pseudonymization and homomorphic encryption," *2021 2nd Global Conference for Advancement in Technology (GCAT)*, 2021.

[129] Y. Zou, Z. Zhao, S. Shi, L. Wang, Y. Peng, Y. Ping, and B. Wang, "Highly secure privacy-preserving outsourced k-means clustering under multiple keys in cloud computing," *Security and Communication Networks*, vol. 2020, 2020.

[130] L. Sun, S. Ci, X. Liu, X. Zheng, Q. Yu, and Y. Luo, "A privacy-preserving density peak clustering algorithm in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 32, 2020.

[131] A. Alabdulatif, I. Khalil, and X. Yi, "Towards secure big data analytic for cloud-enabled applications with fully homomorphic encryption," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 192–204, 3 2020.

[132] P. L. Lekshmy and M. A. Rahiman, "Hybrid approach to speed-up the privacy preserving kernel k-means clustering and its application in social distributed environment," *Journal of Network and Systems Management*, vol. 28, pp. 398–422, 2020.

[133] X. Guo, H. Lin, Y. Wu, and M. Peng, "A new data clustering strategy for enhancing mutual privacy in healthcare iot systems," *Future Generation Computer Systems*, vol. 113, pp. 407–417, 12 2020.

[134] Y. Zhu and X. Li, "Privacy-preserving k-means clustering with local synchronization in peer-to-peer networks," *Peer-to-Peer Networking and Applications*, vol. 13, pp. 2272–2284, 9 2020.

[135] F. O. Catak, I. Aydin, O. Elezaj, and S. Yildirim-Yayilgan, "Practical implementation of privacy preservingclustering methods using a partially homomorphicencryption algorithm," *Electronics*, vol. 9, 2020.

[136] X. Huo and M. Liu, "Privacy-preserving decentralized optimization using homomorphic encryption," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 630–633, 2020.

[137] N. Almutairi, F. Coenen, and K. Dures, "A cryptographic ensemble for secure third party data analysis: Collaborative data clustering without data owner participation," *Data and Knowledge Engineering*, vol. 126, p. 101734, 3 2020.

[138] S. Ci, L. Sun, X. Liu, T. Du, and X. Zheng, "A secure density peaks clustering algorithm on cloud computing," in *International Symposium on Cyberspace Safety and Security*, pp. 533–541, Springer, Springer, 2019.

[139] M. Gao, B. Li, C. Wang, L. Ma, and J. Xu, "User behavior clustering scheme with automatic tagging over encrypted data," *IEEE Access*, vol. 7, pp. 170648–170657, 12 2019.

[140] C. Wang, A. Wang, X. Liu, and J. Xu, "Research on k-means clustering algorithm over encrypted data," in *Cyberspace Safety and Security*, pp. 182–191, Springer International Publishing, 2019.

[141] O. E. Omri, A. Boudguiga, M. Izabachene, and W. Klaudel, "Privacy-preserving k-means clustering: an application to driving style recognition," in *International Conference on Network and System Security*, pp. 685–696, Springer, Springer, 2019.

[142] Y. Cai and C. Tang, "Privacy of outsourced two-party k-means clustering," *Concurrency and Computation: Practice and Experience*, vol. 33, 2019.

[143] J. H. Cheon, D. Kim, and J. H. Park, "Towards a practical cluster analysis over encrypted data," in *International Conference on Selected Areas in Cryptography*, pp. 227–249, Springer, Springer, 2019.

[144] B. K. Song, S. Yoo, M. Hong, and J. W. Yoon, "A bitwise design and implementation for privacy-preserving data mining: From atomic operations to advanced algorithms," *Security and Communication Networks*, vol. 2019, 2019.

[145] G. Sakellariou and A. Gounaris, "Homomorphically encrypted k-means on cloud-hosted servers with low client-side load," *Computing*, vol. 101, pp. 1813–1836, 2019.

[146] S. R. Paladhi, R. Mohan Kumar, A. Deepshika Reddy, C. Vinayak, and T. Pusphavathi, "Enhanced possibilistic c-means clustering on big data while ensuring security," in *International Conference on Computer Networks and Communication Technologies*, pp. 583–588, Springer, Springer, 2019.

[147] B. Liu, L. Chen, X. Zhu, W. Qiu, L. Chen, and X. Zhu, "Encrypted data indexing for the secure outsourcing of spectral clustering," *Knowledge and Information Systems*, vol. 60, pp. 1307–1328, 2019.

[148] Q. Zhang, L. T. Yang, A. Castiglione, Z. Chen, and P. Li, "Secure weighted possibilistic c-means algorithm on cloud for clustering big data," *Information Sciences*, vol. 479, pp. 515–525, 4 2019.

[149] I. V. Anikin and R. M. Gazimov, "Privacy preserving data mining in terms of dbscan clustering algorithm in distributed systems," in *2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pp. 1–4, IEEE, IEEE, 2018.

[150] N. Almutairi, F. Coenen, and K. Dures, "Data clustering using homomorphic encryption and secure chain distance matrices," *KDIR*, pp. 39–48, 2018.

[151] N. Almutairi, F. Coenen, and K. Dures, "Third party data clustering over encrypted data without data owner participation: Introducing the encrypted distance matrix," in *Big Data Analytics and Knowledge Discovery*, pp. 163–173, Springer International Publishing, 2018.

[152] N. Almutairi, F. Coenen, and K. Dures, "Secure third party data clustering using data: Multi-user order preserving encryption and super secure chain distance matrices (best technical paper)," in *Artificial Intelligence XXXV*, pp. 3–17, Springer International Publishing, 2018.

[153] G. S. Kumar, K. Premalatha, G. U. Maheshwari, and P. R. Kanna, "No more privacy concern: A privacy-chain based homomorphic encryption scheme and statistical method for privacy preservation of user's private and sensitive data," *Expert Systems with Applications*, vol. 234, p. 121071, 2023.

[154] Z. Ma, J. Ma, Y. Miao, X. Liu, K.-K. R. Choo, Y. Gao, and R. H. Deng, "Verifiable data mining against malicious adversaries in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, p. 953, 2022.

[155] B. Xie, T. Xiang, and X. Liao, "Access-oblivious and privacy-preserving k nearest neighbors classification in dual clouds," *Computer Communications*, vol. 184, pp. 12–23, 2 2022.

[156] H. Osman and M. Osman, *Cloud based privacy preserving data mining model using hybrid k-anonymity and partial homomorphic encryption*. PhD thesis, Universiti Teknologi Malaysia, Malaysia, 2022.

[157] M. E. Skarkala, M. Maragoudakis, S. Gritzalis, and L. Mitrou, "Ppdm-tan: A privacy-preserving multi-party classifier †," *Computation*, vol. 9, 1 2021.

[158] B. Pulido-Gaytan, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, G. Radchenko, A. Avetisyan, and A. Y. Drozdov, "Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 1666–1691, 1 2021.

[159] J. Liu, C. Wang, Z. Tu, X. A. Wang, C. Lin, and Z. Li, "Secure knn classification scheme based on homomorphic encryption for cyberspace," *Security and Communication Networks*, 11 2021.

[160] M. Alkhelaiwi, W. Boulila, J. Ahmad, A. Koubaa, M. Driss, A. Anand, J. Bullock, and M. Luengo-Oroz, "An efficient approach based on privacy-preserving deep learning for satellite image classification," *Remote Sensing*, vol. 13, 2021.

[161] E. Sarkar, E. Chielle, G. Gürsoy, O. Mazonka, M. Gerstein, and M. Maniatakos, "Fast and scalable private genotype imputation using machine learning and partially homomorphic encryption," *Journal of Parallel and Distributed Computing*, vol. 154, pp. 54–63, 2021.

[162] F. Kuang, B. Mi, Y. Li, Y. Weng, and S. Wu, "Multiparty homomorphic machine learning with data security and model preservation," *Mathematical Problems in Engineering*, vol. 2021, 2021.

[163] M. Sun, R. Yang, and L. Hu, "A secure distributed machine learning protocol against static semi-honest adversaries," *Applied Soft Computing*, vol. 102, p. 107095, 4 2021.

[164] J. Zhang, Z. L.Jiang, P. Li, and S. M. Yiu, "Privacy-preserving multikey computing framework for encrypted data in the cloud," *Information Sciences*, vol. 575, pp. 217–230, 10 2021.

[165] J. Paul, M. S. Muthu, S. Annamalai, W. Ming, A. A. Badawi, B. Veeravalli, K. Mi, and M. I. Aung, "Privacy-preserving collective learning with homomorphic encryption," *IEEE Access*, vol. 9, pp. 132084–132096, 2021.

[166] J. M. Cortés-Mendoza, G. Radchenko, A. Tchernykh, B. Pulido-Gaytan, M. Babenko, A. Avetisyan, P. Bouvry, and A. Zomaya, "Lr-gd-rns: Enhanced privacy-preserving logistic regression algorithms for secure deployment in untrusted environments," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, IEEE, IEEE, 2021. tabla 1 - tomar en cuenta para estado del arte.

[167] H. Huang, Y. Wang, and H. Zong, "Support vector machine classification over encrypted data," *Applied Intelligence*, vol. 51, pp. 1–11, 2021.

[168] J. A. Alzubi, O. A. Alzubi, M. Beseiso, A. K. Budati, and K. Shankar, "Optimal multiple key-based homomorphic encryption with deep neural networks to secure medical data transmission and diagnosis," *Expert Systems*, vol. 38, 11 2021.

[169] J. Yu, Z. Qiao, W. Tang, D. Wang, and X. Cao, "Blockchain-based decision tree classification in distributed networks," *Intelligent Automation and Soft Computing*, vol. 29, pp. 713–728, 2021.

[170] M. Baryalai, *A platform for practical homomorphic encryption in neural network classification: a thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Ph. D.) in Information Technology, Massey University*. PhD thesis, Massey University, 2021.

[171] J. Chen, Y. Feng, Y. Liu, W. Wu, and G. Yang, "Non-interactive privacy-preserving naïve bayes classifier using homomorphic encryption," in *International Conference on Security and Privacy in New Computing Environments*, pp. 192–203, Springer, Springer, 2021.

[172] Y. Fan, J. Bai, X. Lei, Y. Zhang, B. Zhang, K. C. Li, and G. Tan, "Privacy preserving based logistic regression on big data," *Journal of Network and Computer Applications*, vol. 171, p. 102769, 12 2020.

[173] J. Liu, Y. Tian, Y. Zhou, Y. Xiao, and N. Ansari, "Privacy preserving distributed data mining based on secure multi-party computation," *Computer Communications*, vol. 153, pp. 208–216, 3 2020.

[174] T. Ishiyama, T. Suzuki, and H. Yamana, "Highly accurate cnn inference using approximate activation functions over homomorphic encryption," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 3989–3995, IEEE, IEEE, 2020.

[175] A. A. Badawi, L. Hoang, C. F. Mun, K. Laine, K. Mi, and M. I. Aung, "Privft: Private and fast text classification with homomorphic encryption," *IEEE Access*, vol. 8, pp. 226544–226556, 12 2020.

[176] D. Syed, S. S. Refaat, and O. Bouhali, "Privacy preservation of data-driven models in smart grids using homomorphic encryption," *Information*, vol. 11, 2020.

[177] Y. Ping, B. Hao, X. Hei, J. Wu, and B. Wang, "Maximized privacy-preserving outsourcing on support vector clustering," *Electronics*, vol. 9, 2020.

[178] T. T. N. Le, T. V. X. Phuong, A. D. B. P. Street, B. T. District, and H. C. M. City, "Privacy preserving jaccard similarity by cloud-assisted for classification," *Wireless Personal Communications*, vol. 112, pp. 1875–1892, 2020.

[179] G. Qiu, X. Gui, and Y. Zhao, "Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking," *IEEE Access*, vol. 8, pp. 107601–107613, 2020.

[180] X. Dong, J. Chen, and K. Zhang, "Privacy-preserving locally weighted linear regression over encrypted millions of data," *IEEE Access*, vol. 8, pp. 2247–2257, 2020.

[181] W. Wang, Y. Gan, C.-M. Vong, and C. Chen, "Homo-elm: fully homomorphic extreme learning machine," *International Journal of Machine Learning and Cybernetics*, vol. 11, pp. 1531–1540, 2020.

[182] J. Li, X. Kuang, S. Lin, X. Ma, and Y. Tang, "Privacy preservation for machine learning training and classification based on homomorphic encryption schemes," *Information Sciences*, vol. 526, pp. 166–179, 7 2020.

[183] J. Li and H. Huang, "Faster secure data mining via distributed homomorphic encryption," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2706–2714, ACM, ACM, 2020.

[184] R. U. Haque, A. S. M. T. Hasan, Q. Jiang, and Q. Qu, "Privacy-preserving k-nearest neighbors training over blockchain-based encrypted health data," *Electronics*, vol. 9, 2020.

[185] Y. Chai, Y. Zhan, B. Wang, Y. Ping, and Z. Zhang, "Improvement on a privacy-preserving outsourced classification protocol over encrypted data," *Wireless Networks*, vol. 26, pp. 4363–4374, 2020.

[186] J.-C. Bajard, P. Martins, L. Sousa, and V. Zucca, "Improving the efficiency of svm classification with fhe," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1709–1722, 2020.

[187] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, "Effective activation functions for homomorphic evaluation of deep neural networks," *IEEE Access*, vol. 8, pp. 153098–153112, 2020.

[188] Y. Liu, H. Huang, F. Xiao, R. Malekian, and W. Wang, "Classification and recognition of encrypted eeg data based on neural network," *Journal of Information Security and Applications*, vol. 54, 2020.

[189] X. Liu, R. H. Deng, K.-K. R. Choo, Y. Yang, and H. Pang, "Privacy-preserving outsourced calculation toolkit in the cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, pp. 898–911, 2020.

[190] J. Park, D. H. Lee, and D. H. Lee, "Parallelly running k-nearest neighbor classification over semantically secure encrypted data in outsourced environments," *IEEE Access*, vol. 8, pp. 64617–64633, 2020.

[191] Z. Li and M. Sun, "Privacy-preserving classification of personal data with fully homomorphic encryption: An application to high-quality ionospheric data prediction," in *International Conference on Machine Learning for Cyber Security*, pp. 437–446, Springer, Springer, 2020.

[192] E. J. Chou, A. Gururajan, K. Laine, N. K. Goel, A. Bertiger, and J. W. Stokes, "Privacy-preserving phishing web page classification via fully homomorphic encryption," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2792–2796, IEEE, IEEE, 2020.

[193] X. Sun, P. Zhang, J. K. Liu, J. Yu, X. X. Sun, P. Zhang, J. Yu, W. Xie, and J. K. Liu, "Private machine learning classification based on fully homomorphic encryption," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, pp. 352–364, 2020.

[194] E. Zorarpacı and S. A. Özel, "A hybrid approach of homomorphic encryption and differential privacy for privacy preserving classification," *International Journal of Applied Mathematics, Electronics and Computers*, vol. 8, pp. 138–147, 2020.

[195] Y. Yasumura, Y. Ishimaki, and H. Yamana, "Secure naïve bayes classification protocol over encrypted data using fully homomorphic encryption," in *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications Services*, pp. 45–54, Association for Computing Machinery, 2019.

[196] A. Wood, V. Shpilrain, K. Najarian, and D. Kahrobaei, "Private naive bayes classification of personal biomedical data: Application in cancer data analysis," *Computers in Biology and Medicine*, vol. 105, pp. 144–150, 2 2019.

[197] E. Hesamifard, H. Takabi, and M. Ghasemi, "Deep neural networks classification over encrypted data," in *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, pp. 97–108, ACM, ACM, 2019.

[198] X. Xiao, T. Wu, Y. Chen, and X. Fan, "Privacy-preserved approximate classification based on homomorphic encryption," *Mathematical and Computational Applications*, vol. 24, 2019.

[199] M. Izabachène, R. Sirdey, and M. Zuber, "Practical fully homomorphic encryption for fully masked neural networks," in *Cryptology and Network Security*, pp. 24–36, Springer International Publishing, 2019.

[200] R. Bellafqira, G. Coatrieux, E. Genin, and M. Cozic, "Secure multilayer perceptron based on homomorphic encryption," in *Digital Forensics and Watermarking*, pp. 322–336, Springer International Publishing, 2019.

[201] J. Chao, A. A. Badawi, B. Unnikrishnan, J. Lin, C. F. Mun, J. M. Brown, J. P. Campbell, M. Chiang, J. Kalpathy-Cramer, V. R. Chandrasekhar, P. Krishnaswamy, and K. M. M. Aung, "Carenets: Compact and resource-efficient cnn for homomorphic inference on encrypted medical images," *arXiv preprint arXiv:1901.10074*, 2019.

[202] N. J. H. Marcano, M. Moller, S. Hansen, and R. H. Jacobsen, "On fully homomorphic encryption for privacy-preserving deep learning," in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, IEEE, 2019.

[203] Y. Li, Z. L. Jiang, X. X. Wang, J. Fang, and E. Zhang, "Securely outsourcing id3 decision tree in cloud computing," *Wireless Communications and Mobile Computing*, 2018.

[204] Y. Y. Liu, Y. Luo, Y. Zhu, and X. Li, "Secure multi-label data classification in cloud by additionally homomorphic encryption," *Information Sciences*, vol. 468, pp. 89–102, 11 2018.

[205] S. Kim, M. Omori, T. Hayashi, T. Omori, L. Wang, and S. Ozawa, "Privacy-preserving naive bayes classification using fully homomorphic encryption," in *International Conference on Neural Information Processing*, pp. 349–358, Springer, Springer, 2018.

[206] H. Yang, W. He, J. Li, and H. Li, "Efficient and secure knn classification over encrypted data using vector homomorphic encryption," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, IEEE, 2018.

[207] P. Li, T. Li, H. Ye, J. Li, X. Chen, and Y. Xiang, "Privacy-preserving machine learning with multiple data providers," *Future Generation Computer Systems*, vol. 87, pp. 341–350, 10 2018.

[208] C. Bonte and F. Vercauteren, "Privacy-preserving logistic regression training," *BMC Medical Genomics*, vol. 11, pp. 13–21, 10 2018.

[209] H. Park, P. Kim, H. Kim, K. W. Park, and Y. Lee, "Efficient machine learning over encrypted data with non-interactive communication," *Computer Standards and Interfaces*, vol. 58, pp. 87–108, 5 2018.

[210] P. Li, J. Li, Z. Huang, C. Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Computing*, vol. 21, pp. 277–286, 2018.

[211] W.-J. Lu, J.-J. Zhou, and J. Sakuma, "Non-interactive and output expressive private comparison from homomorphic encryption," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 67–74, ACM, 2018.

[212] A. Wood, V. Shpilrain, K. Najarian, A. Mostashari, and D. Kahrobaei, "Private-key fully homomorphic encryption for private classification," in *International Congress on Mathematical Software*, pp. 475–481, Springer, Springer, 2018.

[213] J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46938–46948, 2018.

[214] N. Almutairi, F. Coenen, and K. Dures, "K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction," in *International Conference on Big Data Analytics and Knowledge Discovery*, pp. 274–285, Springer, 2017.

[215] S. Kim, M. Omori, T. Hayashi, T. Omori, L. Wang, and S. Ozawa, "Privacy-preserving naive bayes classification using fully homomorphic encryption," in *International Conference on Neural Information Processing*, pp. 349–358, Springer, 2018.

[216] M. E. Skarkala, M. Maragoudakis, S. Gritzalis, and L. Mitrou, "Ppdm-tan: A privacy-preserving multi-party classifier," *Computation*, vol. 9, no. 1, p. 6, 2021.

[217] P. Valdiviezo-Diaz, F. Ortega, E. Cobos, and R. Lara-Cabrera, "A collaborative filtering approach based on naïve bayes classifier," *IEEE Access*, vol. 7, pp. 108581–108592, 2019.

[218] L. X. Li and S. S. Abdul Rahman, "Students' learning style detection using tree augmented naive bayes," *Royal Society Open Science*, vol. 5, no. 7, p. 172108, 2018.

[219] G. Qiu, X. Gui, and Y. Zhao, "Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking," *IEEE Access*, vol. 8, pp. 107601–107613, 2020.

[220] A. F. Schmidt and C. Finan, "Linear regression and the normality assumption," *Journal of Clinical Epidemiology*, vol. 98, pp. 146–151, 2018.

[221] N. Almutairi, F. Coenen, and K. Dures, "Third party data clustering over encrypted data without data owner participation: Introducing the encrypted distance matrix," in *International Conference on Big Data Analytics and Knowledge Discovery*, pp. 163–173, Springer, 2018.

[222] N. Almutairi, F. Coenen, and K. Dures, "A cryptographic ensemble for secure third party data analysis: Collaborative data clustering without data owner participation," *Data & Knowledge Engineering*, vol. 126, p. 101734, 2020.

[223] C. Yuan and H. Yang, "Research on k-value selection method of k-means clustering algorithm," *J*, vol. 2, no. 2, pp. 226–235, 2019.

[224] S. Zhou and Z. Xu, "A novel internal validity index based on the cluster centre and the nearest neighbour cluster," *Applied Soft Computing*, vol. 71, pp. 78–88, 2018.

[225] M. Zhang, "Use density-based spatial clustering of applications with noise (dbscan) algorithm to identify galaxy cluster members," in *IOP Conference Series: Earth and Environmental Science*, vol. 252, p. 042033, IOP Publishing, 2019.

[226] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[227] S. Sieniutycz, "Complex systems of neural networks," in *Complexity and Complex Thermo-Economic Systems* (S. Sieniutycz, ed.), pp. 51–84, Elsevier, 2020.

[228] C. Fernandez-Basso, A. J. Francisco-Agra, M. J. Martin-Bautista, and M. D. Ruiz, "Finding tendencies in streaming data using big data frequent itemset mining," *Knowledge-Based Systems*, vol. 163, pp. 666–674, 2019.

[229] D. Liu, "Practical fully homomorphic encryption without noise reduction." Cryptology ePrint Archive, Paper 2015/468, 2015. `https://eprint.iacr.org/2015/468`.

[230] S. B. Das, S. K. Mishra, and A. K. Sahu, "A new modified version of standard rsa cryptography algorithm," in *Smart Computing Paradigms: New Progresses and Challenges: Proceedings of ICACNI 2018, Volume 2*, pp. 281–287, Springer, 2020.

[231] D. Liu and S. Wang, "Nonlinear order preserving index for encrypted database query in service cloud environments," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 13, pp. 1967–1984, 2013.

[232] Z. Liu, X. Chen, J. Yang, C. Jia, and I. You, "New order preserving encryption model for outsourced databases in cloud environments," *Journal of Network and Computer Applications*, vol. 59, pp. 198–207, 2016.

[233] N. Almutairi, F. Coenen, and K. Dures, "K-means clustering using homomorphic encryption and an updatable distance matrix: Secure third party data clustering with limited data owner interaction," in *Big Data Analytics and Knowledge Discovery* (L. Bellatreche and S. Chakravarthy, eds.), (Cham), pp. 274–285, Springer International Publishing, 2017.

[234] D. Mittal, D. Kaur, and A. Aggarwal, "Secure data mining in cloud using homomorphic encryption," in *International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 1–7, IEEE, IEEE, 2014.

[235] J. I. C. Barrios, J. L. G. Compeán, and I. L. Arévalo, "Método adaptativo-reactivo de replicación para sistemas de almacenamiento de alta disponibilidad," master's thesis, Cinvestav Unidad Tamaulipas, México, August 2022.

[236] Z. Liu, Z. Bai, Z. Liu, X. Li, C. Kim, V. Braverman, X. Jin, and I. Stoica, "{DistCache}: Provable load balancing for {Large-Scale} storage systems with distributed caching," in *17th USENIX Conference on File and Storage Technologies (FAST 19)*, pp. 143–157, USENIX Association, 2019.