UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Proyecto de grado

Implementación de un algoritmo de anonimización para la plataforma de datos masivos de Plan Ceibal

Bruno Serra Oddo

Diego Rosolino Ruétalo

María Soledad Rivas Masullo

Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Mayo de 2020

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Proyecto de grado

Implementación de un algoritmo de anonimización para la plataforma de datos masivos de Plan Ceibal

Bruno Serra Oddo

Diego Rosolino Ruétalo

María Soledad Rivas Masullo

Tesis de grado presentada en la Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de grado en Ingeniería en Computación.

Directores:

Dra. Lorena Etcheverry

Dr. Eduardo Giménez

Montevideo – Uruguay

Mayo de 2020

Agradecimentos

En primer lugar queremos agradecer al Dr. Alberto Castro, Posdoc del Instituto de Computación en la Facultad de Ingeniería UdelaR, no solo por proporcionar el entorno distribuido necesario para la ejecución, sino también por estar siempre dispuesto a ayudarnos con las dificultades que encontramos. También reconocer a Germán Álvarez, perteneciente al equipo de Gestión de Datos del Plan Ceibal, por configurar la plataforma de la organización con los requisitos necesarios.

Asimismo agradecer especialmente a los tutores del proyecto, el Dr. Eduardo Giménez, investigador del Centro Tecnológico ICT4V, y a la Dra. Lorena Etcheverry, docente del Instituto de Computación en la Facultad de Ingeniería UdelaR. Ambos nos guiaron durante todo el proceso, animándonos y conteniéndonos cuando fue necesario.

Finalmente, no queremos dejar de mencionar a nuestras familias, quienes conformaron el sostén principal y compartieron los logros obtenidos a lo largo de la carrera.

RESUMEN

En la actualidad, cada vez más información es compartida entre distintas organizaciones con distintos fines, como por ejemplo, para extraer estadísticas que permitan tomar ciertas decisiones, dar soporte a investigaciones científicas o para ser utilizados en modelos de Aprendizaje Automático. Cuando esta información refiere a datos personales, es donde se torna fundamental que la privacidad de estos se preserve. Dado que gestiona los datos de los niños, niñas y docentes uruguayos, el Plan Ceibal posee un especial interés en el escenario descripto.

Es con el objetivo de brindar una herramienta que permita a la organización anonimizar los datos personales que posee, que el presente proyecto estudia algunos aspectos de la anonimización de datos en el contexto de *Big Data*, abordando el problema de escalabilidad que tienen hoy en día las técnicas de anonimización para entornos centralizados. Para ello, se exponen distintas técnicas, mencionando en cada una los enfoques que estas utilizan.
Además, se brinda una descripción detallada del marco tecnológico del entorno distribuido utilizado y se proporciona la implementación de un algoritmo de anonimización basado en la técnica *k-anonymity* junto con una comparación de resultados en un ambiente distribuido y uno centralizado, utilizando *PySpark* como interfaz de comunicación con *Spark*.

Se concluye exponiendo los desafíos que enfrenta el responsable del proceso de anonimización de datos, así como también las dificultades que enfrenta al momento de aplicar una de estas técnicas con el fin de mantener la mayor utilidad de los datos posible al mismo tiempo que se preserva su privacidad.
Palabras claves:

Privacidad, Datos personales, Entornos distribuidos, Spark, Big Data.

Lista de figuras

Lista de tablas

Tabla de contenidos

Capítulo 1

El problema

En este capítulo se establece el contexto del problema a resolver así como también la importancia y el desafío que plantea el proceso de anonimización de datos personales. Luego, basados en el marco definido, se establecen los objetivos del proyecto, el resultado esperado y se describe la estructura del informe.

1.1. Contexto

El gran volumen de datos personales que son manejados por organizaciones plantea el siguiente problema: ¿cómo explotar esos datos en beneficio de sus propietarios y al mismo tiempo proteger la privacidad de quiénes los aportan? Esta problemática debe ser abordada por las distintas organizaciones, las cuales utilizan métodos de resolución que varían según la naturaleza de los datos utilizados por cada una.

Un ejemplo de una organización que se ve afectada por este problema es Centro Ceibal. Centro Ceibal es una organización de carácter público que nace en el año 2007 como un plan de inclusión e igualdad de oportunidades, cuyo objetivo es apoyar con tecnología las políticas educativas uruguayas. Su misión es la de:

> *"Promover la integración de la tecnología al servicio de la educación para mejorar su calidad e impulsar procesos de innovación social, inclusión y crecimiento personal."* [1]

Para dar soporte a esta misión, Plan Ceibal necesita gestionar la información personal (tanto actual como histórica) de los niños, niñas y docentes de los centros educativos públicos.

Opción de estudio	Tipo de beneficiario			Cantidad de Centros
	Total	Alumnos	Docentes	
2007	5.266	4.964	302	-
2008	179.505	164.711	14.794	-
2009	350.004	327.736	22.268	1.779
2010	417.445	393.521	23.924	1.882
2011	503.609	476.065	27.544	2.127
2012	566.552	527.312	39.240	2.392
2013	607.459	559.836	47.623	2.546
2014	665.145	607.757	57.388	2.805
2015	707.418	656.276	51.142	3.151
2016	718.702	669.498	49.204	4.769
2017	765.701	714.983	50.718	3.351

Figura 1.1: Cobertura del Plan Ceibal, por tipo de beneficiario y cantidad de centros, según año lectivo [1].

A escala de Uruguay, el Plan Ceibal maneja un conjunto de datos considerable. Como se puede observar en la Figura 1.1, la cantidad de personas contempladas por el Plan se incrementó en casi 50.000 individuos cada año, durante los últimos diez años, excepto en 2016. Este crecimiento anual, junto con la gestión de datos históricos, genera que la cantidad de información que debe ser mantenida por la organización sea cada vez mayor. Es en parte por el gran volumen de datos manejados que la realidad descripta se considera un contexto de aplicación de *Big Data*.

Enfrentado a esta realidad, el Plan Ceibal debió sortear el desafío tecnológico de desplegar una plataforma que permita almacenar y procesar esos datos eficientemente. Con ese objetivo, Ceibal desplegó en 2018 dos instancias de la plataforma de datos masivos *Hortonworks Data Platform*[2] (HDP).

Hortonworks Data Platform es una plataforma de código abierto que permite el almacenamiento y procesamiento distribuido de grandes volúmenes de datos provenientes de múltiples fuentes. Su implementación está basada en componentes pertenecientes al *Ecosistema Hadoop* [2] como por ejemplo *Hadoop Dis-*

[1]Fuente: Anuario Estadístico 2018, INE. http://www.ine.gub.uy/anuario-estadistico
[2]https://www.cloudera.com/products/hdp.html

tributed File System[3] (HDFS) para el almacenamiento y *Apache Spark*[4] para el procesamiento distribuido.

La aspiración de Plan Ceibal es utilizar dicha plataforma para entrenar modelos de Aprendizaje Automático [3] que permitan construir predicciones basadas en datos estadísticos. Estos modelos son construidos por Plan Ceibal, pero también pueden serlo por investigadores externos pertenecientes a la Fundación Ceibal, a universidades o a otras instituciones. En el escenario más habitual, Plan Ceibal entrega los datos necesarios a los investigadores, quienes desarrollan el modelo en un entorno propio, a partir de herramientas de modelado que pueden ser muy variadas. Los conjuntos de datos que son compartidos deben preservar la privacidad como dicta la *Ley de Protección de Datos Personales* [4]. Al momento de compartir información, Plan Ceibal debe aplicar técnicas de anonimización para cumplir con la ley antes mencionada.

La anonimización de datos personales es un mecanismo para proteger la identidad de un individuo y/o información sensible sobre él mismo dentro de un conjunto de datos a ser expuesto o compartido públicamente [5]. Existen diferentes técnicas que pueden ser utilizadas para lograr la anonimización de dichos conjuntos, las cuáles tienen como objetivo evitar que posibles atacantes accedan o infieran datos confidenciales sobre los individuos de los mismos. El mayor reto cuando se aplican estas técnicas es encontrar el mejor compromiso entre maximizar la protección brindada y mantener el valor estadístico de los datos. Las técnicas utilizadas por Ceibal hasta el momento consisten, mayoritariamente, en la eliminación de datos que puedan llevar a la reidentificación de un individuo del conjunto de datos liberado, por ejemplo, eliminación de columnas. Esto implica la pérdida de información estadísticamente útil.

Para que un conjunto de datos se considere correctamente anonimizado, no debe ser posible asociar unívocamente un registro del conjunto de datos anonimizado con un registro del conjunto de datos original (reidentificación) ni tampoco deducir información confidencial sobre ningún individuo específico. Estas propiedades pueden ser alcanzadas mediante la aplicación de modelos de anonimización sobre un conjunto de datos. Un modelo posible es el de k-

[3] http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[4] https://spark.apache.org

anonymity [6], el cuál consiste en sustituir ciertos valores de algunos atributos por un valor más general que los englobe. El objetivo final es que la probabilidad de reidentificación de un individuo dentro del conjunto anonimizado sea inferior a $\frac{1}{k}$.
El desafío que presenta este modelo es encontrar una sustitución de valores que produzca un conjunto que cumpla la condición de *k-anonymity* provocando la menor pérdida de información sobre los datos del conjunto original. Surgen de la literatura varias técnicas centralizadas y algunas propuestas para hacerlo en ambientes distribuidos.

1.2. Objetivos

El objetivo del proyecto es desarrollar un algoritmo que garantice *k-anonymity*, usando Spark, a ser ejecutado en el entorno de *Hortonworks Data Platform* perteneciente al Plan Ceibal. Para validar y evaluar el algoritmo, se utilizará el conjunto de datos *Adult* [7], que presenta datos personales relevados en el censo del año 1994 en Estados Unidos y es un estándar de facto para la evaluación de las técnicas de anonimización.

1.3. Resultados esperados

El resultado esperado es una herramienta de anonimización adaptada al contexto organizacional y tecnológico descrito anteriormente. La implementación se realizará en el lenguaje de programación Python y haciendo uso de la interfaz de *Apache Spark* denominada *PySpark*[5].
El algoritmo mencionado implementa la condición de *k-anonymity* y, a través de parámetros de entrada, permite especificar el conjunto de datos necesario para su ejecución. De este modo, por cada ejecución, se obtendrá un conjunto de datos que satisfaga la condición de *k-anonymity*.

Además de la implementación del algoritmo, se incluirán dos módulos de Python adicionales: uno encargado de preprocesar el conjunto de datos original, con el fin de dar el formato necesario al conjunto de datos, y otro para realizar validaciones sobre el resultado obtenido luego de la ejecución. Finalmente,

[5]https://spark.apache.org/docs/latest/api/python/index.html

también se incluirá un instructivo que explicará cómo configurar y utilizar el algoritmo.

1.4. Estructura del documento

El resto del documento se estructura de la siguiente manera:

- En el Capítulo 2 se detalla el marco teórico del problema de la anonimización de datos personales. Se presenta una taxonomía de técnicas de anonimización descritas en la literatura y se ubica dentro de la misma el modelo de *k-anonymity*, definiendo formalmente el problema a resolver y las métricas utilizadas.
- En el Capítulo 3 se describe el marco tecnológico en el cual se realizará la implementación.
- En el Capitulo 4 se brinda la descripción de la solución implementada y las estructuras de datos que dan soporte a la misma. Se incluyen también una breve descripción de la metodología y herramientas utilizadas para la gestión del proyecto.
- En el Capítulo 5 se muestran los resultados obtenidos con el conjunto de datos *Adult* y se comparan con aquellos resultantes de utilizar otras herramientas de anonimización disponibles, como ARX[6] y UTD[7].
- En el Capítulo 6 se presentan las conclusiones y el trabajo futuro.

[6]https://arx.deidentifier.org
[7]http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php

Capítulo 2

Marco teórico

A continuación se explica en detalle el marco teórico del problema de la anonimización de datos, sus características y conceptos más relevantes. El contenido del capítulo se basa principalmente en los capítulos 2 al 6 del libro *Database Anonymization* [5].

2.1. Definiciones

2.1.1. Formatos de liberación de datos

La anonimización de datos es particularmente relevante cuando las organizaciones deciden publicar o liberar un conjunto de datos. Liberar un conjunto de datos refiere a exponer información que pertenece a una organización para que pueda ser utilizada por agentes externos a la misma. Al momento de realizar una liberación de datos se debe considerar que la naturaleza de los mismos determina los posibles ataques a los que estos se pueden enfrentar. Cuanta mayor sea la anonimización del conjunto, más difícil será para un atacante extraer información del mismo. Existen, al menos, tres tipos de formatos utilizados para realizar publicaciones con información estadística: Microdatos, datos tabulados y bases de datos interactivas.

En el formato denominado Microdatos, cada registro perteneciente al conjunto de datos contiene información relacionada a un individuo especifico. Su volumen y formato los hacen accesibles, informativos y procesables para la toma de decisiones. Por ejemplo, la Tabla 2.1 presenta un conjunto de datos médicos inventado donde cada registro representa a un paciente y su diagnóstico.

C.I.	Edad	Diagnóstico
4.928.457-3	20	Diabetes
4.489.324-1	24	Infección
4.213.983-8	26	Infección
4.382.093-4	29	Cáncer
3.382.572-3	45	Diabetes
3.528.457-3	40	Cáncer
2.993.098-9	50	Cáncer
4.288.739-7	30	Diabetes
4.002.458-1	35	Cáncer
3.924.492-6	39	Infección

Tabla 2.1: Ejemplo inventado de conjunto de datos en formato Microdatos.

En las publicaciones que siguen el formato conocido como datos tabulados, el conjunto de datos muestra valores agregados para distintos grupos de individuos. Este formato es utilizado comúnmente para presentar el resultado de estadísticas oficiales. En la Tabla 2.2 se brinda un ejemplo que presenta la cantidad de establecimientos educativos, maestros y estudiantes correspondientes al año 2017, agrupados por las categorías `Pública` y `Privada`, `Urbana` y `Rural`.

Categoría	Establecimientos	Maestros	Estudiantes
`Total País`	2.357	17.701	295.788
`Pública`	2.005	15.022	243.955
`Privada`	352	2.679	51.833
`Urbana`	1.281	15.463	282.523
`Pública`	929	12.784	230.690
`Privada`	352	2.679	51.833
`Rural`	1.076	2.238	13.265
`Pública`	1.076	2.238	13.265

Tabla 2.2: Ejemplo de conjunto de datos de tipo datos tabulados [1].

En las bases de datos interactivas los datos no son liberados directamente, sino que se brinda una interfaz que los usuarios pueden utilizar para enviar consultas estadísticas como sumas y promedios. Como ejemplo de uso de este formato

[1]Fuente: Anuario Estadístico 2018, INE. http://www.ine.gub.uy/anuario-estadistico

se presenta la base de datos estadística de la Comisión Económica para América Latina y el Caribe (CEPAL)[2] conocida como CEPALSTAT[3]. En la interfaz desplegada al usuario se brinda la posibilidad de seleccionar un país comprendido en América Latina o el Caribe junto con una de las siguientes categorías: `Económicos`, `Sociales`, `Ambientales`, `Género` y `Objetivos de Desarrollo del Milenio`. Como resultado, se retornan los indicadores pertinentes según la información de entrada. En la Figura 2.1 se presenta una porción del resultado obtenido al utilizar la categoría `Económicos` para Uruguay.

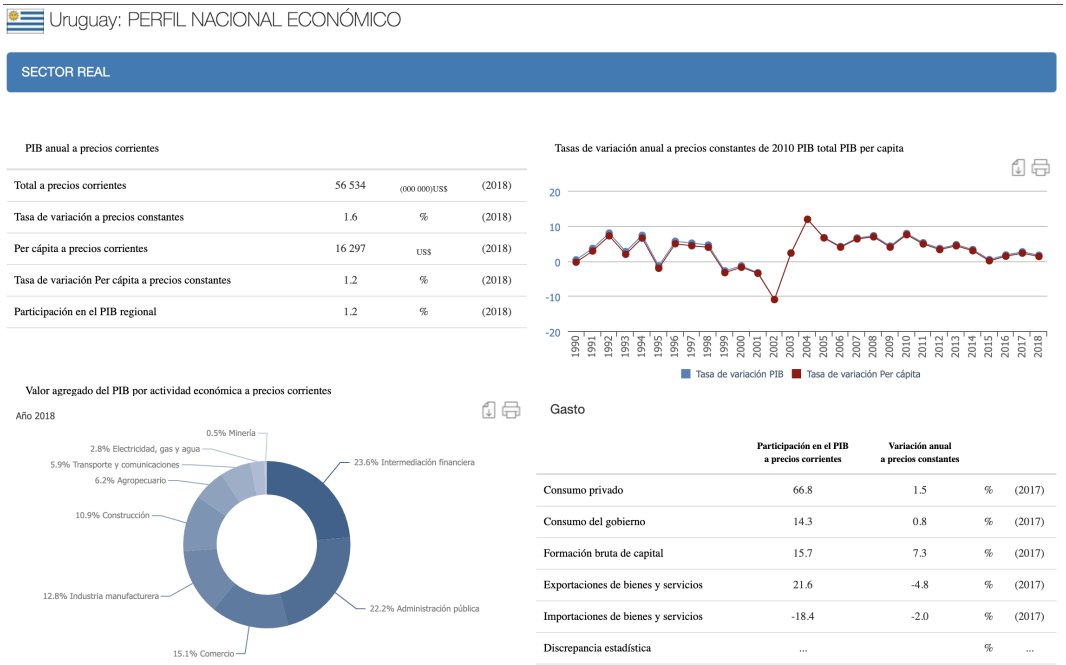


Figura 2.1: Ejemplo de conjunto de datos de tipo base de datos interactiva.

Fuente: http://interwp.cepal.org/cepalstat/Perfil_Nacional_Economico.html?pais=URY.

Las liberaciones del tipo Microdatos son las que exponen la mayor cantidad de información y, por lo tanto, son las más desafiantes en términos de la protección de la privacidad. En el resto del documento, se asume que se trabajará sobre un conjunto de datos de este tipo, conformado por n registros, donde cada uno contiene la información de un individuo concreto. A su vez, los registros están compuestos por un conjunto de m atributos. Por ejemplo, en la Tabla

[2]https://www.cepal.org/es
[3]https://estadisticas.cepal.org/cepalstat/Portada.html

2.1 el conjunto de datos está formado por diez registros donde cada uno posee tres atributos (C.I. Edad y Diagnóstico). Se brindan más detalles sobre los atributos de un conjunto de datos en la siguiente sección.

Este proyecto se focaliza en generar un conjunto de datos llamado **conjunto anonimizado**, denotado como Δ, a partir de otro llamado **conjunto original**, denotado como Θ. Al proceso de generación del conjunto Δ lo llamaremos **proceso de anonimización**. En la actualidad, Ceibal posee diferentes grupos, tanto internos como externos, que solicitan la posibilidad de hacer estudios sobre los datos que posee la organización. Cada grupo utiliza herramientas diferentes para realizar estos estudios, donde Spark, R[4] y *scikitlearn*[5] son algunas de ellas. Hoy en día, la forma que posee Ceibal para facilitar los datos a los diferentes grupos es entregando un conjunto de datos independiente a cada grupo, esta es la razón de porqué se genera un nuevo conjunto en cada proceso de anonimización.

2.1.2. Categorización de atributos

Cada atributo en el conjunto de origen describe una característica particular de un individuo. A su vez, a cada atributo se le asocia un tipo de dato, donde los tipos de datos disponibles son dos: **numéricos** o **categóricos** (que aluden a valores pertenecientes a un dominio finito). Cada atributo X tiene un dominio que se denota como $DOM(X)$. En el marco de este proyecto y dado el alcance del mismo, cada vez que se mencione el término atributo numérico se está haciendo referencia a valores numéricos enteros.
Los atributos se clasifican en las siguientes categorías no-exclusivas:

- `Identificadores:` Un atributo se considera un Identificador si permite la identificación de un individuo de forma inequívoca. Por ejemplo, la Cédula de Identidad de una persona es un identificador. Si un registro contiene algún identificador, el mismo se asocia de forma inmediata al individuo, por lo que este tipo de atributo debe ser removido durante el proceso de anonimización.

[4]https://www.r-project.org/
[5]https://scikit-learn.org/

- `Quasi-identificadores (QI)`: A diferencia de los Identificadores, los `QI` por si sólos no permiten reidentificar a un individuo específico. Sin embargo, la combinación de varios `QI` podría llevar a la identificación inequívoca de algunos individuos o a reducir significativamente la cantidad de individuos asociados a dicha combinación. Como posibles ejemplos de este tipo de atributo podemos considerar la fecha de nacimiento, el sexo y el estado civil de un individuo.
  En la práctica, definir si un atributo es o no un `QI` no es un problema sencillo ya que cualquier información que un atacante posea sobre el conjunto de datos puede ser utilizada para reidentificar a uno o más individuos.

- `Confidenciales:` Contienen información sensible de los individuos del conjunto de datos. Posibles ejemplos son la información salarial o el estado de salud.
  El objetivo principal de las técnicas de protección de Microdatos es evitar que un atacante obtenga información confidencial de un individuo específico y posibles inferencias sobre el atributo.

- `No confidenciales:` Es un atributo que no pertenece a ninguna de las categorías anteriores.

A continuación se detallan los potenciales riesgos a los que se enfrenta un conjunto de datos liberado y se establecen las distintas técnicas utilizadas para mitigarlos.

2.1.3. Riesgos de divulgación

Cuando un conjunto de datos es liberado la información del mismo queda expuesta a dos tipos de riesgos independientes: la divulgación de identidad y la divulgación de atributos.

La **divulgación de identidad** refiere a que un atacante sea capaz de asociar un registro perteneciente al conjunto Δ con uno del conjunto Θ, asociando así la identidad de un individuo con un registro del conjunto anonimizado.
Para saber que tan efectivo es el resultado de un proceso de anonimización se debe tener una forma de cuantificar el riesgo de divulgación de identidad de

un conjunto de datos. Si bien existe una serie de métricas que permiten medir la exposición a este riesgo, usualmente las más utilizadas son dos: Unicidad y Registros enlazados.

En la métrica de Unicidad, el riesgo de divulgación de identidad se mide como la probabilidad de que una combinación de valores poco usual ya se diera en el conjunto no anonimizado Θ.

Por otra parte, la técnica de Registros enlazados busca asociar un registro del conjunto Δ con un registro del conjunto Θ, el cuál podría *potencialmente* ser el registro que lo originó. El riesgo de divulgación se mide como la cantidad de correspondencias exitosas que se consigan entre los registros de ambos conjuntos. Para implementar este enfoque se aplican técnicas existentes de *record linkage* [8], para intentar reconstruir exitosamente los registros del conjunto de datos.

Mientras que la divulgación de identidad refiere a la re-identificación de un individuo dentro del conjunto, el riesgo de **divulgación de atributos** alude a que un atacante es capaz de determinar el valor de un atributo confidencial de un individuo con un gran nivel de seguridad. Por ejemplo, un atacante sería capaz de saber con seguridad que un individuo posee una determinada enfermedad.

El proceso de anonimización de datos tiene como objetivo construir un conjunto de datos que pueda ser compartido y no presente los riesgos mencionados. Para que el resultado de este proceso se considere satisfactorio, no debe ser posible reidentificar a un individuo ni conocer información confidencial de ningún sujeto específico dentro del conjunto Δ. Estas dos últimas propiedades se denominan propiedad de Anonimidad y Confidencialidad respectivamente y están directamente ligadas a los riesgos de divulgación, ya que la divulgación de identidad viola la propiedad de Anonimidad y la divulgación de atributos la de Confidencialidad. La anonimización de un conjunto de datos puede ser alcanzada utilizando distintos métodos, algunos de los cuáles son mencionados a continuación.

2.2. Métodos de protección de datos

De acuerdo a lo presentado en la sección anterior a continuación se recopilan las técnicas que se pueden aplicar para reducir los riesgos de divulgación en un conjunto de datos de tipo Microdatos. Las mismas se dividen en dos categorías, el enmascaramiento de datos y la generación de datos sintéticos.

2.2.1. Enmascaramiento

El conjunto de técnicas denominadas de **enmascaramiento** son caracterizadas por realizar modificaciones sobre el conjunto de datos original hasta que los riesgos de divulgación hayan sido reducidos por debajo de un umbral dado, usualmente determinado por alguien que tiene conocimiento sobre el dominio de interés.

La construcción del conjunto anonimizado Δ consiste en modificar cada registro del conjunto original Θ, lo que crea una relación entre los registros de los dos conjuntos ya que cada registro del conjunto anonimizado se crea a raíz de uno del original.
Las técnicas de enmascaramiento se subdividen en dos tipos, enmascaramiento con y sin perturbación. Esta división depende de si la versión anonimizada de los datos se genera mediante la perturbación en los registros originales o mediante la eliminación de información.

Enmascaramiento sin perturbación

Las técnicas que producen una eliminación parcial de información o, reducción en el nivel de detalle del conjunto Θ, se denominan de **enmascaramiento sin perturbación**. Dentro de este subconjunto de técnicas se destacan las de muestreo, generalización, codificación superior e inferior y supresión local, las cuáles serán explicadas brevemente.

Con el fin de presentar un ejemplo de cada técnica se considerará el conjunto de datos definido en la Tabla 2.3, el cuál corresponde a una versión ampliada del ejemplo presentado en la Tabla 2.1 habiendo eliminado previamente el atributo `C.I`, y se modificará según el resultado de cada técnica aplicada.

#	Sexo	Edad	Código postal	Estado civil	Diagnóstico
1	M	20	12500	Soltero/a	Diabetes
2	F	24	12700	Soltero/a	Infección
3	F	26	12600	Soltero/a	Infección
4	M	29	12900	Divorciado/a	Cáncer
5	F	45	11100	Casado/a	Diabetes
6	F	40	11300	Divorciado/a	Cáncer
7	M	50	11500	Viudo/a	Cáncer
8	M	30	11800	Soltero/a	Diabetes
9	M	35	11700	Casado/a	Cáncer
10	F	39	11200	Divorciado/a	Infección

Tabla 2.3: Conjunto de datos de ejemplo para enmascaramiento sin perturbación.

La técnica de **muestreo** establece que el conjunto de datos a liberar corresponde a una muestra aleatoria del original. Es adecuada para atributos del tipo categórico pero no para atributos numéricos, ya que los mismos proveen información muy específica y, debido a que no realiza perturbaciones sobre los datos, es altamente probable que se encuentren coincidencias únicas con la muestra publicada y otras fuentes de datos externas. Por lo tanto, de poseer atributos numéricos, esta técnica se debe combinar con otras o eliminar este tipo de atributos. Un ejemplo de aplicar muestreo en el conjunto de datos definido, habiendo eliminado previamente los atributos numéricos, se presenta en la Tabla 2.4.

#	Sexo	Estado civil	Diagnóstico
1	M	Soltero/a	Diabetes
2	F	Soltero/a	Infección
3	F	Soltero/a	Infección
8	M	Soltero/a	Diabetes

Tabla 2.4: Ejemplo de aplicación de la técnica de muestreo sobre el conjunto de datos definido en la Tabla 2.3.

En la técnica de **generalización**, también conocida como **Registro Global** en el contexto del *Control de Divulgación Estadística* (SDC según su nombre en Inglés) [9], los valores de los atributos son agrupados siguiendo estrategias definidas según el tipo de atributo. Para los atributos categóricos, se crean categorías a partir de la combinación de otras menos específicas y los valores de los registros son remplazados por esas nuevas categorías. Por ejemplo, en el conjunto de la Tabla 2.3, el atributo `Estado civil` presenta los valo-

res `Soletero/a`, `Casado/a`, `Divorciado/a` o `Viudo/a` y se decide crear una categoría que contenga los valores [`Divorciado/a o Viudo/a`]. Por tanto, los valores que podrá tomar el atributo en el conjunto de datos anonimizado serán tres: `Soletero/a`, `Casado/a` o [`Divorciado/a o Viudo/a`]. Por otro lado, los atributos numéricos son reemplazados por una versión discretizada del valor original, donde típicamente se utilizan conjuntos de valores que representan un rango del dominio original. Por ejemplo, si el atributo `Edad` posee el valor `24`, el mismo podría ser reemplazado por el intervalo `[20 - 29]`.
Comúnmente, estas categorías son definidas por el usuario y cada una posee un significado semántico dentro del contexto de cada atributo. Por ejemplo, el significado semántico de la categoría [`Divorciado/a o Viudo/a`] podría referir a individuos que estuvieron casados en algún momento y ya no lo están. Un posible resultado de aplicar generalización en el conjunto de ejemplo se presenta en la Tabla 2.5.

#	**Sexo**	**Edad**	**Código postal**	**Estado civil**	**Diagnóstico**
1	M	[20 - 29]	12***	Soltero/a	Diabetes
2	F	[20 - 29]	12***	Soltero/a	Infección
3	F	[20 - 29]	12***	Soltero/a	Infección
4	M	[20 - 29]	12***	[Divorciado/a o Viudo/a]	Cáncer
5	F	[40 - 50]	11***	Casado/a	Diabetes
6	F	[40 - 50]	11***	[Divorciado/a o Viudo/a]	Cáncer
7	M	[40 - 50]	11***	[Divorciado/a o Viudo/a]	Cáncer
8	M	[30 - 39]	11***	Soltero/a	Diabetes
9	M	[30 - 39]	11***	Casado/a	Cáncer
10	F	[30 - 39]	11***	[Divorciado/a o Viudo/a]	Infección

Tabla 2.5: Ejemplo de aplicación de la técnica generalización al conjunto de datos definido en la Tabla 2.3.

Existe un caso particular de esta técnica denominada **codificación superior e inferior** que solo puede ser aplicada cuando se cuenta con atributos que pueden ser ordenados (presentan un orden total). La misma se basa en que los valores que se encuentran por encima de un umbral determinado se consideran valores superiores y son asignados a la misma categoría. Análogamente, se definen los valores inferiores como aquellos que se encuentran por debajo del umbral. Para presentar el resultado de aplicar esta técnica sobre el conjunto de ejemplo se eliminaron los atributos categóricos ya que los mismos no presentan

un orden total. Para los atributos numéricos `Edad` y `Código postal`, se definen los umbrales `35` y `12000` respectivamente. Los resultados son desplegados en la Tabla 2.6.

#	**Edad**	**Código postal**	**Diagnóstico**
1	< 35	≥ 12000	Diabetes
2	< 35	≥ 12000	Infección
3	< 35	≥ 12000	Infección
4	< 35	≥ 12000	Cáncer
5	≥ 35	< 12000	Diabetes
6	≥ 35	< 12000	Cáncer
7	≥ 35	< 12000	Cáncer
8	< 35	< 12000	Diabetes
9	< 35	< 12000	Cáncer
10	≥ 35	< 12000	Infección

Tabla 2.6: Ejemplo de aplicación de la técnica codificación superior e inferior sobre el conjunto de datos definido en la Tabla 2.3.

La última técnica de enmascaramiento sin perturbación que analizaremos es la denominada **supresión local**. Se basa en la eliminación de elementos del domino de ciertos atributos con el fin de aumentar la cantidad de registros que comparten determinados valores en sus atributos, por lo que su aplicación tiene sentido si está orientada a atributos categóricos. Para eliminar estos valores, se reemplazan todas las apariciones de los mismos por un valor especial, diferente de los demás valores del dominio.

#	**Sexo**	**Edad**	**Código postal**	**Estado civil**	**Diagnóstico**
1	M	20	12500	Soltero/a	Diabetes
2	F	24	12700	Soltero/a	Infección
3	F	26	12600	Soltero/a	Infección
4	M	29	12900	Divorciado/a	Cáncer
5	F	45	11100	`?`	Diabetes
6	F	40	11300	Divorciado/a	Cáncer
7	M	50	11500	`?`	Cáncer
8	M	30	11800	Soltero/a	Diabetes
9	M	35	11700	`?`	Cáncer
10	F	39	11200	Divorciado/a	Infección

Tabla 2.7: Ejemplo de aplicación de la técnica supresión local al conjunto de datos definido en la Tabla 2.3.

Los resultados de aplicar la técnica suprimiendo los valores `Viudo/a` y

`Casado/a` y remplazándolos por el valor `?` en el atributo `Estado civil`, se encuentran en la Tabla 2.7.

Enmascaramiento con perturbación

Este conjunto de técnicas se caracteriza por el hecho de que los datos del conjunto original Θ son alterados antes de su publicación. El mayor desafío presentado al aplicarlas, es el poder preservar la mayor cantidad de información estadísticamente útil del conjunto Θ en el conjunto Δ. Dentro de este subconjunto de técnicas las más destacadas son la adición de ruido y la microagregación. Si bien este tipo de técnicas no serán el centro de la investigación, creemos conveniente mencionarlas, por lo que a continuación se brindará un breve resumen de cada una.

Una forma posible de lograr la perturbación se basa en agregar ruido aleatorio en el conjunto Θ, esto es, modificar las distribuciones estadísticas de los atributos de forma tal que el valor estadístico se preserve lo más posible y que los datos presenten cierto nivel de protección. A esta técnica se le denomina **adición de ruido** y se debe considerar que las propiedades estadísticas del ruido agregado determinan el resultado final.

La técnica de **microagregación** es aplicable exclusivamente a atributos numéricos y la idea detrás de ella es que la privacidad de un conjunto de datos se preserva si los datos publicados contienen grupos de registros que contengan al menos cierta cardinalidad, definida por el usuario. Los valores de los atributos son reemplazados por valores agregados dentro de cada grupo. En otras palabras, podríamos considerar un conjunto de datos donde sus registros se combinan para formar grupos registros, y para cada atributo, se calcula el valor promedio y luego es utilizado para reemplazar los valores de los registros originales.

Las tres técnicas disponibles para el enmascaramiento con perturbación se ilustran en la Figura 2.2a.

Hasta el momento nos hemos concentrado exclusivamente en las técnicas de enmascaramiento, las cuáles generan cierto grado de dependencia entre el con-

junto Θ y Δ. Sin embargo, existe otra familia de técnicas conocida como generación de datos sintéticos las cuáles brindan el mayor nivel de independencia entre los conjuntos. La misma se detalla en la siguiente sección.

2.2.2. Generación de datos sintéticos

En las técnicas de generación de datos sintéticos el conjunto resultante Δ está conformado por registros simulados que siguen la distribución probabilística del conjunto Θ pero no derivan directamente de él. Por esto se caracterizan por brindar el mayor grado de independencia posible entre los conjuntos de datos. La única relación entre ellos es que el conjunto Δ preserva algunas propiedades estadísticas de Θ.

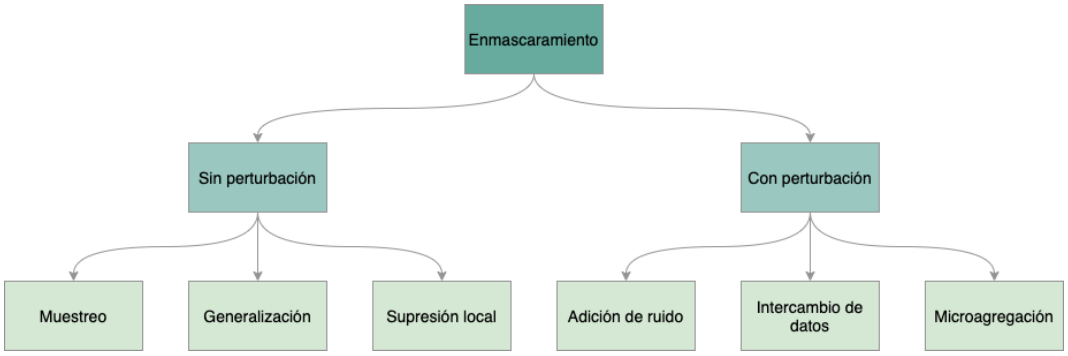
Los conjuntos de datos sintéticos pueden ser de tres tipos: completamente sintéticos, parcialmente sintéticos o híbridos. En un conjunto de datos **completamente sintéticos** se sintetizan todos los valores de los atributos de cada registro. Dado que los registros del conjunto generado no están relacionados directamente con los del conjunto Θ, el riesgo de divulgación es relativamente bajo. Una variante de la categoría anterior son los conjuntos de datos **parcialmente sintéticos**, donde solamente se sintetizan los valores de los atributos cuya divulgación sea más riesgosa. Por último, los conjuntos de datos **híbridos** resultan de mezclar algunos registros del conjunto Θ con un conjunto completamente sintético generado a partir del mismo. Es el tipo de datos sintéticos que presenta la menor protección dado que contiene registros del conjunto original. La complejidad del proceso de generar datos sintéticos provoca que este conjunto de técnicas sean utilizadas con menor frecuencia que las técnicas de enmascaramiento.

A modo de síntesis, en la Figura 2.2 se presenta un esquema con todas las técnicas presentadas.

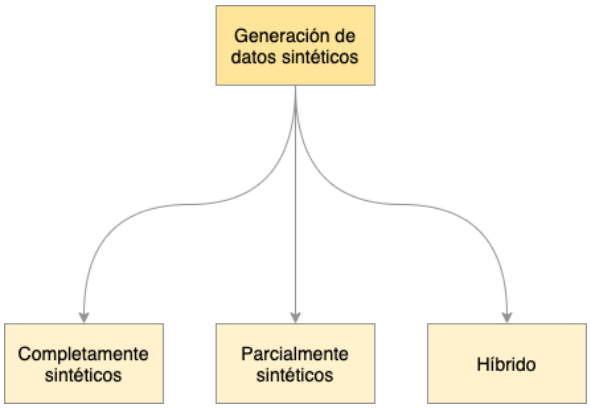
Se debe considerar que si bien aplicar una técnica de anonimización brinda protección a los datos, al mismo tiempo quita precisión y detalles del conjunto. La cantidad de información perdida varía según la técnica y los parámetros usados, pero siempre es deseable que se preserve la mayor cantidad de información posible, por lo que será necesario utilizar algún mecanismo para

cuantificar esta pérdida con el fin de minimizarla.

Teniendo una visión general de los métodos más usados para la anonimización de un conjunto de datos, de aquí en más abordaremos el modelo de privacidad conocido como *k-anonymity*, el cuál puede ser implementado tanto con técnicas de enmascaramiento sin y con perturbación.



(a) Métodos de enmascaramiento.



(b) Métodos de generación de datos sintéticos.

Figura 2.2: Métodos de protección de datos.

2.3. El modelo de privacidad *k-anonymity*

2.3.1. Aspectos generales

k-anonymity es un modelo de privacidad propuesto por Latanya Sweeney en el año 2002 [6] cuyo objetivo es evitar la reidentificación básandose en el valor en un conjunto de atributos. Si bien el nivel de protección que ofrece es

limitado, su popularidad ha crecido debido a que es muchas veces considerado como requisito mínimo para limitar el riesgo de divulgación de identidad. El modelo asume que se conocen los atributos que pueden ser utilizados para reidentificar un registro y asegura que cada uno sea indistinguible de al menos k registros. En consecuencia, la probabilidad de reidentificar a un individuo en un conjunto que satisface la condición de *k-anonymity* es siempre igual o inferior a $\frac{1}{k}$, siendo k un parámetro dado.

Denominaremos como Quasi-identificador (de ahora en más abreviado como `QI`) al conjunto de atributos que pueden ser utilizados para reidentificar un registro. Este conjunto está conformado por atributos del tipo `Quasi-identificador`, definidos en la Sección 2.1.2. Para que un conjunto de datos cumpla con la condición del modelo *k-anonymity*, toda combinación de valores del `QI` en el conjunto Δ, debe ser compartida por al menos k registros. De esta condición se desprende que el conjunto Δ puede ser particionado en subconjuntos de registros que comparten la misma combinación de valores en `QI`, por lo que llamaremos a estos conjuntos **clases de equivalencia**.

Definición 2.3.1 ***Clase de equivalencia*** *[5] Dado un `QI`, la clase de equivalencia de un registro es el subconjunto de registros que comparten el mismo valor en todos los atributos pertenecientes a `QI`.*

Por tanto, se considera que un conjunto de datos cumple la condición de *k-anonymity* si, y sólo si, la clase de equivalencia de cada registro del conjunto tiene por lo menos k registros (ver Definición 2.3.2).

Definición 2.3.2 ***Condición de k-anonymity.*** *Dado el conjunto de clases de equivalencia Ω de un conjunto de datos Δ, se dice que Δ cumple la condición de k-anonymity si se cumple que*

$$\{\forall \omega \in \Omega, k \leqslant \mid \omega \mid\}$$

Con el fin de ilustrar los conceptos hasta ahora mencionados consideremos el conjunto de datos definido en la Tabla 2.8, el cuál corresponde a una muestra de registros médicos que contiene atributos del tipo `Identificador` (número de Cédula de Identidad), `QI` (Edad y Código postal) y `Confidenciales` (Diagnóstico).

	Identificador	*Quasi-identificadores*		*Confidencial*
#	**C.I.**	**Edad**	**Código postal**	**Diagnóstico**
1	4.928.457-3	20	12500	Diabetes
2	4.489.324-1	24	12700	Infección
3	4.213.983-8	26	12600	Infección
4	4.382.093-4	29	12900	Cáncer
5	3.382.572-3	45	11100	Diabetes
6	3.528.457-3	40	11300	Cáncer
7	2.993.098-9	50	11500	Cáncer
8	4.288.739-7	30	11800	Diabetes
9	4.002.458-1	35	11700	Cáncer
10	3.924.492-6	39	11200	Infección

Tabla 2.8: Muestra de registros médicos original.

Si consideramos el valor del parámetro k como `k = 3`, un posible resultado de aplicar el modelo *k-anonymity* sobre el conjunto de datos definido en la Tabla 2.8, es el presentado en la Tabla 2.9. Donde los atributos de tipo `Identificador` fueron suprimidos y los `QI` reemplazados por una versión más general de los valores originales. Las clases de equivalencia son tres y quedan definidas por los registros {1, 2, 3, 4}, {5, 6, 7} y {8, 9, 10}.

	Identificador	*Quasi-identificadores*		*Confidencial*
#	**C.I.**	**Edad**	**Código postal**	**Diagnóstico**
1	*	[20 - 29]	12***	Diabetes
2	*	[20 - 29]	12***	Infección
3	*	[20 - 29]	12***	Infección
4	*	[20 - 29]	12***	Cáncer
5	*	[40 - 50]	11***	Diabetes
6	*	[40 - 50]	11***	Cáncer
7	*	[40 - 50]	11***	Cáncer
8	*	[30 - 39]	11***	Diabetes
9	*	[30 - 39]	11***	Cáncer
10	*	[30 - 39]	11***	Infección

Tabla 2.9: Muestra de registros médicos anonimizada.

Existen múltiples implementaciones que permiten alcanzar la condición de *k-anonymity* dentro de un conjunto de datos, en [10] se presenta una revisión exhaustiva de ellas. En la siguiente sección se detallan dos.

2.3.2. Algoritmos de *k-anonymity*

A continuación se explican dos algoritmos que se pueden utilizar para implementar el modelo de privacidad *k-anonymity*, uno basado en Generalización y Supresión y otro basado en *Top-Down Specialization*.

Generalización y Supresión

Generalización y Supresión es una técnica de enmascaramiento sin perturbación. Para alcanzar la condición de *k-anonymity* en un conjunto de datos, esta técnica reduce la precisión de los valores de los atributos remplazándolos por versiones más generales. Antes de definir formalmente qué implica remplazar un valor por una versión más general, consideraremos el ejemplo de la Figura 2.3. En ella se muestra un árbol balanceado cuyos nodos contienen todos los valores que podrá presentar el atributo `Edad` en el conjunto Δ. Se puede apreciar que los valores más específicos del árbol se encuentran en sus hojas y corresponden a todos los valores que presenta el atributo en la Tabla 2.8. El resto de los nodos no contienen valores específicos sino que contienen rangos que agrupan los valores del nivel inmediatamente inferior.

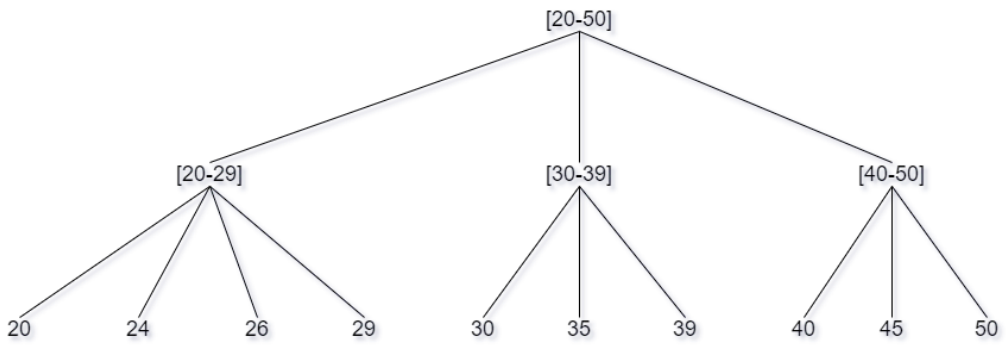


Figura 2.3: Ejemplo de árbol taxonómico para el atributo Edad.

Por ejemplo, los valores {`20, 24, 26, 29`} están contemplados en el rango de valores `[20 - 29]`. A su vez, los rangos {`[20 - 29], [30 - 39], [40 - 50]`} están contenidos en el valor del nodo raíz, `[20 - 50]`. En este ejemplo, diremos que rango `[20 - 29]` es un valor más general que los de sus nodos hijos {`20, 24, 26, 29`} y que `[20 - 50]` es más general que cualquiera de los intervalos {`[20 - 29], [30 - 39], [40 - 50]`}.

A los árboles que contienen todos los valores que podrá presentar un deter-

minado atributo, como el de la Figura 2.3, los denominaremos **árboles de taxonomía**. Cada atributo presenta su propio árbol taxonómico y como regla general diremos que cuanta mayor sea la altura de un nodo dentro de un árbol, más general será su valor, estableciendo así una relación de orden parcial entre los nodos del árbol. La estructura de estos árboles es usualmente definida por el usuario y son creados antes de comenzar el proceso de anonimización de un conjunto de datos.

Al proceso de remplazar valores de atributos por una versión más general lo llamaremos **aplicar una generalización** a un conjunto de datos. El mismo refiere a cambiar valores que se encuentran a una determinada altura dentro del árbol taxonómico correspondiente, por el valor de un nodo que se encuentra en un nivel superior. Por ejemplo, el atributo `Edad` en la Tabla 2.8 presenta valores cuya altura dentro del árbol taxonómico de la Figura 2.3 es cero. Sin embargo, el conjunto anonimizado de la Tabla 2.9 presenta valores cuya altura es uno. Dentro de un árbol taxonómico existen varias generalizaciones posibles. Por ejemplo, en la Tabla 2.9 el atributo `Edad` se sustituyó con valores del nivel inmediatamente superior, pero podría haber sido sustituido por el valor del nodo raíz (`[20 - 50]`), y aún así cumpliría con la condición de *k-anonymity*. Análogo al concepto de aplicar una generalización, se define aplicar una **especialización** como el proceso de remplazar los valores de un atributo por otros que presenten una menor altura dentro del árbol. La técnica de Generalización y Supresión requiere que exista un árbol taxonómico para cada atributo perteneciente a `QI`. A continuación se proveerá la definición formal de los conceptos abarcados hasta ahora en la sección.

Definición 2.3.3 ***Preorden parcial*** *Un preorden parcial es una relación binaria $\sqsubseteq$ sobre un conjunto que cumple las propiedades reflexiva, antisimétrica y transitiva.*

Definición 2.3.4 ***Árbol taxonómico*** *Sea X un atributo y $DOM(X)$ su dominio de valores. Un árbol taxonómico es un preorden parcial sobre un superconjunto $DOM(\sqsubseteq)$ de $DOM(X)$ tal que:*

$$\{\forall s \subseteq DOM(\sqsubseteq), \exists z \in DOM(\sqsubseteq), \forall x \in s : x \leq z\}$$

De esta definición se deprende que existe un valor $\tau \in DOM(\sqsubseteq)$, raíz del árbol taxonómico, que cumple que $\forall z \in DOM(\sqsubseteq), \tau \sqsubseteq z$

El objetivo es seleccionar un nivel dentro de cada árbol taxonómico de los atributos del `QI` de forma tal que, sustituyendo los valores del conjunto Θ por los del nivel seleccionado en cada árbol, el resultado sea un conjunto Δ que cumpla la condición de *k-anonymity*. La selección de los niveles se realiza a través de la aplicación de generalizaciones sobre los árboles de los atributos del `QI`. Cada vez que se aplica una generalización a un atributo y sus valores pasan a ser más generales, el conjunto de datos pierde información concreta. En el ejemplo de la Tabla 2.9 se sabe que cuatro individuos contienen entre `[20 - 29]` años, tres entre `[30 - 39]` y tres entre `[40 -50]` años. Si aplicamos una generalización al atributo `Edad` y todos sus valores pasaran a ser `[20 - 50]`, solamente sabríamos que se tienen diez individuos que tienen entre veinte y cincuenta años, perdiendo el nivel de detalle que teníamos antes de aplicar la generalización. Por lo tanto, el objetivo al aplicar una generalización, es aplicar aquella que genere la menor pérdida de información posible y mantener la condición de *k-anonymity* en el conjunto. Encontrar esta generalización es una de las mayores desventajas de la técnica de Generalización y Supresión, ya que se deben considerar todas las generalizaciones posibles de todos los atributos del `QI`. Este problema está demostrado que es un problema *NP-hard* [11].

Teniendo en cuenta este resultado teórico, la técnica *Top-Down Specialization* aplica una heurística que permite encontrar un compromiso aceptable entre pérdida de información y tiempo de ejecución. A continuación se describe cómo utilizar esta técnica para alcanzar la condición de *k-anonymity* de un conjunto de datos.

Top-Down Specialization

En contraposición a la técnica de Generalización y Supresión que reemplaza valores por otros más generales, *Top-Down Specialization* (TDS) propone alcanzar la condición de *k-anonymity* partiendo del valor más general de cada atributo (raíz de cada árbol taxonómico) y aplicar especializaciones para obtener valores más específicos.
Esta técnica plantea una ventaja principal respecto a la técnica anterior. Esta ventaja refiere a que la ejecución de TDS es más eficiente debido a que el espacio de soluciones que debe considerar es menor que en la técnica de Generalización y Supresión, en el resto de la sección se brindan más detalles sobre

el espacio de soluciones. El nombre *Top-Down* refiere a que se comienza desde el nivel más alto de los árboles taxonómicos y luego se especializan (se baja en el árbol taxonómico).
De ahora en más se abrevia el término de árboles taxonómicos como `TT` debido a su nombre en Inglés (*Taxonomy tree*). Dado que estamos asumiendo que cada `TT` es un árbol balanceado, diremos que todos los valores que se encuentran a la misma altura dentro de un `TT`, definen un **corte** en el TT.

Definición 2.3.5 ***Corte*** *Consideremos los conjuntos* $\Gamma_0, \Gamma, \Gamma_1 \in DOM(\sqsubseteq)$. *Un corte* Γ *en un árbol taxonómico del atributo* X *es:*

$$\{\forall z \in \Gamma_0, \exists x \in \Gamma : z \sqsubset x\}$$

$$\{\forall y \in \Gamma, \exists x \in \Gamma_1 : x \sqsubset y\}$$

Donde $\Gamma_0 \cup \Gamma \cup \Gamma_1 = DOM(\sqsubseteq)$ *y*

$$\begin{cases} \Gamma_0 \cap \Gamma = \emptyset \\ \Gamma_1 \cap \Gamma = \emptyset \\ \Gamma_0 \cap \Gamma_1 = \emptyset \end{cases}$$

Además, se debe cumplir que

$$DOM(X) \subseteq \Gamma_0$$

Con el fin de acotar el espacio de soluciones que se deben evaluar durante la ejecución del algoritmo TDS, se considerarán solamente aquellos cortes cuyos nodos tengan la misma altura dentro de un árbol taxonómico. Como ejemplo, consideremos el árbol taxonómico del atributo `Edad` de la Figura 2.4, donde se muestra un posible corte $\Gamma = \Gamma_{Edad_1}$ para el árbol y los conjuntos $\Gamma_0 = \Gamma_{Edad_0}$ y $\Gamma_1 = \Gamma_{Edad_2}$.

Cabe mencionar que si se considera el corte Γ formado por la raíz del árbol taxonómico, el conjunto Γ_0 es el conjunto vacío.

Para cambiar el corte considerado para un atributo por otro, se debe aplicar una especialización. Formalmente, una especialización se define en la Definición 2.3.6.

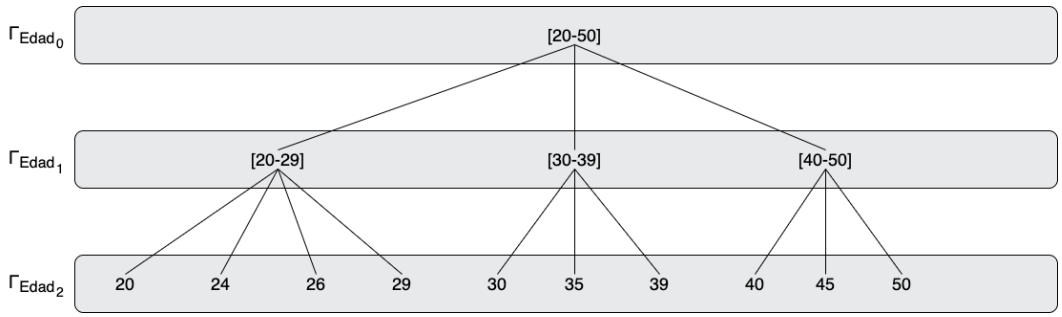


Figura 2.4: Ejemplo de corte para el árbol taxonómico para el atributo Edad.

Definición 2.3.6 ***Especialización (de un corte)*** *Dado un árbol taxonómico y dos cortes* Γ_1, Γ_2 *en dicho árbol, se dice que* Γ_1 *es una especialización de* Γ_2 *(*$\Gamma_1 \sqsubseteq \Gamma_2$*) si:*

$$\Gamma_1 \sqsubseteq \Gamma_2 : \{\forall x \in \Gamma_1, \exists y \in \Gamma_2 : x \sqsubseteq y\}$$

Diremos además que una **especialización válida** es una especialización que, al ser aplicada, produce un conjunto de datos que cumple la condición de *k-anonymity*.

Al igual que en la técnica de Generalización y Supresión, cada atributo tiene su propio `TT` y se debe conocer cuál es el corte considerado para cada atributo en cada momento. Por lo tanto, se define el concepto de **nivel de anonimicidad**, abreviado de ahora en más como `AL` por su nombre en Inglés (*Anonymization Level*).

Definición 2.3.7 ***Nivel de anonimicidad (AL).*** *[12] Se denomina nivel de anonimicidad al conjunto de cortes de todos los atributos.*

$$AL = (\Gamma_1, \Gamma_2, ..., \Gamma_m)$$

Donde Γ_i, $1 \leq i \leq m$, *refiere al corte considerado para* TT_i, *siendo* TT_i *el árbol taxonómico considerado para el atributo* X_i.

Una vez que se tiene definido un nivel de anonimización, diremos que **aplicar un AL a un conjunto de datos** corresponde a reemplazar el valor de cada atributo, de cada registro del conjunto de datos original Θ, por el nodo correspondiente en el corte seleccionado para ese atributo dentro del `TT` en el AL considerado. Vale mencionar que si el corte seleccionado refiere al nivel más

bajo del TT, el nodo correspondiente es aquel nodo que presenta el mismo valor que el atributo. En caso contrario, el nodo corresponde al único ancestro del valor del atributo que pertenece al corte seleccionado. Formalmente, se define en la Definición 2.3.8.

Definición 2.3.8 ***Aplicar un AL a un conjunto de datos*** *Se considera el conjunto de datos original* Θ *donde el conjunto* QI *está compuesto por* m *atributos, el nivel de anonimicidad se define como* $AL = (\Gamma_1, ..., \Gamma_m)$*, donde* $\Gamma_i, i = 1, ..., m$ *refiere al corte considerado para el atributo* i*. Diremos que el resultado de aplicar un AL al conjunto de datos* Θ *es un conjunto de datos* Δ*, tal que*

$$\Delta = \{\delta \,|\, \forall \theta \in \Theta, \delta = \nu(\theta)\}$$

Donde $\nu(\theta) = (\psi_1, ..., \psi_m)$ *tal que*

$$\forall x_i \in \theta, i = 1, ..., m : \psi_i \in \Gamma_i \text{ y existe un camino entre } \psi_i \text{ y } x_i \text{ en } TT_i$$

Para ilustrar la técnica con un ejemplo, consideraremos nuevamente el conjunto de datos definido en la Tabla 2.8, los árboles taxonómicos de los atributos Edad y Código postal definidos en las Figuras 2.3 y 2.5 respectivamente y el parámetro de anonimización k = 3. La técnica parte de los valores más generales de cada TT, siendo el AL inicial $AL_0 = (\{[20 - 50]\}, \{1 * * * *\}) = (\Gamma_{Edad_0}, \Gamma_{Postal_0})$ y donde cuya aplicación sobre el conjunto de datos se muestra en la Tabla 2.10. Esto provoca que el conjunto de datos comience el proceso de anonimización cumpliendo la condición de *k-anonymity*.

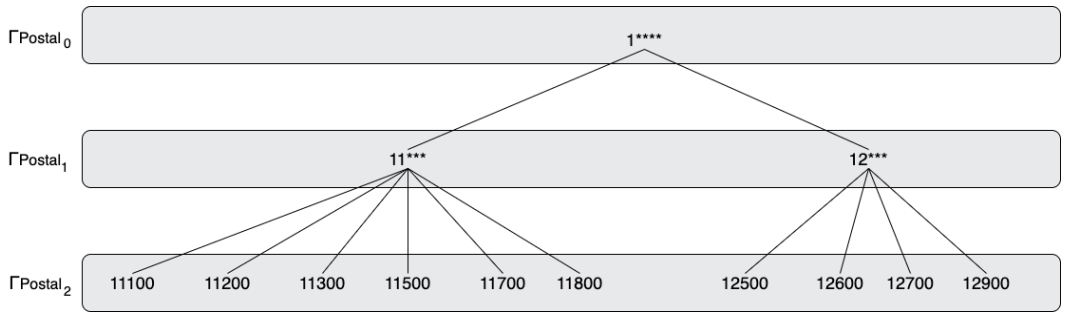


Figura 2.5: Ejemplo árbol taxonómico para el atributo Código postal.

Luego, se evalúa si existe alguna especialización válida para alguno de los atributos del QI. En este caso, considerando los árboles taxonómicos de las figuras

	Identificador	*Quasi-identificadores*		*Confidencial*
#	**C.I.**	**Edad**	**Código postal**	**Diagnóstico**
1	*	[20 - 50]	1****	Diabetes
2	*	[20 - 50]	1****	Infección
3	*	[20 - 50]	1****	Infección
4	*	[20 - 50]	1****	Cáncer
5	*	[20 - 50]	1****	Diabetes
6	*	[20 - 50]	1****	Cáncer
7	*	[20 - 50]	1****	Cáncer
8	*	[20 - 50]	1****	Diabetes
9	*	[20 - 50]	1****	Cáncer
10	*	[20 - 50]	1****	Infección

Tabla 2.10: Muestra de registros médicos anonimizada según la técnica TDS sin aplicar ninguna especialización.

2.3 y 2.5, existen dos posibles especializaciones válidas, una para el atributo `Edad` y otra para `Código postal`. Comenzaremos por aplicar la primera de ellas, generando un nuevo AL que corresponde a $AL_1 = (\Gamma_{Edad_1}, \Gamma_{Postal_0})$. El resultado de su aplicación sobre el conjunto de datos se muestra en la Tabla 2.11.

	Identificador	*Quasi-identificadores*		*Confidencial*
#	**C.I.**	**Edad**	**Código postal**	**Diagnóstico**
1	*	[20 - 29]	1****	Diabetes
2	*	[20 - 29]	1****	Infección
3	*	[20 - 29]	1****	Infección
4	*	[20 - 29]	1****	Cáncer
5	*	[40 - 50]	1****	Diabetes
6	*	[40 - 50]	1****	Cáncer
7	*	[40 - 50]	1****	Cáncer
8	*	[30 - 39]	1****	Diabetes
9	*	[30 - 39]	1****	Cáncer
10	*	[30 - 39]	1****	Infección

Tabla 2.11: Muestra de registros médicos anonimizada según la técnica TDS habiendo especializado el atributo `Edad`.

Nuevamente se debe evaluar si existe alguna especialización válida para alguno de los atributos del `QI`. En este caso, la única que queda es la del atributo `Código postal` y, al aplicarla, provoca que el nuevo AL sea $AL_2 = (\Gamma_{Edad_1}, \Gamma_{Postal_1})$. El resultado de aplicar este último AL al conjunto de datos

se despliega en la Tabla 2.12 y, dado que no existen más especializaciones válidas, éste también corresponde al resultado final del proceso de anonimización.

	Identificador	*Quasi-identificadores*		*Confidencial*
#	**C.I.**	**Edad**	**Código postal**	**Diagnóstico**
1	*	[20 - 29]	12***	Diabetes
2	*	[20 - 29]	12***	Infección
3	*	[20 - 29]	12***	Infección
4	*	[20 - 29]	12***	Cáncer
5	*	[40 - 50]	11***	Diabetes
6	*	[40 - 50]	11***	Cáncer
7	*	[40 - 50]	11***	Cáncer
8	*	[30 - 39]	11***	Diabetes
9	*	[30 - 39]	11***	Cáncer
10	*	[30 - 39]	11***	Infección

Tabla 2.12: Muestra final de registros médicos anonimizada según la técnica TDS.

Cabe mencionar que existen varios tipos de especializaciones. En este proyecto utilizaremos la denominada *full-domain* [13], que al aplicar una especialización a un corte Γ_1, retorna el corte Γ_2 definido por nivel inmediatamente inferior dentro del `TT`. Por ejemplo, si se aplica la función de especialización *full-domain* al corte $\Gamma_{Edad_1} = \{$`[20 - 29], [30 - 39], [40 - 50]`$\}$ del `TT` del atributo `Edad`, el resultado es el corte $\Gamma_{Edad_2} =$ {`20, 24, 26, 29, 30, 34, 39, 40, 45, 50`}.

Métrica para seleccionar la mejor especialización

La técnica TDS parte de los valores más generales y aplica especializaciones sobre los atributos del conjunto `QI` hasta que la condición de *k-anonymity* deje de cumplirse. Los algoritmos que implementan esta técnica son iterativos y, en cada iteración, seleccionan la mejor especialización dentro del conjunto de especializaciones válidas. Para poder definir cuál es la mejor especialización dentro de este último, es necesario definir una métrica que permita generar un indicador para comparar las aplicaciones de las especializaciones. Recordemos que cuando se aplica una especialización, se gana información en el conjunto de datos (se pasa de un valor general a valores más concretos), pero al mismo tiempo se pierde anonimización, es por este motivo que la métrica utilizada idealmente debe ser capaz de cuantificar estos dos aspectos de un conjunto de

datos. Este es un problema de optimización del tipo MinMax.

Dentro del conjunto de métricas disponibles que permiten cuantificar la ganancia de información obtenida al aplicar una especialización a un conjunto de datos, utilizaremos la denominada `Ganancia de Información`. La misma se define a continuación.

Definición 2.3.9 ***Ganancia de información*** *[12] Denominaremos* `IG(spec)` *a la ganancia de información luego de aplicada la especialización spec. La misma se calcula utilizando información estadística del conjunto de datos y toma en cuenta el estado del mismo previo a aplicar la especialización spec representado en R_p, y luego de aplicarla, representado con R_c.*

$$IG(spec) = I(R_p) - \sum_{c \in spec(p)} \left(\frac{|R_c|}{|R_p|} \right) I(R_c)$$

$I(R_x)$ refiere a la entropía del conjunto de datos x y es calculada mediante la siguiente expresión:

$$I(R_x) = - \sum_{sv \in SV} \left(\frac{|(R_x, sv)|}{|R_x|} \right) \log_2 \left(\frac{|(R_x, sv)|}{|R_x|} \right)$$

Donde R_x refiere los registros del conjunto de datos original, que contienen valores de atributos que pueden ser generalizados al valor x. $|R_x|$ es la cantidad de registros del conjunto R_x. El término $|(R_x, sv)|$ refiere a la cantidad de registros del conjunto R_x, que poseen el valor confidencial sv.

El segundo aspecto que debe ser considerado al momento de aplicar una especialización, es la pérdida de privacidad. Esta propiedad se cuantifica utilizando la métrica de `Pérdida de Privacidad` denotada `PL`, la cuál al igual que la métrica anterior, se calcula utilizando información estadística del conjunto de datos. Previo a definir `PL` es necesario introducir la siguiente definición.

Definición 2.3.10 ***Anonimidad.*** *[12] La anonimidad* `A` *de un conjunto de datos se define como $A = min_{\omega \in \Omega}\{|\omega|\}$, donde Ω es el conjunto de clases de equivalencias del conjunto de datos (definidas en 2.3.1) y $|\omega|$ el tamaño de la clase ω. Por lo tanto A es igual al tamaño de la clase más pequeña.*

Considerando la definición de anonimidad brindada en la Definición 2.3.10, se denota A_p^{spec} como la anonimidad del conjunto de datos previo a aplicar la

especialización *spec*. Por otra parte, A_c^{spec} refiere a la anonimidad luego de que la especialización fue aplicada. La `Pérdida de Privacidad` causada por una especialización se calcula utilizando la Fórmula 2.1.

$$PL(spec) = A_p^{spec} - A_c^{spec} \tag{2.1}$$

Si bien las métricas `Ganancia de Información` y `Pérdida de Privacidad` se definen de manera separada, existe una métrica de compensación que considera ambos aspectos englobados en un solo indicador. El mismo se denomina *Information Gain per Privacy Loss* (IGPL) [12], y será el empleado para seleccionar la mejor especialización en cada iteración del algoritmo. Esta métrica se calcula mediante la expresión presentada en la Fórmula 2.2.

$$IGPL(spec) = \frac{IG(spec)}{PL(spec) + 1} \tag{2.2}$$

A continuación se brindan detalles de los artículos utilizados como base para la implementación del proyecto, donde cada uno de ellos plantea una alternativa de implementación de la técnica TDS. Además, se mencionan otros algoritmos que también permiten alcanzar la condición de *k-anonymity* mediante otras técnicas.

2.3.3. Trabajos relacionados

Top-down specialization

El algoritmo *Top-Down Specialization* [14] determina una versión generalizada del conjunto de datos tal que la información confidencial es enmascarada. El mismo presenta dos etapas. En la primera se realiza un preprocesamiento del conjunto de datos, en donde se eliminan los identificadores y todos aquellos atributos que no se consideren parte del `QI`, y se agrupan todos los registros que compartan el mismo valor para los atributos del `QI`. En una segunda etapa, se ejecuta un proceso iterativo mediante el cuál se especifican los árboles taxonómicos hasta violar la condición de *k-anonymity*. El resultado de la iteración consiste en un conjunto de árboles taxonómicos, cada uno especializado en un nivel determinado. Finalmente, cada valor de cada atributo del conjunto Θ, es sustituido por el ancestro que pertenezca al nivel especificado de su árbol taxonómico correspondiente.

Two-Phase Top-Down Specialization

En el artículo *A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud* [12] se presenta una implementación distribuida del algoritmo descripto en la sección anterior, utilizando un enfoque *MapReduce* (definido en la siguiente sección). Se parte de la base de que el ambiente distribuido para la ejecución del algoritmo cuenta con nodos y que cada uno de ellos posee una porción del conjunto Θ.
La implementación consta de dos fases. En la primera se ejecuta el algoritmo *Top-Down Specialization* basado en *MapReduce* en cada uno de los nodos. El resultado de esta fase es un `AL` por cada nodo. La segunda fase toma como entrada la salida de la fase anterior, combina los AL para formar un único `AL` final y luego realiza una última ejecución del algoritmo *Top-Down Specialization* basado en el enfoque *MapReduce*, pero esta vez, utilizando el conjunto completo de datos. Esta segunda ejecución se realiza con el objetivo de especializar aún más, en caso de ser posible, el AL final considerando todo el conjunto de datos.

Con el fin de brindar una descripción más detallada, consideremos las siguientes notaciones:

- Θ**:** Conjunto de datos original. Θ_i refiere a la partición *i-esima* del conjunto, donde $1 \leq i \leq p$ y p es el número total de particiones.
- k, k^I: Parámetros de anonimización que cumplen la relación $k \leq k^I$.
- **AL:** Conjunto de cortes de los árboles taxonómicos.
  - AL^0: `AL` inicial.
  - AL'_i: `AL` intermedio correspondiente a la partición i.
  - AL^*: `AL` final.
- **MRTDS:** Implementación del algoritmo *Top-Down Specialization* basado en el enfoque *MapReduce*.

En el Algoritmo 2.3.1 se brinda una representación de la ejecución del algoritmo.
El algoritmo comienza su ejecución particionando el conjunto Θ en p partes, donde cada registro del conjunto es asignado a una de las partes en forma aleatoria. Luego, cada partición contiene un subconjunto del conjunto de datos

Algoritmo 2.3.1: Pseudo código del algoritmo TPTDS

Entrada: Θ, k, k^I, p
Salida: Conjunto Δ que satisface la condición de *k-anonymity*

1 Particionar Θ en p particiones Θ_i, $1 \leq i \leq p$
2 En cada partición, ejecutar $MRTDS(\Theta_i, k^I, AL^0) \rightarrow AL'_i$
3 $combinar(AL'_1, AL'_2, ..., AL'_p) \rightarrow AL^I$
4 Ejecutar $MRTDS(\Theta, k, AL^I) \rightarrow AL^*$
5 Especializar el conjunto de datos Θ respecto a AL^* para generar el conjunto Δ

original (Θ_i) y el algoritmo MRTDS es aplicado en cada partición. Como resultado, se retorna un nuevo nivel de anonimización denominado AL'_i. Una vez que se tiene un nivel intermedio para cada partición, los mismos son combinados para conseguir un nivel común que englobe a todos los niveles intermedios. Para esto, se toman de los niveles intermedios AL'_i, los árboles taxonómicos de todos los atributos. El nivel de anonimización común se obtiene tomando el ancestro común más general de cada árbol taxonómico de cada atributo (este proceso se representa con la función *combinar* en el pseudocódigo 2.3.1).

En la segunda fase del algoritmo se aplica MRTDS al conjunto Θ de datos partiendo del nivel AL'_i con el fin de encontrar el nivel de anonimización final denominado AL^*. Esta fase tiene como objetivo especializar el conjunto de datos completo si, luego de la combinación de los niveles intermedios, algún corte del AL^I quedó en un nivel más general y aún quedan especializaciones válidas por aplicar. Por último, se reemplazan los valores de los atributos del `QI` de los registros del conjunto de datos original por los definidos en AL^*.

Si bien existen otras implementaciones para el algoritmo TDS distribuido, como por ejemplo los descriptos [15] y [16], las mismas hacen uso de una estructura de datos denominada *Taxonomy Indexed PartitionS* (TIPS) [17]. Esta estructura se encarga de gestionar un conjunto de índices que tienen como objetivo mejorar la eficiencia del cómputo de los algoritmos mediante la indexación de registros. El uso de TIPS permite mejorar el proceso de especificación dado que, al mantener los registros indexados, le basta con recorrer el conjunto de datos completo con menor frecuencia y evita los cómputos recurrentes del calculo iterativo. El problema con el uso de TIPS cuando se tiene una gran cantidad de registros (como es el escenario de Ceibal) es que el con-

sumo de memoria para almacenar la información de los índices es muy grande, reduciendo la cantidad de memoria que queda disponible para la ejecución del algoritmo de anonimización.

Los artículos anteriores fueron seleccionados por el hecho de presentar distintas implementaciones que relacionan los conceptos de *k-anonymity*, *Top-Down Specialization*, *MapReduce* y *Apache Spark*, pero también existen otras familias de técnicas e implementaciones que, si bien escapan al alcance del proyecto, permiten alcanzar la anonimización de un conjunto de datos como por ejemplo técnicas *Bottom-Up Generalization* [18], de agrupación o *clustering* [19] y de perturbación de datos entre otras [20].

En el siguiente capítulo se definirán aquellos elementos pertenecientes al denominado Ecosistema *Hadoop* que serán utilizados en la implementación.

Capítulo 3

Marco tecnológico

En los capítulos anteriores se enmarcó el problema de la anonimización de datos personales desde el punto de vista teórico, ahora es el momento de dar un marco tecnológico al mismo y mencionar las características más importantes de los componentes donde se ejecutará la implementación.

Como se mencionó en el Capítulo 1 el entorno distribuido que posee Plan Ceibal se basa en *Hortonworks Data Platform* y *Hadoop*. Es por este motivo que el contenido del presente capítulo se basa fuertemente en el libro *Hadoop: The Definitive Guide* [2]. Previo a definir los componentes de *Hadoop* que se utilizarán, se proporciona un breve contexto de la realidad y de cómo los entornos distribuidos han tomado tanto protagonismo en los últimos años, tornándose indispensables en la actualidad.

El interés por mantener la privacidad de los datos personales se ha incrementado en los últimos tiempos debido a múltiples factores. Entre estos factores se encuentra la aplicación de métodos de Aprendizaje Automático a la resolución de problemas automatizables, los cuáles requieren una gran cantidad de datos. Otro de los factores relevantes refiere a la gran cantidad de contextos de *Big Data* que existen en la actualidad. El término *Big Data* surge hace 15 años y refiere a contextos que presentan las siguientes características conocidas como las *4 Vs del Big Data* [21]:

1. **Volumen:** Si bien no existe un mínimo volumen de datos a partir del cuál un contexto es denominado como de *Big Data*, solo son considerados los que poseen un volumen de datos que se encuentra en el orden de los

Terabytes o Petabytes.

2. **Velocidad de cambio:** Refiere a la frecuencia en la cuál los datos existentes cambian y los nuevos datos necesitan ser procesados. Como caso de ejemplo podríamos considerar la frecuencia en la cuál los usuarios de la plataforma *Instagram*[1] editan y crean nuevas publicaciones.

3. **Variedad:** Refiere a todos los posibles formatos y fuentes de origen de datos. Entre los posibles formatos se encuentran todas las representaciones de texto (plano, estructurado, etiquetado), fotografías, grabaciones de voz, información extraída a través de sensores y muchos otros.

4. **Veracidad:** Refiere a que no es posible saber con exactitud si los datos obtenidos son veraces, ya que realizando análisis sobre los mismos, se pueden encontrar inconsistencias, inferir discrepancias o contradicciones.

Ya que almacenar los distintos tipos de datos en un contexto de *Big Data* utilizando un solo equipo se torna absolutamente inviable, se utilizan entornos de almacenamiento distribuido conocidos como **clusters de computadoras**. Un cluster de computadoras refiere a un conjunto de computadoras típicamente conectadas mediante una red de área local (LAN), que conforman un servidor virtual de gran capacidad donde cada una de las computadoras se denomina **nodo**. Lo usual es que todos los nodos de un mismo *cluster* presenten las mismas características y sistema operativo.

La paralelización del procesamiento distribuido que se genera dentro del *cluster* es orquestada de forma programática en una capa que se encuentra por encima de los nodos y que permite al usuario manipular el *cluster* como si fuera una sola computadora que posee un gran poder de cómputo y almacenamiento.
Los *clusters* son normalmente utilizados para mejorar el rendimiento y aumentar la tolerancia a posibles fallas de los sistemas, debido a que es posible utilizar el poder de cómputo y almacenamiento de todos los nodos en conjunto.
Si bien existen distintos tipos según la función para la cuál será utilizado el *cluster*, nos concertaremos solamente en aquellos que contengan el marco de trabajo definido por *Hadoop*.

[1] https://www.instagram.com

Apache Hadoop [22] es un conjunto de tecnologías de código abierto implementado por *Apache Software Foundation* [23] que permite realizar el procesamiento de grandes cantidades de datos en *clusters* de computadoras de manera sencilla, eficiente, confiable y escalable, donde cada computadora ofrece cómputo local y almacenamiento. Fue diseñado para brindar alta disponibilidad ya que es capaz de detectar y manejar fallas ocurridas en la Capa de Aplicación y no depender enteramente del *hardware* para lograrlo.

Gran parte de la eficiencia de *Hadoop* radica en que reparte los datos entre los nodos del *cluster* donde realizará el procesamiento. Esta característica se denomina *data locality* y provoca que el acceso a los datos se realice de forma local en cada nodo, incrementando la velocidad de acceso sin consumir parte del ancho de banda de la red que conecta los distintos nodos. Para lograr esta eficiencia, se necesita que los datos estén almacenados en un sistema de archivos distribuidos. Cuando se trabaja con *Hadoop*, este sistema de archivos distribuido es *Hadoop Distributed File System* (HDFS).

Asimismo, *Hadoop* se encarga de la planificación de tareas y de procesos necesarios para la coordinación de los nodos del *cluster* además del manejo de errores en caso de que estos ocurran. Para lograr la ejecución de los procesos de forma paralela dentro del *cluster*, *Hadoop* define que cada unidad de procesamiento que el usuario desea ejecutar es un **trabajo** y que, a su vez, se dividen en **tareas** que pueden ser de dos tipos: **tareas de *map*** o **tareas de *reduce***. Estas tareas son programadas para su ejecución en los distintos nodos y, en caso de que una tarea falle en un nodo, se vuelve a programar de forma automática para ser ejecutada en un nodo diferente.

El denominado *Ecosistema Hadoop* está conformado por distintos componentes y proyectos, algunos de los cuáles se muestran en la Figura 3.1. A continuación se presentan los aspectos más relevantes de los componentes utilizados en este proyecto.

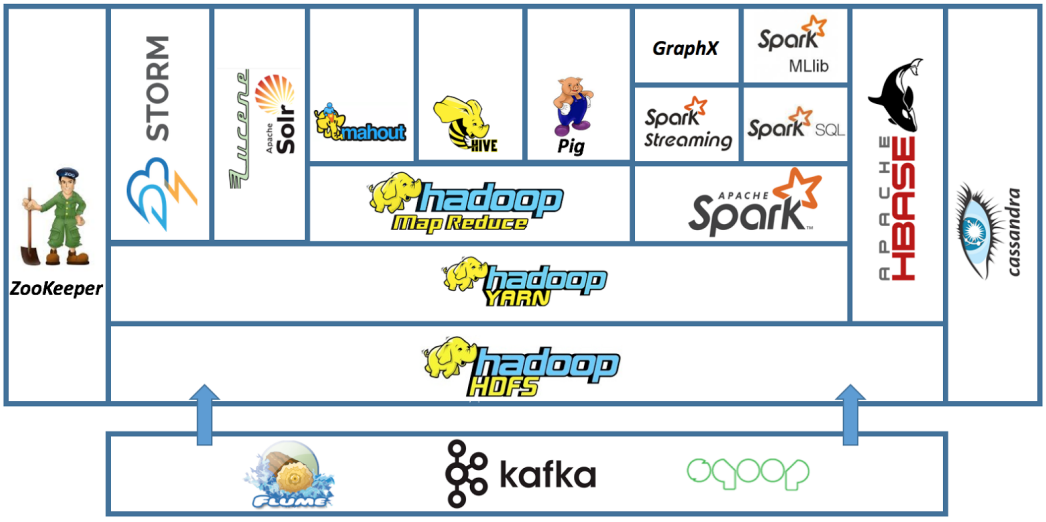


Figura 3.1: Ecosistema Hadoop [2].

3.1. HDFS

HDFS es el sistema de archivos distribuido que presenta *Hadoop*. Si bien es el recomendado cuando se trabaja dentro de este marco de trabajo (dadas las optimizaciones que presenta), también se tiene la posibilidad de utilizar otros sistemas de archivos distribuidos gracias a que *Hadoop* cuenta con una abstracción del sistema de archivos de propósito general, lo que permite la integración con otros sistemas como sistemas locales o *Amazon S3*[3].

HDFS posee múltiples características entre las que se destacan dos, su **gran capacidad de almacenamiento** siendo capaz de almacenar archivos del orden de Gigabytes, Terabytes y hasta Petabytes y la implementación de un enfoque de **una escritura, muchas lecturas**. HDFS está construido utilizando el patrón "escribir solo una vez y leer muchas veces", lo que significa que un conjunto de datos se copia desde su origen una sola vez y luego se utiliza para realizar varios análisis sobre los datos. Posiblemente estos análisis requieran utilizar el conjunto de datos completo, por lo que se torna más importante el tiempo que lleva leer el conjunto completo que simplemente la latencia que existe para leer el primer registro.

[2]http://blog.newtechways.com/2017/10/apache-hadoop-ecosystem.html
[3]https://aws.amazon.com/s3

Como en todo sistema de archivos, los archivos almacenados son particionados (en caso de que el tamaño del archivo supere el tamaño de un bloque) y las particiones son almacenadas en **bloques**. Un bloque HDFS refiere a la mínima cantidad de datos que pueden ser leídos o escritos. El tamaño por defecto del bloque en HDFS es de `128 MB` y presenta como gran diferencia con un sistema de archivos centralizado que, si un archivo es más pequeño que el tamaño de un bloque, el mismo no ocupa el bloque completo. Por ejemplo, si se desea almacenar un archivo de tamaño `1 MB`, este solo ocupará `1 MB` dentro de un bloque de `128 MB` y el resto del bloque (`127 MB`) podrá ser utilizado para almacenar otro archivo. El principal beneficio de tener un sistema de archivos distribuido es que se pueden almacenar archivos que sean más grandes que un disco de la red. Además, no es necesario que los bloques pertenecientes a un archivo se almacenen en el mismo disco del *cluster*, teniendo la posibilidad de utilizar cualquier disco de la red.

En un *cluster* HDFS se presentan dos tipos de nodos, el nodo maestro y los nodos esclavos. El nodo maestro es el encargado de gestionar el espacio de nombres del sistema de archivos: mantiene el árbol que representa al sistema y los metadatos de todos los archivos y directorios de dicho árbol. También conoce los nodos esclavos donde se encuentran los bloques de cada archivo, información que obtiene de los nodos esclavos cuando se inicializa el sistema. Por otro lado, los nodos esclavos son los encargados de almacenar y retornar los bloques del HDFS cuando el nodo maestro, u otro cliente, lo solicita. Además, mantienen actualizado al nodo maestro respecto a los bloques que almacena de forma periódica.

En la siguiente sección se presenta YARN, un gestor de recursos que opera sobre HDFS (ver Figura 3.1).

3.2. YARN

Apache *Yet Another Resource Negotiator*[4] (YARN) es el manejador de recursos que posee *Hadoop*. Para realizar la gestión de los recursos, YARN provee una API para solicitar y trabajar con los mismos.

[4]https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Los servicios principales de YARN son brindados a través de dos gestores, uno de recursos y uno de nodos. El gestor de recursos tiene la tarea de gestionar el uso de los recursos del *cluster* y existe solamente un gestor de este tipo. Por otro lado, los gestores de nodos se encargan de inicializar y controlar los contenedores que se están ejecutando en los nodos (un contenedor ejecuta un proceso específico de la aplicación con una cantidad de recursos limitada).

Para ejecutar una aplicación utilizando YARN, un cliente debe primero invocar al gestor de recursos y solicitar que este ejecute un proceso de tipo `aplicación maestra`. Luego, el gestor de recursos solicita a uno de los gestores de nodos que inicialice el proceso en un contenedor. Una vez que el proceso se ejecutó y se obtiene el resultado, depende de la `aplicación maestra` si retorna el resultado al cliente o si solicita más contenedores al gestor de nodos para ejecutar un cómputo distribuido. Este último comportamiento es el que presenta la aplicación *MapReduce* ejecutada sobre YARN, la cuál se describirá a continuación.

3.3. MapReduce

Si bien el entorno distribuido del Plan Ceibal utiliza *Spark* (detallado en la Sección 3.4), es importante comprender como funciona el paradigma *MapReduce* debido a que su funcionamiento es muy similar al de *Spark*, por lo que el esta sección se centra en explicar *MapReduce*.

Hadoop presenta su propia implementación del modelo de programación *MapReduce*, concepto presentado por Google en el año 2004 [24], que permite que la escalabilidad del procesamiento se produzca de forma lineal a medida que se incrementa el volumen de datos.
El concepto principal detrás del modelo consiste en que los datos serán sometidos a dos fases de procesamiento: la fase de *Map* y la fase de *Reduce*. Se pueden presentar varias fases de *Map* y *Reduce* dentro de una misma implementación.

La **fase de *Map*** es la primera parte del procesamiento, representada en color amarillo en la Figura 3.2. El usuario debe definir una función que toma los datos de entrada y genera como resultado un conjunto de pares clave/valor

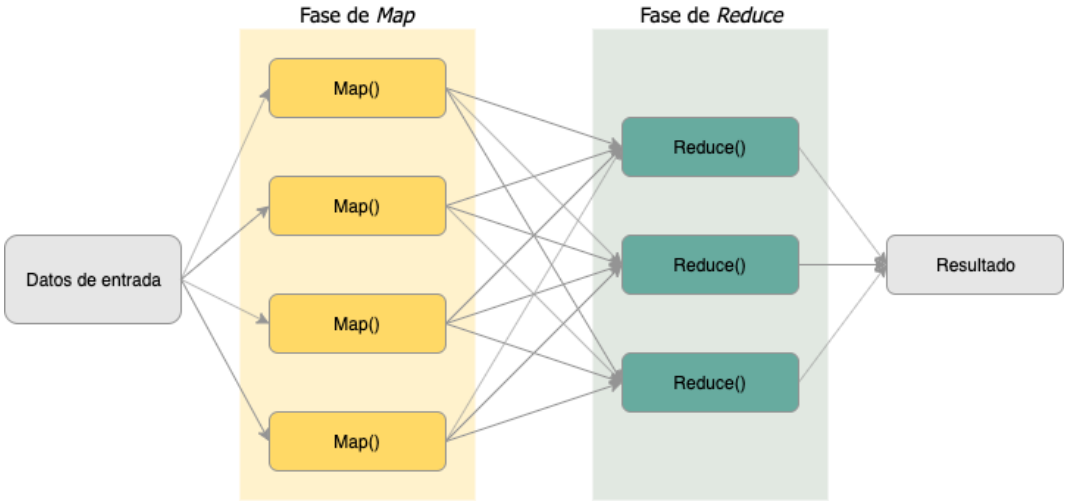


Figura 3.2: Ilustración de un trabajo *MapReduce*

las cuáles servirán como entrada de la fase de *Reduce*. Luego de finalizada la fase de *Map*, se ejecuta la **fase de *Reduce*** (representada en color verde en la Figura 3.2), la cuál se encarga de la combinación de todos los pares clave/valor (mediante la definición de una función la cuál toma como entrada el resultado de la aplicación de la fase de *Map*) los cuáles contengan la misma clave para generar el resultado final del procesamiento. Aquellos programas que son implementados siguiendo este modelo son paralelizados automáticamente. Para lograr este paralelismo, *Hadoop* particiona el conjunto de datos original y, utilizando primitivas del modelo *MapReduce*, procesa cada partición por separado de forma paralela.

A modo de ejemplo, se considera el problema de conteo de palabras, en el cuál, dado un texto de entrada se debe contar la cantidad de ocurrencias de cada palabra. Para atacar un problema utilizando el enfoque *MapReduce* se debe definir la función que se aplicará en la fase de *Map* y la función que se aplicará en la fase de *Reduce*. En este caso, utilizaremos las funciones que se definen en [24], mostradas en las Figura 3.3.
Aplicaremos estas funciones sobre un texto inventado que contiene tipos de animales. La explicación paso a paso de cómo se aplican las funciones de la Figura 3.3 en el conjunto de datos se ilustran en la Figura 3.4.

Un **trabajo *MapReduce*** está conformado por: los datos de entrada, el programa *MapReduce* y la información de la configuración. Para gestionar este

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Figura 3.3: Funciones de *Map* y *Reduce*

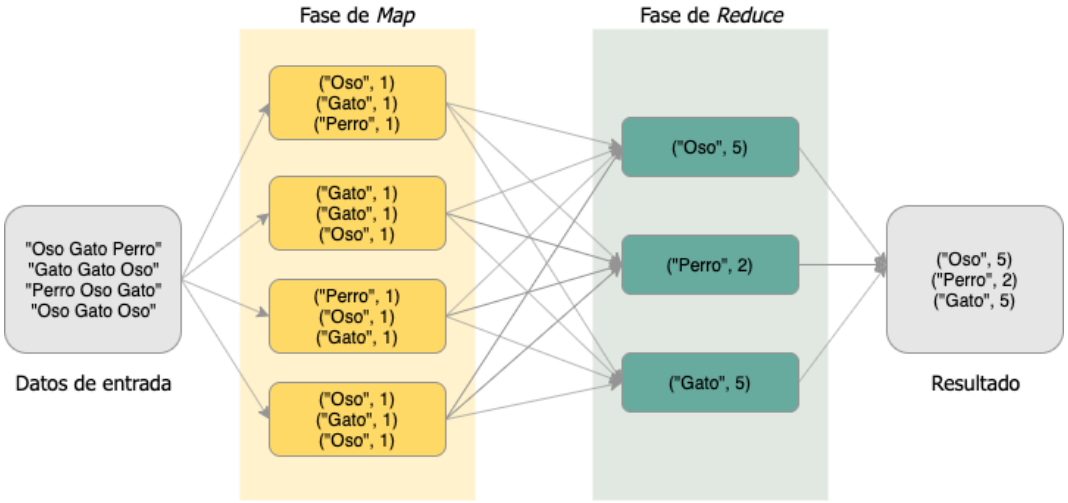


Figura 3.4: Ejemplo *MapReduce* paso a paso

tipo de trabajo, *Hadoop* divide los datos de entrada en particiones de tamaño fijo (comúnmente se toma como tamaño `128 MB` ya que es el valor por defecto que posee un bloque HDFS) y crea una tarea de *map* para cada partición en donde se ejecuta la función de *map* definida por el usuario para cada registro de la partición. *Hadoop* siempre intentará ejecutar la tarea de *map* en el mismo nodo donde se encuentran los datos, es por este motivo que se recomienda que el tamaño de cada partición no supere los `1128 MB`, ya que esta es la cantidad máxima de datos que se pueden almacenar en un solo nodo (un bloque de HDFS).

El proceso de ejecución de un trabajo *MapReduce* dentro de un *cluster* involucra a cinco entidades independientes:

1. **Cliente:** Es quien solicita la ejecución del trabajo *MapReduce*.

2. **Gestor de Recursos YARN:** Coordina los recursos necesarios para la ejecución del proceso.

3. **Gestores de Nodos YARN:** Inicializan y controlan los contenedores en el *cluster*.

4. **Aplicación MapReduce:** Coordina las tareas *MapReduce* necesarias para completar el trabajo. Cabe mencionar que tanto la aplicación cómo las tareas *MapReduce*, son ejecutadas en contenedores que son programados por el Gestor de Recursos y son gestionados por los Gestores de Nodos.

5. **Sistema de archivos distribuido:** Utilizado para compartir los archivos que involucran los trabajos entre las otras entidades mencionadas. Comúnmente se utiliza HDFS.

A continuación se explica el funcionamiento de Spark y se destacan los puntos en común que posee con *MapReduce*.

3.4. Spark

Apache Spark es un marco de trabajo de cómputo para *clusters* utilizado en procesamiento de datos a gran escala. Si bien no utiliza *MapReduce* como su motor de ejecución, posee múltiples características en común como se destacarán a lo largo de la sección. Spark puede ser ejecutado utilizando YARN y utilizar HDFS como almacenamiento, y además, cuenta con interfaces para distintos lenguajes como Scala, Java y Python.

Una de las características más destacables de Spark es su capacidad de mantener en memoria una gran cantidad de datos entre la ejecución de los trabajos, lo que le permite superar el rendimiento de un trabajo equivalente en *MapReduce*, ya que los datos no necesitan ser cargados desde los discos.
Además, posee ciertas abstracciones que utiliza durante su ejecución. La más destacada es el concepto de *Resilient Distributed Dataset* (RDD). Un **Resilient Distributed Dataset** es una colección de objetos de sólo lectura la cuál es particionada en múltiples máquinas de un *cluster*. El término *resilient* refiere a que, si por algún motivo una partición se pierde, Spark es capaz de reconstruirla automáticamente a partir del RDD por el cuál fue generada.
Los RDDs pueden ser creados mediante tres métodos: a partir de una colección

de objetos que se encuentra en memoria, a partir de un conjunto de datos que se encuentra en un almacenamiento externo o a partir de otro RDD.

En una típica ejecución de un programa de Spark, uno o más RDDs son cargados con información y, a través de la aplicación de un conjunto de operaciones, son convertidos en otro conjunto de RDDs conteniendo el resultado deseado. Spark ofrece dos tipos de operaciones que pueden ser aplicadas sobre un RDD: las **transformaciones** y las **acciones**.

Las operaciones del tipo **transformación** generan un RDD a partir de otro ya existente. Además, no se ejecutan de forma inmediata, sino que para que efectivamente se ejecute la transformación, se debe ejecutar una **acción** sobre el RDD, es por esto que se denominan *perezosas*. Como ejemplos de transformaciones se tienen `map, filter, flatMap, mapPartitions, groupByKey, reduceByKey`. En contraparte, una operación del tipo **acción** ejecuta un proceso de cómputo sobre un RDD y luego, retorna el resultado al usuario o lo guarda en un almacenamiento externo. Al contrario que las **transformaciones**, las acciones son ejecutadas de forma inmediata. Como ejemplos de acciones se presentan `collect, count, first, take, countByKey`.

Entre los paralelismos que posee Spark con *MapReduce*, se encuentra el concepto de **trabajo Spark**. Este tipo de trabajos son más generales que los trabajos *MapReduce* debido a que son generados a partir de un grafo acíclico dirigido y arbitrario de **etapas**, donde cada etapa es, a grandes rasgos, equivalente a una fase de *map* o una fase *reduce* en *MapReduce*. A su vez, las **etapas** se dividen en **tareas** que son ejecutadas en paralelo en las distintas particiones de un RDD distribuidas en el *cluster*. La cantidad de particiones que presenta un RDD quedan definidas por el usuario (al momento de crearlo) o automáticamente por Spark. Por ejemplo, si se crea un RDD a partir de un archivo HDFS, Spark generará una partición por cada bloque HDFS.

Cada trabajo Spark es ejecutado en el contexto de una **aplicación** representada por la clase *SparkContext* que permite agrupar RDDs y variables compartidas. Esta aplicación posee la capacidad de ejecutar más de un trabajo Spark, de forma serial o paralela, y provee un mecanismo a través del cuál un trabajo puede acceder a un RDD el cuál se encuentra en la memoria *caché* cómo re-

sultado de un trabajo anterior ejecutado en la misma aplicación. El nivel de paralelización que puede ser alcanzado por un trabajo es determinado por una propiedad de Spark que depende dónde se ejecute el mismo; si se ejecuta de forma local, la propiedad toma como valor la cantidad de núcleos de la máquina, mientras que si se ejecuta en un *cluster*, se considera el número total de núcleos de todos los nodos esclavos del *cluster*.

En la ejecución un trabajo Spark participan, a muy alto nivel, dos entidades: el *driver* y los *ejecutores*. El *driver* contiene la aplicación (la definición de la instancia del *SparkContext*) y programa las tareas para un trabajo, mientras que los *ejecutores* son exclusivos para la aplicación: se ejecutan exclusivamente por el tiempo que ésta dure y ejecutan sus tareas. Usualmente el *driver* es ejecutado como un cliente no gestionado por el manejador del *cluster* y los *ejecutores* se ejecutan en las máquinas del mismo.
Cada vez que se ejecuta una acción sobre un RDD, se crea un trabajo, se crean las tareas pertinentes y se ejecutan en cada partición del RDD. Luego, el resultado de cada partición es devuelto al *driver*, el cuál combina los resultados de todas las particiones para generar el resultado final.

En este capítulo se brindó un breve resumen de cada componente de *Hadoop* relevante en la ejecución de un trabajo Spark. En los capítulos subsiguientes se especificará la implementación realizada y los resultados obtenidos.

Capítulo 4

Solución implementada

La solución implementada genera un conjunto de datos que satisface la condición de *k-anonymity*, adaptándose a las condiciones tecnológicas planteadas en el Capítulo 3. La misma consta de una biblioteca desarrollada en el lenguaje Python versión 3.7, que utiliza la API de *Apache Spark* para este lenguaje denominada *PySpark*[1] y toma como referencia el algoritmo TPTDS [12], el cuál se adaptó a la necesidades del problema en cuestión.

La biblioteca consta de tres módulos. Un módulo de preprosesamiento el cuál, entre otras cosas, es el encargado de realizar validaciones sobre los datos de entrada provistos por el usuario. Por ejemplo, en él se verifica si el parámetro de anonimización k es válido, se elimina información redundante del conjunto de datos y, en caso de que el mismo presente valores faltantes en sus registros, se lo completa. Además, se crean las estructuras necesarias para la ejecución del algoritmo. Un segundo módulo, llamado de anonimización, es el encargado de ejecutar el algoritmo de anonimización, tomando como parámetro de entrada el resultado de la etapa anterior. Por último, se tiene un módulo de validación que es utilizado por los dos módulos anteriores. Éste posee la implementación de las distintas validaciones que se ejecutan sobre datos provisto por el usuario y sobre el resultado del proceso de anonimización. Para ejecutar una técnica de anonimización sobre un conjunto de datos utilizando la biblioteca implementada se debe preprocesar el conjunto de datos inicial Θ para luego aplicar el algoritmo de anonimización. La Figura 4.1 presenta el diagrama de clases de la biblioteca y a continuación se detalla cada módulo.

[1] https://spark.apache.org/docs/latest/api/python/index.html

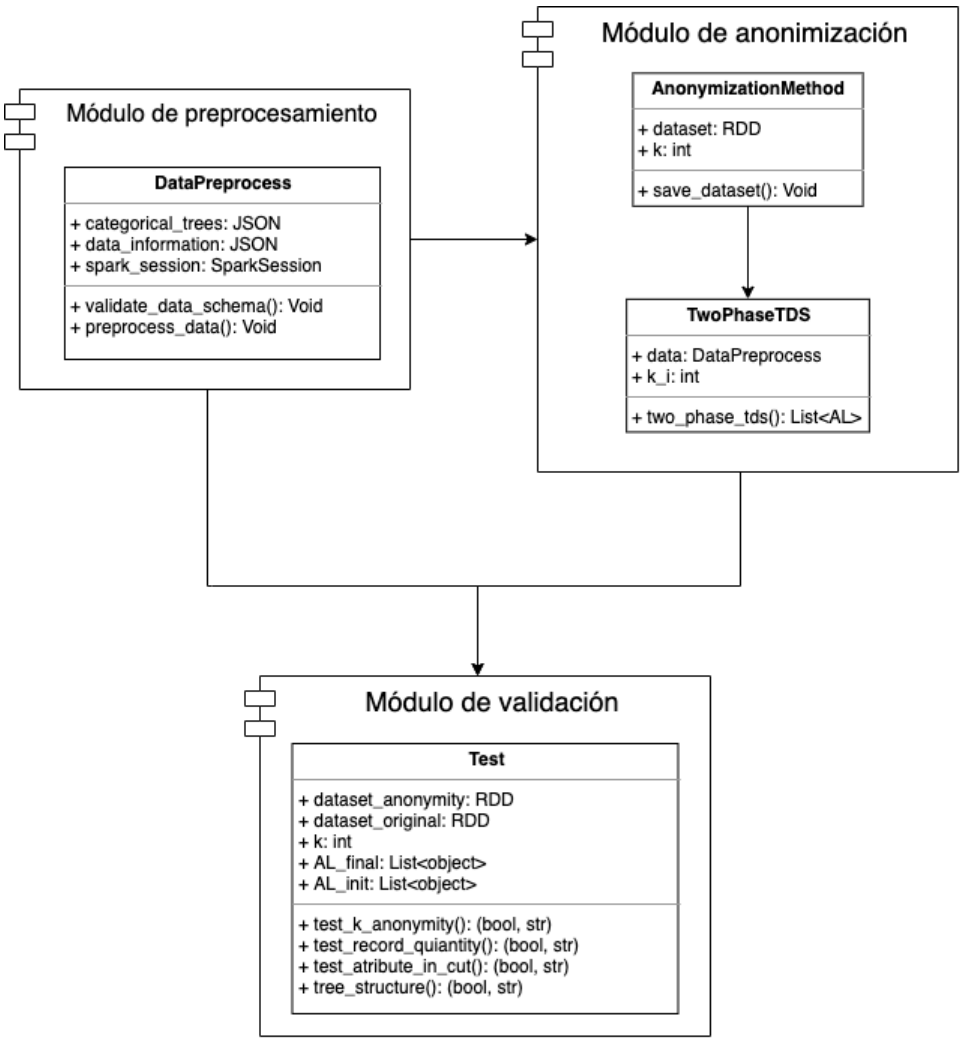


Figura 4.1: Diagrama de clases de la biblioteca Python implementada.

4.1. Módulo de preprocesamiento

La etapa de preprocesamiento de datos es la primera parte del proceso de anonimización y se implementa en la clase `DataPreprocess`. La necesidad de preprocesar el conjunto de datos Θ radica en que comúnmente este tipo de conjunto contiene información redundante (más de un atributo aporta la misma información en diferentes formatos), valores faltantes en algunos de sus registros y/o información errónea o ruido. Es con esta motivación, que esta etapa tiene tres objetivos: validar los datos de entrada, crear las estructuras necesarias para la ejecución del algoritmo de anonimización y dar el formato necesario al conjunto de datos. A continuación se describe cada uno de ellos.

4.1.1. Validación de datos de entrada

La información que debe proveer el usuario al instanciar la clase `DataPreprocess` se reduce a dos parámetros de entrada obligatorios y un parámetro opcional. El primer parámetro obligatorio (`data_information`) es un objeto JSON que contiene la ruta donde se encuentra el conjunto de datos a anonimizar Θ dentro del HDFS y especifica las características del mismo. Dentro de las características se incluyen: el tipo de cada atributo (numérico o categórico), se indican los atributos redundantes que se deben remover del conjunto y cuáles son los atributos confidenciales. Además, se especifica el formato en el cuál se encuentran los archivos del conjunto Θ dentro de las dos opciones disponibles, JSON y CVS, se define la cantidad de particiones que serán aplicadas y se indica si el resultado del proceso de anonimización debe ser almacenado o no dentro del entorno distribuido. Un posible ejemplo de entrada se define en el objeto 4.1.

```
{
    "attributes_to_remove": [
        "fnlwgt", "education-num", "capital-gain",
        "capital-loss"
    ],
    "sensitive_attributes": ["class"],
    "categorical_attributes": [
        "workclass", "education", "marital-status",
        "occupation", "relationship", "race",
        "sex", "native-country"
    ],
    "continous_attributes": ["age", "hours-per-week"],
    "dataset_missing_values": true,
    "save_anonymization_result": true,
    "result_file_name": "tds_anonymization",
    "partitions": 3,
    "file_path": "../data/adult/adult.json"
}
```

Listing 4.1: Ejemplo de posible valor para el atributo `data_information`.

Para validar el objeto JSON que representa el parámetro de entrada `data_information` se utiliza un JSON Schema[2] y las claves que deben estar presentes

[2]https://json-schema.org

en el objeto son las definidas a continuación.

- `attributes_to_remove`: Lista de nombres de aquellos atributos que deben ser removidos del conjunto de datos.
- `sensitive_attributes`: Lista de nombres de los atributos que conforman el atributo confidencial. La información de los atributos se sintetiza en una tupla donde cada valor de la misma refiere al valor de cada atributo de la lista. Luego, esta tupla es considerada el atributo confidencial.
- `categorical_attributes`: Lista de nombres de los atributos categóricos.
- `continous_attributes`: Lista de nombres de los atributos numéricos.
- `dataset_missing_values`: Booleano que indica si el conjunto de datos presenta valores faltantes en sus registros. En caso se ser afirmativo, los valores faltantes serán completados utilizando aquel valor que contenga más ocurrencias en el conjunto de datos para el atributo correspondiente.
- `save_anonymization_result`: Booleano que indica si el resultado de la anonimización debe ser almacenado. De lo contrario el resultado se retornará al *driver*.
- `result_file_name`: Nombre del archivo que contendrá el resultado de la anonimización. Al nombre especificado se le agregará el parámetro k utilizado en el método de anonimización junto con la fecha de la ejecución.
- `partitions`: Cantidad de particiones en las que se dividirá el conjunto de datos.
- `file_path`: Ruta donde se encuentra el conjunto de datos de entrada.

Si la validación del objeto provisto por el usuario contra el JSON Schema definido falla, se emite una excepción indicando que el objeto no respeta el esquema esperado.

El segundo parámetro obligatorio que requiere la clase `DataPreprocess` (`categorical_trees`) es un objeto JSON que describe el árbol de taxonomía de cada atributo de tipo categórico. Cada nodo de un árbol es representado por un objeto el cuál debe contener dos claves: `value` y `children`. La clave `value` tiene asociado un *String* que representa la etiqueta del nodo que lo identifica y `children` corresponde a una lista con sus nodos hijos. En caso de que el nodo sea una hoja, no se debe especificar el valor de la clave `children`. A continuación se provee un JSON de ejemplo que define el árbol taxonómico

del atributo `marital-status` presentado en la Figura 4.2.

```
{
    ...,
    "marital-status": {
        "value": "All marital status",
        "children": [
            {
                "value": "Single",
                "children": [
                    {"value": "Married-spouse-absent"},
                    {"value": "Divorced"},
                    {"value": "Separated"},
                    {"value": "Widowed"},
                    {"value": "Never-married"}
                ]
            },
            {
                "value": "Couple",
                "children": [
                    {"value": "Married-civ-spouse"},
                    {"value": "Married-AF-spouse"}
                ]
            }
        ]
    }
    ...,
}
```

Listing 4.2: Ejemplo de posible valor para el atributo `marital-status` dentro del objeto `categorical_trees`.

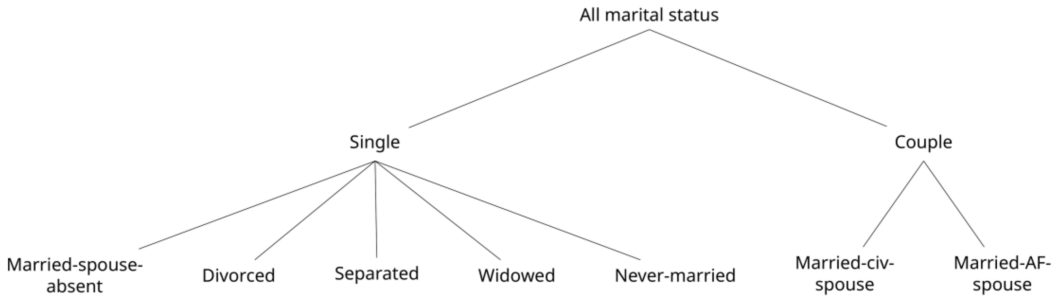


Figura 4.2: Ejemplo de árbol taxonómico para el atributo `marital-status`

Por último, la clase `DataPreprocess` presenta un parámetro opcional (`spark`) que permite pasar una sesión de Spark creada previamente. En caso de que este parámetro no sea especificado, la sesión de Spark necesaria para la ejecución del algoritmo de anonimización será creada al comienzo de la etapa de preprocesamiento. La posibilidad de usar una sesión creada previamente se justifica dado que, cuando la biblioteca se utiliza en un entorno distribuido HDP este inicializa una sesión de Spark, por lo que no es necesario generar una nueva.

4.1.2. Creación de estructuras

Luego que el formato de los parámetros de entrada fue validado, se crean las estructuras necesarias para la ejecución del algoritmo de anonimización. Como se mencionó en el Capítulo 2, los enfoques TDS hacen uso de determinadas estructuras de datos que posibilitan su funcionamiento, algunas de las cuáles serán provistas por el usuario como parámetro de entrada y otras que serán generadas de forma automática por la implementación. En esta etapa se construyen tres estructuras fundamentales: el RDD que contiene conjunto de datos, los árboles de taxonomía y el nivel de anonimización inicial (AL_0).

En primer lugar se crea un RDD con el conjunto de datos Θ, contiendo la cantidad de particiones definidas por el usuario mediante la clave `partitions` o, en caso de no ser especificada, la cantidad de particiones es establecida por Spark. Una vez creado el RDD, se crean los árboles taxonómicos. Los árboles de los atributos categóricos se construyen a partir de los datos provistos por el usuario en el parámetro de entrada `categorical_trees`, mientras que los de los atributos numéricos se crean automáticamente a partir del conjunto Θ. Se implementó un algoritmo capaz de construirlos a partir de los datos presentes en este. Para cada atributo numérico, el algoritmo construye un árbol taxonómico que cumple las siguientes condiciones.

- El árbol es balanceado
- Cada nodo no hoja es un intervalo de valores del atributo numéricos
- Cada nodo hoja representa un valor dentro del intervalo de su nodo padre
- Cada nodo no hoja tiene dos hijos, con la excepción de los nodos que tiene hojas como hijos, los cuales pueden tener más de dos hijos para mantener balanceado el árbol

El algoritmo parte de un intervalo, $[x_1, x_N]$ que corresponde al nodo raíz del árbol y contiene todos los valores del atributo, siendo x_1 el menor y x_N el mayor de ellos. Para generar los hijos de este primer nodo, se busca un valor x_i perteneciente al intervalo padre de forma de crear dos intervalos, $[x_1, x_i]$ y $[x_{i+1}, x_N]$. Este proceso de dividir los intervalos en otros más pequeños se repite de forma iterativa, hasta alcanzar en los valores que corresponden a las hojas del árbol. En el Algoritmo 4.1.1 se proporciona un pseudo código de la implementación.

Algoritmo 4.1.1: Pseudo código del algoritmo de creación de árboles taxonómicos de los atributos numéricos.

Entrada: RDD Θ, lista de atributos numéricos *numericos*
Salida: Conjunto TT de árboles taxonómicos de atributos numéricos
para $n \in numericos$ **hacer**
    Crear lista val_n de los valores de n en Θ.
    Obtener $x_1 = min(val\_n)$ y $x_N = max(val\_n)$.
    Crear $nodo\_raiz$ del árbol con intervalo $[x_1, x_N]$
    $TT = nodo\_raiz$
    $padres = [nodo\_raiz]$
    mientras TT balanceado y $val\_n \neq \emptyset$ **hacer**
        $hijos = \emptyset$
        para $p = [p_1, p_2] \in padres$ **hacer**
            Obtener $hijo \in val\_n$ de p que $max(IG)$
            si $hijo < p_1$ y $hijo + 1 < p_2$ **entonces**
                Crear $hijo\_izquierdo = [p_1, hijo]$ y
                $hijo\_derecho = [hijo + 1, p_2]$ del nodo p
                Agregar $hijo\_derecho$ y $hijo\_izquierdo$ a $hijos$
            en otro caso
                Crear nodos hojas $hijo\_izquierdo = p_1$ y
                $hijo\_derecho = p_2$ del nodo p
                Quitar $hijo\_derecho$ y $hijo\_izquierdo$ de val_n
        $padres = hijos$
    si TT no balanceado **entonces**
        para $nodo \in TT$ y $nodo$ desbalancea TT **hacer**
            $p = nodo.padre$
            $nuevo\_padre = p.padre$
            $nodo.padre = nuevo\_padre$

En la implementación, se utiliza la biblioteca *AnyTree*[3] para representar los

[3]https://anytree.readthedocs.io/en/latest/

árboles de taxonomía.

4.1.3. Acondicionamiento del conjunto de datos

La última etapa del preprocesamiento es el acondicionamiento del conjunto Θ. Este acondicionamiento del conjunto refiere a eliminar aquellos atributos definidos en `attributes_to_remove`, generar una tupla con los valores definidos en `sensitive_attributes` con el fin de mantener un solo atributo sensible que los contemple a todos y completar los valores faltantes de los registros si el valor de `dataset_missing_values` así lo indica.

El algoritmo TPTDS contempla solamente un atributo sensible, pero en la realidad del Plan Ceibal, un registro puede presentar más de un atributo sensible. Es por este motivo que se permite especificar una lista de atributos, a partir de los cuáles se genera una tupla que contiene los valores de los atributos definidos en `sensitive_attributes`. Por otro lado, si el usuario indica que se deben completar los valores de los atributos faltantes mediante la clave `dataset_missing_values`, se calcula para cada atributo cuál es el valor con más ocurrencias dentro del conjunto Θ. Luego, cada registro que posea valores faltantes en algún atributo, es completado con el valor con más ocurrencias del mismo. El cálculo de ocurrencias y la sustitución para cada atributo se realiza utilizando de operaciones `map` sobre el RDD que contiene el conjunto Θ. Por ejemplo, consideremos que el valor del parámetro `data_information` es el objeto definido en 4.1 y el registro de ejemplo perteneciente al conjunto de datos Θ definido en 4.3. Además, consideraremos que el valor con más ocurrencias para el atributo `native-country` es `United-States`.

```
{
    "age": 50,
    "workclass": "Self-emp-not-inc",
    "fnlwgt": 83311,
    "education": "Bachelors",
    "education-num": 13,
    "marital-status": "Married-civ-spouse",
    "occupation": "Exec-managerial",
    "relationship": "Husband",
    "race": "White",
    "sex": "Male",
```

```
    "capital-gain": 0,
    "capital-loss": 0,
    "hours-per-week": 13,
    "native-country": ?,
    "class": "<=50K"
  }
```

Listing 4.3: Ejemplo de registro antes de la etapa de preprocesamiento.

Luego de la etapa de preprocesamiento, el registro tiene el formato definido en el objeto 4.4.

```
{
    "age": 50,
    "workclass": "Self-emp-not-inc",
    "education": "Bachelors",
    "marital-status": "Married-civ-spouse",
    "occupation": "Exec-managerial",
    "relationship": "Husband",
    "race": "White",
    "sex": "Male",
    "hours-per-week": 13,
    "native-country": "United-States",
    "class": ("<=50K")
  }
```

Listing 4.4: Ejemplo de registro luego de la etapa de preprocesamiento.

Finalmente, una vez que se tienen todos los árboles taxonómicos, se crea el AL_0, el cuál es un objeto JSON donde las claves son los nombres de los atributos y los valores son las referencias al nodo raíz del árbol de cada atributo.

El resultado de la etapa de preprocesamiento es un objeto, instancia de la clase `DataPreprocess`, que concentra todos los datos y metadatos necesarios para la etapa de anonimización. Entre estos datos se encuentran el RDD que contiene el conjunto de datos transformado con los datos preprocesado y el `AL` inicial.

4.2. Módulo de anonimización

La segunda etapa del proceso de anonimización es la aplicación del algoritmo de anonimización basado en la técnica TPTDS.

Con el objetivo de que la implementación pueda ser fácilmente extendida por posibles trabajos futuros que incorporen otros algoritmos a la biblioteca, se optó por modelar el módulo de anonimización como una generalización de clases, donde la clase abstracta tiene el nombre de `AnonymizationMethod` y engloba los datos transversales a los distintos algoritmos de anonimización. Esta clase toma dos parámetros de entrada, el primero se denomina `dataset` y alude al RDD que contiene el conjunto de datos previamente procesado. El segundo parámetro, `k`, es un número entero que define la condición de *k-anonymity*. La clase concreta de la generalización se denomina `TwoPhaseTDS`, esta hereda las propiedades de `AnonymizationMethod` e implementa el algoritmo asociado a la técnica de TPTDS. Presenta tres parámetros de entrada. El primero se denomina `data` y refiere al objeto, instancia de la clase `DataPreprocess`, resultado de la etapa de preprocesamiento. El segundo, denominado `k`, es el mismo que el de la clase `AnonymizationMethod`. En cuanto al tercer parámetro, denominado k^I, es el que define la condición de *k-anonymity* que será utilizada en la ejecución del algoritmo TDS en cada nodo y debe cumplir con la condición $k \leq k^I$.

Se realizaron algunas adaptaciones sobre la técnica TPTDS para que la misma pueda ser aplicada sobre la realidad de Plan Ceibal. Estas adaptaciones refieren a los dos primeros pasos presentados en el Algoritmo 2.3.1 sobre particionar el conjunto Θ en p particiones y aplicar en cada una de estas particiones el algoritmo MRTDS.
Respecto a particionar el conjunto de datos, a diferencia del Algoritmo 2.3.1, la solución propuesta no necesita realizar una división el conjunto de datos mediante aplicaciones de operaciones `map` y `reduce` debido a que PySpark provee la primitiva `repartition(p)`, la cuál se encarga de particionar el RDD en p particiones. La segunda adaptación realizada refiere a que el algoritmo MRTDS no es ejecutado en cada partición, sino que este paso fue remplazado por una implementación secuencial del algoritmo TDS. Esto se llevó a cabo dado que la primitiva de *Pyspark* `mapPartitions()` (utilizada en el *driver* de

la implementación para aplicar una función a cada partición del RDD) utiliza funciones de Python del tipo Generador[4], no siendo posible realizar operaciones de *map* y *reduce* dentro de la función a ejecutar en cada partición. En el Algoritmo 4.2.1 se proporciona el pseudo código de la versión modificada del Algoritmo 2.3.1.

Algoritmo 4.2.1: Pseudo código del algoritmo TPTDS adaptado

Entrada: RDD Θ, k, k^I, AL^0, p
Salida: Conjunto Δ que satisface la condición de *k-anonymity*
1 En cada partición θ_i del RDD, ejecutar $TDS(\theta_i, k^I, AL^0) \rightarrow AL'_i$
2 $combinar(AL'_1, AL'_2, ..., AL'_p) \rightarrow AL^I$
3 Ejecutar $MRTDS(\Theta, k, AL^I) \rightarrow AL^*$
4 Especializar el conjunto de datos Θ respecto a AL^* para generar el conjunto anonimizado Δ

A continuación se describe cada uno de los pasos del algoritmo adaptado.

4.2.1. Primer paso: procesamiento secuencial

El primer paso del algoritmo de anonimización adaptado es la ejecución del algoritmo TDS sobre cada partición del RDD. El pseudo código del mismo se presenta en el Algoritmo 4.2.2.

Algoritmo 4.2.2: Pseudo código del algoritmo TDS

Entrada: θ_i, k^I, AL^0
Salida: Nivel de anonimización AL'
1 Inicializar los valores de la métrica IGPL para cada especialización $spec \in \cup_{j=1}^{m} Cut_j$
2 **mientras** $\exists spec \in \cup_{j=1}^{m} Cut_j$ válida **hacer**
  Buscar la mejor especialización en AL_i
  Actualizar AL_i a AL_{i+1}
  Actualizar los valores de la métrica IGPL para cada nueva especialización de AL_{i+1}
3 $AL' = AL$

En cada partición del RDD se aplica el Algoritmo 4.2.2, utilizando k^I para definir la condición de *k-anonymity* y se considera que, la partición del conjunto de datos que posee cada una de ellas, es el conjunto completo de datos.

[4]https://wiki.python.org/moin/Generators

Esta estrategia permite aplicar el algoritmo TDS a porciones pequeñas del conjunto de datos haciendo uso del cómputo del entorno distribuido. Realizar este cálculo distribuido en una primera etapa evita tener que realizarlo tantas veces sobre el conjunto completo de datos, lo que refiere a una ejecución más costosa en términos de recursos del entorno.

Búsqueda de mejor especialización

Recordemos que una especialización se define como válida si, luego de aplicada sobre el conjunto de datos, el mismo no viola la condición de *k-anonymity*. Buscar la mejor especialización en cada iteración del algoritmo implica calcular la métrica `IGPL` para cada una de las especializaciones válidas, por lo que se deben calcular las métricas `IG` y `PL` para cada especialización.

A raíz de que el cálculo de la métrica `IGPL` se debe realizar en múltiples instancias durante la ejecución del proceso de anonimización, se decidió utilizar una estructura auxiliar al comenzar cada iteración la cuál contiene, para cada especialización válida, un objeto JSON donde las claves son los valores de las hojas del `TT` y los valores son una dupla que contiene la información del nodo padre (p_i) y del nodo hijo (c_i) más próximos en la especialización seleccionada. Mediante el uso de esta estructura, se logra acceder en O(1) a los valores de p_i y c_i en lugar de realizar la búsqueda en cada iteración.

Como resultado de este paso se genera, para cada partición, un AL'_i intermedio que es retornado al *driver*.

4.2.2. Segundo paso: combinación de niveles intermedios

Cuando el resultado de todas las particiones se encuentran en el *driver*, se combinan todos los `AL` intermedios para cada atributo con el fin de generar el AL^I, que será utilizado para la ejecución del algoritmo MRTDS sobre el conjunto de datos completo. El mismo se genera tomando el conjunto de todos los cortes correspondientes a cada atributo de los `AL` intermedios y encontrando, dentro del `TT` de cada atributo, el mínimo ancestro común a todos los cortes.

Para ejemplificar lo mencionado consideremos nuevamente el ejemplo presen-

tado en la Figura 4.2 y supongamos que se tienen dos `AL` intermedios (AL'_1 y AL'_2), donde solo consideraremos los valores para el atributo `marital-status`, los mismos se definen como:

$$AL'_1(\texttt{marital-status}) = \{\texttt{Single, Couple}\}$$

$$AL'_2(\texttt{marital-status}) = \{\texttt{Married-spouse-absent, Divorced, Separated,}$$

$$\texttt{Widowed, Never-married, Married-civ-spouse, Married-AF-spouse}\}$$

Representados en la Figura 4.3. En este ejemplo, el mínimo común ancestro para ambos cortes dentro del `TT` del atributo `marital-status`, corresponde al corte $AL'_1 = \{Single, Couple\}$, por lo tanto este será el corte seleccionado para este atributo para formar el AL^I.

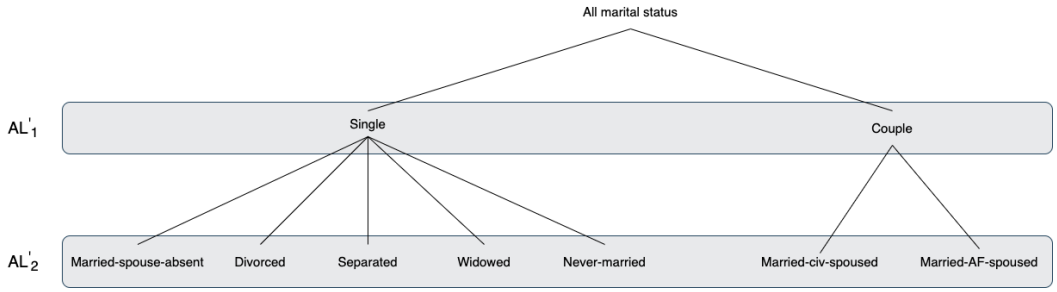


Figura 4.3: Ejemplo de $AL'_1(marital-status) = \{$Single, Couple$\}$ y $AL'_2(marital-status) = \{$Married-spouse-absent, Divorced, Separated, Widowed, Never-married, Married-civ-spouse, Married-AF-spouse$\}$ del árbol taxonómico para el atributo `marital-status`

4.2.3. Tercer paso: procesamiento MapReduce

El tercer paso del Algoritmo 4.2.1 refiere a ejecutar el algoritmo TDS sobre el conjunto completo de datos. A diferencia del primer paso donde el algoritmo se ejecuta de manera secuencial, en esta ocasión, la misma se realiza mediante un enfoque *MapReduce* con el fin de optimizar los recursos del entorno distribuido. Se utiliza el parámetro k para definir la condición de *k-anonymity* en lugar de k^I y el `AL` inicial es el computado como resultado del paso anterior (AL^I).

La ejecución del algoritmo con el AL^I sobre todo el conjunto de datos es necesaria porque, al combinar todos los niveles intermedios para construirlo,

se puede generar pérdida de información por el hecho de que el resultado de la combinación no garantiza alcanzar la especialización minimal. Sin embargo, al ejecutar el algoritmo TDS sobre el conjunto total, nos aseguramos que se alcance la especialización minimal buscada. Como resultado de la ejecución del algoritmo MRTDS, se obtiene el `AL` final, denominado AL^*, que será utilizado para anonimizar el conjunto de datos.

4.2.4. Cuarto paso: aplicación de AL final

El último paso de aplicar esta técnica de anonimización a un conjunto de datos refiere a transformar los registros del RDD para que cada valor, de cada atributo, tome el valor correspondiente a un nodo definido en el corte seleccionado para el atributo en AL^*. Por ejemplo, si $AL^* = \{Single, Couple\}$, todos los registros del conjunto de datos que presenten el valor `Divorced`, serán remplazados por el valor `Single`, ya que este corresponde al ancestro que pertenece al corte seleccionado para el atributo `marital-status` en AL^*.

Finalmente, luego de terminada la aplicación de la técnica de anonimización, se obtiene un conjunto de datos anonimizado representado en un RDD, el cuál puede ser retornado al usuario o almacenado en el *cluster*, dependiendo del valor proporcionado en la clave `save_anonymization_result` del parámetro `data_information` ingresado en la etapa de preprocesamiento de datos.

4.3. Módulo de validación

El módulo de validación evalúa por un lado los parámetros de entrada del módulo de preprocesamiento, y por otro el resultado del proceso de anonimización. Esta constituido por la clase `Test`. La misma requiere cinco parámetros de entrada: el conjunto de datos anonimizado, el parámetro de anonimización k utilizado, el conjunto de datos original, el nivel de anonimización final (AL^*) y el nivel de anonimización inicial (AL_0). Estos parámetros de entrada son utilizados para verificar si se cumplen cinco condiciones que son evaluadas en los módulos de preprocesamiento y anonimización.

Respecto al preprocesamiento se evalúan dos condiciones. La primera es que el valor del parámetro k no sea mayor que la cantidad de registros del conjunto

de datos, de lo contrario es imposible cumplir con la condición de *k-anonymity*. La segunda condición refiere a los árboles taxonómicos provistos por el usuario y verifica que los mismos estén balanceados y no presentan ciclos. En caso de no cumplirse alguna de las condiciones anteriores, se lanza una excepción informando al usuario que los datos ingresados no son validos. La condición de árboles balanceados se justifica porque nuestro algoritmo realiza especialización *full-domain*, esto indica que una vez seleccionada una especialización todos los valores de ese atributo deben pertenecer a algún valor del nodo especializado.

Respecto al módulo de anonimización, luego de finalizar la ejecución del algoritmo, se comprueba si el conjunto anonimizado cumple con la condición de *k-anonymity*. Para esto se cuenta la cantidad de ocurrencias de cada combinación de valores dentro de los atributos pertenecientes a `QI` y se compara con el parámetro de anonimización k. Si el número de ocurrencias de cada combinación supera o iguala este valor, entonces el conjunto cumple con la condición de *k-anonymity*. Otro aspecto que se evalúa en este módulo refiere a que la cantidad de registros presentes en el conjunto de datos original deber ser la misma que la del conjunto de datos anonimizado. Para verificar esta condición basta con evaluar la cantidad de registros de los dos conjuntos. Por último, la tercera condición a verificar alude a que todos los valores presentes en el conjunto de datos anonimizado deben pertenecer al corte seleccionado para cada atributo en AL^*. En caso de no cumplirse alguna de las condiciones, se lanza una excepción notificando al usuario.

Para la gestión del proyecto se utilizaron herramientas que permitieron cumplir con los tiempos pre-establecidos. Se usó Trello[5] para a la planificación y asignación de tareas, las mismas fueron divididas entre los integrantes del equipo de manera aleatoria y cada una fue definida con un objetivo puntual. Respecto al versionado del código se utilizó GitLab[6], respetando una metodología de trabajo que incluyó creación de *pull requests*[7] para permitir la revisión del código por parte de un integrante del equipo (distinto al desarrollador que lo creó).

[5]https://trello.com/
[6]https://gitlab.fing.edu.uy/maria.soledad.rivas/anonymization-project-groupbds
[7]https://docs.gitlab.com/ee/topics/gitlab_flow.html

En el siguiente capítulo se presentan los resultados obtenidos de utilizar la implementación realizada sobre dos conjuntos de datos.

Capítulo 5

Resultados experimentales

Las pruebas realizadas se dividen en dos tipos, las pruebas de comparación y las de rendimiento. Las primeas se realizaron para validar los resultados en comparación con otras herramientas de anonimización y fueron ejecutadas en un entorno local. Por otro lado, las pruebas de rendimiento fueron ejecutadas en el *cluster* con el fin de evaluar la ejecución de la biblioteca implementada en un entrono distribuido. De ahora en más, la solución implementada se denomina SparkTPTDS.

Si bien para llevar a cabo las pruebas se utilizó el conjunto de datos *Adult*, también se hizo uso del conjunto de datos *Autism*[1] con el fin de validar la implementación con otro conjunto de datos. El mismo tiene 704 registros que contienen datos personales y el atributo sensible indica si el individuo presenta algún trastorno del espectro autista. Los resultados obtenidos con este conjunto no son mencionados debido a que el mismo fue utilizado simplemente para validar la implementación y asegurar que la misma no esté sesgada a un conjunto de datos particular.

Dado que en la literatura no se encontró una forma estándar de comparar el resultado de diferentes procesos de anonimización, es que fue necesario definir una métrica para comparar los resultados de los distintos procesos. La definición de la misma plantea un desafío ya que es necesario definir qué aspectos de un conjunto anonimizado merece la pena cuantificar y cuáles no. En esta instancia, se decidió cuantificar la información perdida en el conjunto

[1] https://archive.ics.uci.edu/ml/datasets/Autism+Screening+Adult

de datos anonimizado Δ respecto al conjunto original Θ haciendo uso de los árboles taxonómicos.

5.1. Métrica de comparación

La métrica de comparación cuantifica la pérdida de información que se produce en un conjunto de datos al aplicar un algoritmo de anonimización sobre este. Esta métrica se presenta en la Definición 5.1.1.

Definición 5.1.1 ***Métrica de pérdida de información.*** *Consideremos el conjunto de datos anonimizado Δ, que cuenta con n cantidad de registros. Cada registro $r \in \Delta$ tiene la forma $r = (v_1, ..., v_m)$, donde $v_i \in DOM(TT_i)$ con $1 \leq i \leq m$, siendo m la cantidad de atributos de Δ. La métrica de comparación se define como*

$$\frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{m} \frac{H(v_j)}{H(TT_j)} \right)}{n}$$

Donde $H(TT_j)$ es la altura del árbol de taxonomía TT_j y $H(v_j)$ es la altura del valor v_j dentro de TT_j.

La idea detrás de esta métrica es la siguiente. Los atributos de los registros del conjunto original Θ presentan valores que se encuentran en las hojas de los árboles taxonómicos. Sin embargo, los atributos del conjunto Δ presentan valores que pertenecen a algún corte dentro de cada árbol taxonómico, pero este no necesariamente corresponde al corte de las hojas del mismo. Es utilizando la altura del corte seleccionado para cada atributo que se genera una noción de cuánta información perdió cada atributo respecto al conjunto de datos original. El valor de la métrica propuesta se incrementa a medida que el conjunto de datos pierde información durante el proceso de anonimización. Por tanto, cuanto más elevado sea su valor, significa que más información perdió el conjunto de datos.

Consideraremos que un registro $r = (X_1, ..., X_m)$ está conformado por m atributos. El nivel de anonimización final asociado a una ejecución se presenta como $AL = (N_1, ..., N_m)$, siendo N_i con $1 \leq i \leq m$ la altura del corte considerado en el árbol de taxonomía TT_i. Bajo estas condiciones, podemos apreciar que la mayor pérdida de información se genera cuando todas las entradas del

AL corresponden a las alturas de los árboles taxonómicos y denominaremos a este caso como AL_{max}.

5.2. Comparación de resultados con otras herramientas de anonimización

Con el propósito de evaluar los resultados de SparkTPTDS con respecto a otras herramientas de anonimización, se ejecutó una serie de pruebas de forma local. Las herramientas de anonimización utilizadas fueron ARX[2], una de las más utilizadas en el área, y UTD[3]. A continuación se brinda una descripción de ambas.

ARX Data Anonymization Tool es una herramienta de código abierto que permite transformar conjuntos de datos estructurados utilizando diferentes métodos de anonimización. Permite eliminar los atributos del tipo `Identificador` del conjunto de datos y aplicar transformaciones a los quasi-identificadores para minimizar el riesgo de re-identificación.

La herramienta provee varias técnicas de anonimización, entre ellas *k-anonymity*, así como modelos de transformación de datos como muestreo aleatorio o micro-agregación. Es capaz de manejar grandes conjuntos de datos y cuenta con una interfaz gráfica multiplataforma, además de una API de integración con Java para implementar algoritmos de anonimización de datos de forma programática en este lenguaje.

Para la ejecución del algoritmo *k-anonymity*, ARX puede utilizar varios algoritmos, entre ellos el de Generalización y Supresión, donde la supresión se realiza sobre un registro completo, dejando todos sus atributos en el valor más general de cada árbol de taxonomía. Además, la herramienta no considera el atributo sensible en el cálculo de la métrica de decisión y el máximo valor de k permitido es $k = 1.000$.

UTD es una herramienta de código abierto desarrollada por la *University*

[2]https://arx.deidentifier.org
[3]http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php

of Texas at Dallas, Data Security and Privacy Lab, que implementa varios métodos de anonimización para uso público por parte de investigadores. Estos métodos pueden ser utilizados directamente sobre un conjunto de datos mediante una interfaz gráfica o de forma programática mediante el uso de una biblioteca provista. Para su uso se debe especificar en un documento de configuración XML los siguientes parámetros: los árboles de taxonomía, el algoritmo de anonimización a utilizar y, en el caso de *k-anonymity*, el parámetro de anonimización k.

Cabe mencionar que ambas herramientas presentaron limitaciones a la hora de realizar los experimentos. Por ejemplo, ARX no considera la pérdida de información en el cálculo de las métricas de decisión, sino que garantiza la privacidad del conjunto de datos utilizando Generalización y Supresión. Sin embargo, presenta la ventaja de ser una herramienta muy rápida por el uso de memoria caché, almacenando estructuras auxiliares que permiten indexar los registros del conjunto de datos.

En contraste, UTD no limita el valor del parámetro k y considera tanto la pérdida de privacidad como la ganancia de información en el cálculo de las métricas de decisión, por lo tanto, la comparación con SparkTPTDS es más representativa. Sin embargo, presenta la desventaja de ser poco eficiente, no utiliza estructuras auxiliares causando que el tiempo de ejecución sea mayor a SparkTPTDS.

Para poder comparar el resultado obtenido con cada una de las herramientas es necesario utilizar la métrica de comparación definida anteriormente.

5.2.1. Resultados de la comparación

La pruebas realizadas con las herramientas ARX y UTD se ejecutaron en un entorno Windows, con un procesador Intel(R) Core(TM) i7-9750H 2.60 GHz (12 CPUs) y 16 GB de RAM. Para las mismas se utilizó el conjunto de datos *Adult*, el cuál cuenta con 32.561 registros, donde a 2.399 de ellos les falta el valor de por lo menos un atributo. Asimismo, otra característica que presenta el conjunto, es que para el atributo `native-country` el 89.5 % de los registros (29.143) poseen el valor `United-States`, lo que representa un sesgo

en los datos. Para las pruebas se utilizaron varios valores para el parámetro k, siendo estos 5, 50, 500, 1.000 y 5.000.

Consideraremos un registro $r = (X_1, ..., X_8, sv)$ está conformado por los atributos `sex`, `age`, `race`, `marital-status`, `education`, `native-country`, `workclass`, `occupation` y `class`, respectivamente. El nivel de anonimización final asociado a una ejecución se presenta como $AL = (N_1, ..., N_8)$ y, en el conjunto de datos *Adult*, este corresponde a $AL_{max} = (1, 4, 1, 2, 3, 3, 3, 1)$. Si computamos la métrica definida en la Definición 5.1.1 sobre este último, obtenemos que la mayor pérdida de información que se puede generar es 8, ya que se consideran ocho atributos y cada uno aporta una unidad a la métrica cuando la pérdida de información es máxima.

Los resultados obtenidos con ARX, UTD y SparkTPTDS se presentan en las tablas 5.1, 5.2 y 5.3 respectivamente.

Parámetro *k*	**AL final**	**Registros suprimidos**	**Tiempo (m)**
5	$(1, 3, 0, 0, 0, 0, 0, 1)$	4.982	1
50	$(1, 3, 0, 0, 3, 0, 0, 1)$	7.522	1
500	$(1, 3, 0, 2, 2, 0, 1, 1)$	6.835	1
1000	$(1, 4, 0, 2, 2, 0, 2, 1)$	10.487	1

Tabla 5.1: Resultados obtenidos con la herramienta ARX.

Parámetro *k*	**AL final**	**Registros suprimidos**	**Tiempo (m)**
5	$(0, 3, 0, 0, 1, 3, 2, 0)$	2.399	137
50	$(0, 3, 0, 0, 1, 3, 2, 0)$	2.399	109
500	$(0, 3, 0, 1, 2, 3, 2, 0)$	2.399	73
1000	$(1, 3, 0, 1, 2, 3, 2, 0)$	2.399	56
5000	$(1, 4, 0, 1, 2, 3, 2, 0)$	2.399	47

Tabla 5.2: Resultados obtenidos con la herramienta UTD.

A modo de comparación, se calculó la métrica definida en la Definición 5.1.1 sobre cada resultado obtenido. Esta información se brinda en la Tabla 5.4 y en ella se puede observar que el valor de la métrica para cada valor de k utilizado, está comprendido entre los valores 0 y 8, como se adelantó anteriormente.

En la Figura 5.7 se muestra de forma gráfica la pérdida de información para cada una de las herramientas en función del parámetro k.

Parámetro k	AL final	Registros suprimidos	Tiempo (m)
5	$(0, 1, 1, 0, 2, 3, 2, 0)$	0	2
50	$(1, 1, 1, 0, 2, 3, 2, 0)$	0	2
500	$(1, 1, 1, 0, 2, 3, 2, 0)$	0	2
1000	$(1, 1, 1, 0, 3, 3, 2, 1)$	0	1
5000	$(1, 3, 0, 0, 2, 3, 2, 1)$	0	1

Tabla 5.3: Resultados obtenidos con SparkTPTDS.

Parámetro k	SparkTPTDS	ARX	UTD
5	3,58	2,78	2,92
50	4,58	3,81	2,92
500	4,58	5,01	3,58
1000	4,91	5,54	4,58
5000	5,08	-	4,83

Tabla 5.4: Resultados del cálculo de la métrica definida para las herramientas ARX, UTD y SparkTPTDS.

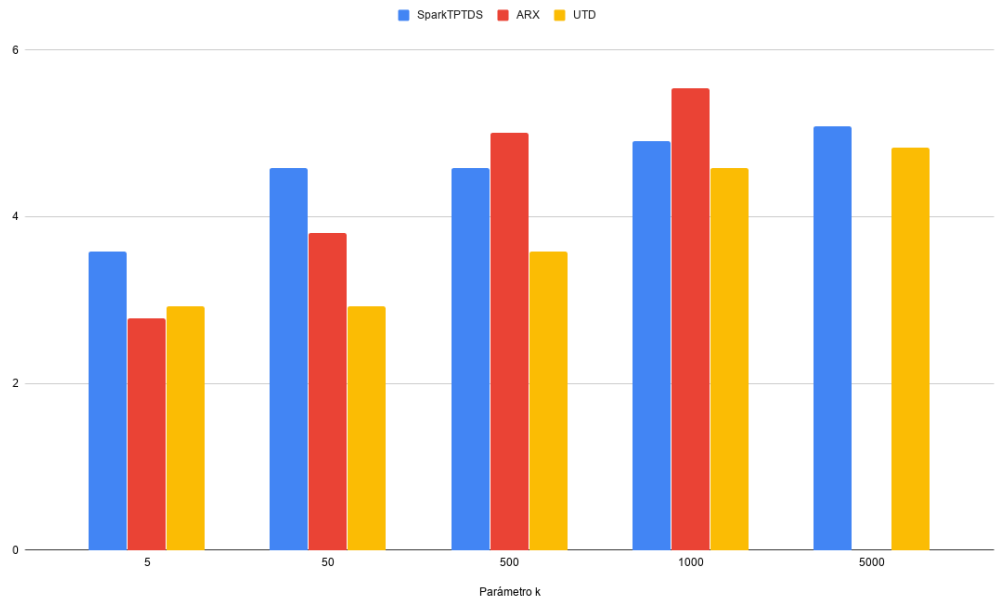


Figura 5.1: Gráfica comparativa de resultados entre las herramientas ARX, UTD y SparkTPTDS.

Durante la ejecución del proceso de anonimización con la herramienta ARX se observó que la misma suprime registros del conjunto de datos teniendo en cuenta cuán sesgado está un atributo, y a su vez, remueve aquellos registros

con valores de atributos faltantes. Por ejemplo, en el conjunto de datos utilizado, suprime los registros que no posean el valor `United-States` en el atributo `native-country`, provocando que el atributo tome un único valor y, en consecuencia, se pierde valor estadístico en el conjunto de datos anonimizado Δ.

Por otro lado, si bien la herramienta UTD y SparkTPTDS tienen en consideración la pérdida de información durante el proceso de anonimización, al momento de tomar la decisión de cuál atributo se debe generalizar/especializar se seleccionan distintos atributos. Esto es en parte causado porque UTD utiliza una técnica *Bottom-Up* [18] mientras que SparkTPTDS utiliza *Top-Down*.

En conclusión, si consideramos solamente la pérdida de información, la herramienta UTD es la que presenta mejores resultados. Sin embargo, el tiempo de demora la misma en ejecutar el proceso de anonimización es considerablemente más elevado que ARX y SparkTPTDS. Por otro lado, ambas herramientas suprimen registros del conjunto de datos mientras que SparkTPTDS no lo hace. Por lo tanto, SparkTPTDS presenta un balance entre la pérdida de información del conjunto de datos y el tiempo que requiere el proceso de anonimización.

5.3. Resultados obtenidos

Las pruebas de rendimiento fueron ejecutadas en el *cluster* de la Facultad de Ingeniería, UdelaR. En este, se desplegó la plataforma HDP configurada con un solo nodo y cuenta con 125 GB de memoria RAM y 32 CPUs. Lamentablemente, la implementación no pudo ser ejecutada en el entorno distribuido de Plan Ceibal debido a que SparkTPTDS se construyó utilizando Python 3.7 y la plataforma HDP de Ceibal posee Python 2.7, no siendo posible su actualización a la versión necesaria por posibles incompatibilidades con implementaciones que hoy en día son ejecutadas en esta.
Las versiones de las herramientas instaladas en el *cluster* de la Facultad de Ingeniería son las enumeradas a continuación.

- HDP (Hortonworks Data Platform): 3.0.1.
- Apache Hadoop: 3.1.1.
- Apache Spark: 2.3.1.

- MapReduce: 3.1.1.
- PySpark: 2.3.1.

Los casos de prueba ejecutados en el entorno distribuido, comparan los distintos resultados obtenidos con la ejecución de SparkTPTDS al variar sus parámetros de entrada. Estos parámetros determinan la condición de cada ejecución y permiten evaluar cómo se comporta la implementación en distintos escenarios. Los parámetros que se modifican en cada ejecución son los que definen la condición de anonimización, siendo estos k y k_I, el tamaño del conjunto de datos Θ inicial y la cantidad de particiones considerada. Por otro lado, los aspectos que se relevan al finalizar cada ejecución son el nivel de anonimización final alcanzado y el tiempo que requirió la ejecución.

Las pruebas se realizaron sobre diferentes variantes del conjunto de datos *Adult* [7]. Las variantes fueron obtenidas incrementando el volumen de datos del conjunto original mediante dos tipos de técnicas, de duplicación y de aleatoriedad. La técnica de duplicación toma el conjunto original y genera un nuevo conjunto duplicando o triplicando sus registros. Por otro lado, la técnica de aleatoriedad toma todos los valores presentes en el conjunto original para cada atributo y genera nuevo registros seleccionando aleatoriamente valores para cada atributo. Los conjuntos de datos generados con esta última presentan el término *Random* en su nombre. Para cada uno de estos conjuntos, se cumple que cada registro $r = (X_1, ..., X_{10}, sv)$ está conformado por los atributos `workclass`, `education`, `marital-status`, `occupation`, `relationship`, `race`, `sex`, `native-country`, `age`, `hours-per-week` y `class`. La totalidad de los conjuntos utilizados para las pruebas son listados en la Tabla 5.5.

Nombre del conjunto	**Cantidad de registros**
Adult	32.561
RandomAdult	32.561
Adulx2	65.122
RandomAdultx2	65.122
Adultx3	97.683
RandomAdultx3	97.683
Adultx10	325.610

Tabla 5.5: Conjuntos de datos usados para las pruebas.

Para cada uno de los conjuntos definidos anteriormente, se ejecutó SparkT-PTDS cinco veces, cada una con un parámetro de anonimización k distinto: 5, 50, 500, 1.000 y 5.000. Los valores de k_I utilizados fueron a 10, 100, 1.000, 2.000, 10.000 respectivamente. A su vez, cada ejecución se realizó considerando escenarios de 3, 5 y 20 particiones. A continuación se detallan los resultados obtenidos en cada caso de prueba.

Adult

			Tiempo (m) por particiones		
k	k^I	**AL final**	**3 part.**	**5 part.**	**20 part.**
5	10	$(3, 3, 1, 1, 0, 1, 1, 3, 3, 4)$	4	5	7
50	100	$(3, 2, 1, 1, 1, 1, 1, 3, 3, 3)$	5	6	9
500	1000	$(3, 2, 1, 1, 1, 1, 1, 3, 3, 4)$	4	4	7
1000	2000	$(3, 3, 1, 1, 1, 1, 1, 3, 3, 4)$	3	3	6
5000	10000	$(3, 2, 1, 1, 1, 1, 1, 3, 4, 4)$	3	3	5

Tabla 5.6: Resultados obtenidos con 3, 5 y 20 particiones en el conjunto de datos *Adult*.

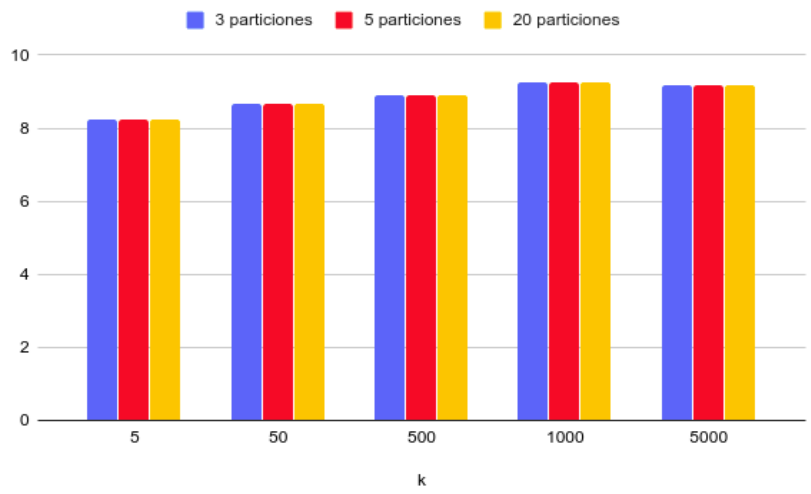


Figura 5.2: Pérdida de información con 3, 5 y 20 particiones en el conjunto de datos *Adult* en función del parámetro k.

Para el conjunto de datos *Adult* se observa que, para las distintas particiones evaluadas, el resultado del proceso de anonimización refiere al mismo AL. Esto implica que, si bien el tiempo de ejecución no es el mismo, la información

perdida es igual en las tres ejecuciones. Se debe considerar que cuanta mayor cantidad de particiones seleccionadas, mayor es el tiempo de ejecución. Este incremento de tiempo se podría explicar porque, al tener más particiones, la combinación de los AL' intermedios tiene como resultado que el corte seleccionado para cada atributo presente una altura cercana a la altura del árbol, implicando que el tercer paso del proceso de anonimización ejecutado sobre todo el conjunto de datos, requiera más tiempo.

RandomAdult

		Tiempo (m) por particiones		
k	k^I	**3 part.**	**5 part.**	**20 part.**
5	10	11	11	7
50	100	9	10	9
500	1000	7	6	7
1000	2000	7	7	6
5000	10000	4	5	5

Tabla 5.7: Resultados obtenidos con 3, 5 y 20 particiones en el conjunto de datos *RandomAdult*.

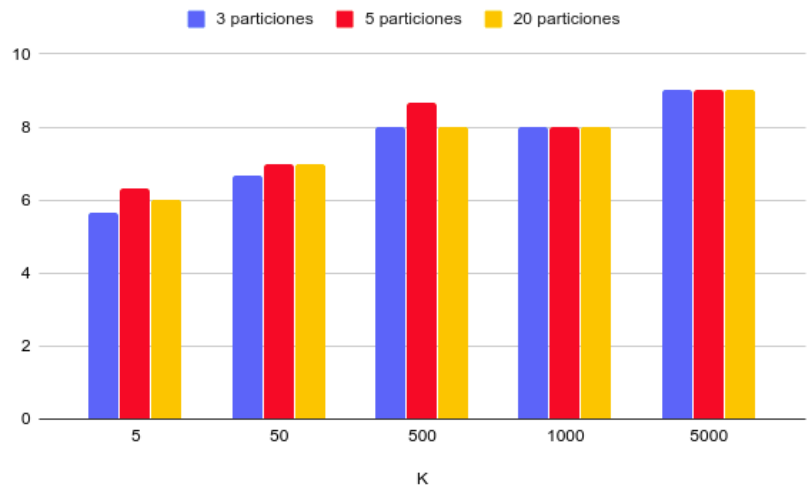


Figura 5.3: Pérdida de información con 3, 5 y 20 particiones en el conjunto de datos *RandomAdult* en función del parámetro k.

Para el caso del conjunto *RandomAdult*, las distintas particiones generan diferentes valores en la perdida de información, este caso es contrario al del conjunto *Adult*. Sin embargo, cuando el valor del parámetro se incrementa, el

resultado del proceso de anonimización es el mismo, se pierde la misma información para las diferentes particiones. El conjunto *RandomAdult* tiene la misma cantidad de registros que *Adult* pero con una distribución aleatoria, por lo tanto, si comparamos las figuras 5.2 y 5.3, se puede observar que se perdió menos información en el conjunto *RandomAdult* cuando k es pequeño.

*Adult*x2

		Tiempo (m) por particiones		
k	k^I	**3 part.**	**5 part.**	**20 part.**
5	10	5	8	26
50	100	4	10	24
500	1000	4	7	15
1000	2000	4	9	15
5000	10000	4	6	11

Tabla 5.8: Resultados obtenidos con 3, 5 y 20 particiones en el conjunto de datos *Adult*x2.

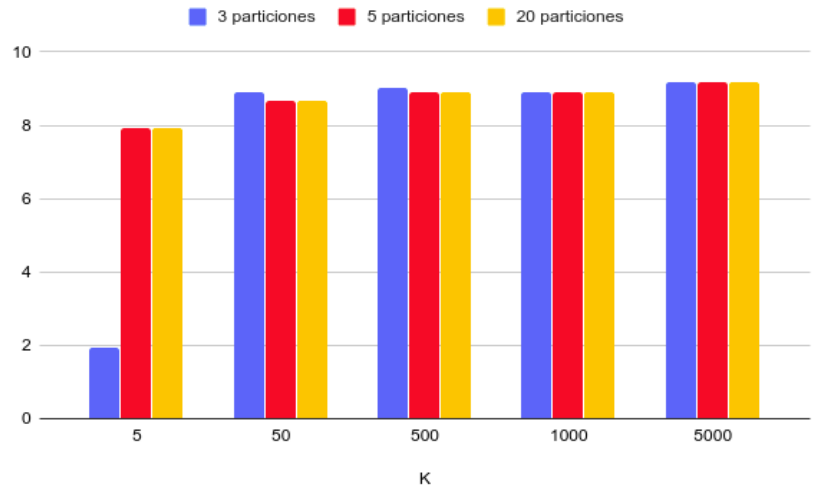


Figura 5.4: Pérdida de información con 3, 5 y 20 particiones en el conjunto de datos *Adult*x2 en función del parámetro k.

Los resultados en cuanto a la pérdida de información en el conjunto *Adult*x2 son similares a los del conjunto *Adult*, lo que indica que un aumento en la cantidad de registros que siguen la misma distribución estadística, termina obteniendo el mismo resultado en el proceso de anonimización.

***RandomAdult*x2**

			Tiempo (m) por particiones		
k	k^I	**AL final**	**3 part.**	**5 part.**	**20 part.**
5	10	$(3, 3, 2, 0, 0, 1, 1, 0, 2, 3)$	15	18	32
50	100	$(3, 3, 2, 0, 1, 1, 1, 0, 2, 3)$	12	14	24
500	1000	$(3, 3, 2, 0, 0, 1, 1, 3, 2, 3)$	4	7	11
1000	2000	$(2, 3, 2, 0, 1, 1, 1, 3, 2, 3)$	7	7	12
5000	10000	$(3, 3, 2, 1, 0, 1, 0, 3, 2, 3)$	4	7	11

Tabla 5.9: Resultados obtenidos con 3, 5 y 20 particiones en el conjunto de datos *RandomAdult*x2.

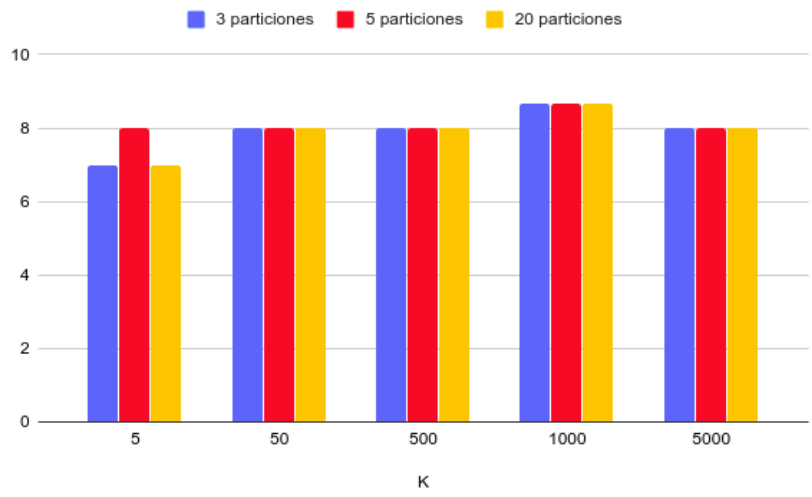


Figura 5.5: Pérdida de información con 3, 5 y 20 particiones en el conjunto de datos *RandomAdult*x2 en función del parámetro k.

Si consideramos los resultados obtenidos para el conjunto *RandomAdult*x2, observamos que son completamente distintos a los obtenidos con el conjunto *Random*. Esto es porque, por más que *RandomAdult*x2 contenga el doble de registros que *Random*, los mismos no son comparables dado que los valores de los atributos son aleatorios en ambos casos.

*Adult*x3

		Tiempo (m) por particiones		
k	k^I	**3 part.**	**5 part.**	**20 part.**
5	10	10	13	21
50	100	11	14	12
500	1000	12	13	21
1000	2000	9	11	16
5000	10000	7	8	12

Tabla 5.10: Resultados obtenidos con 3, 5 y 20 particiones en el conjunto de datos *Adult*x3.

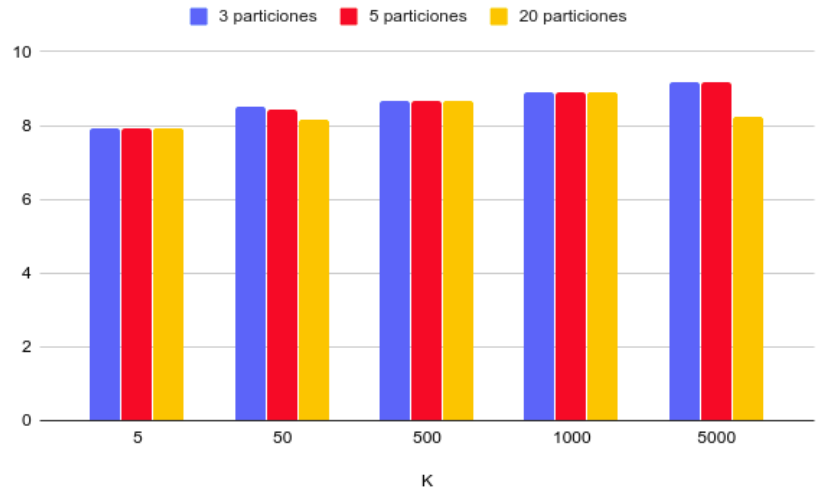


Figura 5.6: Pérdida de información con 3, 5 y 20 particiones en el conjunto de datos *Adult*x3 en función del parámetro k.

*RandomAdult*x3

			Tiempo (m) por particiones		
k	k^I	**AL final**	**3 part.**	**5 part.**	**20 part.**
5	10	$(2, 3, 2, 0, 1, 1, 0, 3, 2, 0)$	14	18	33
50	100	$(2, 3, 2, 0, 1, 1, 0, 3, 2, 1)$	11	14	26
500	1000	$(2, 3, 2, 0, 1, 1, 0, 3, 2, 2)$	8	11	19
1000	2000	$(2, 3, 2, 0, 1, 1, 0, 3, 2, 3)$	7	9	15
5000	10000	$(3, 3, 2, 0, 1, 1, 1, 3, 2, 3)$	4	4	7

Tabla 5.11: Resultados obtenidos con 3, 5 y 20 particiones en el conjunto de datos *RandomAdult*x3.

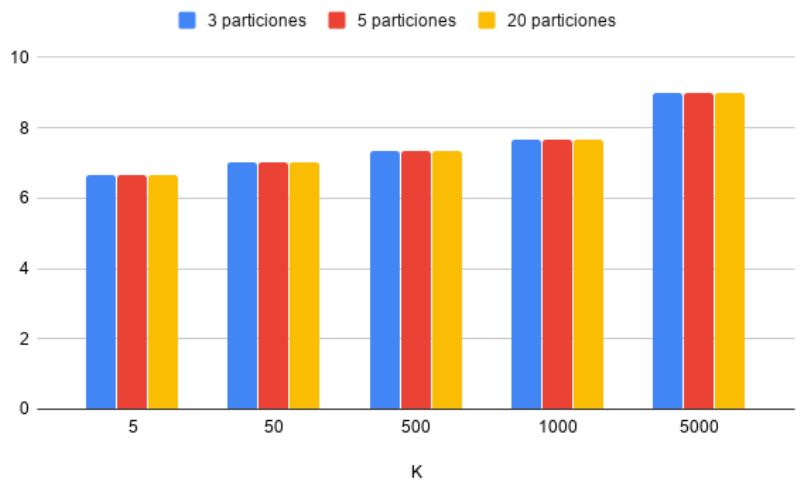


Figura 5.7: Pérdida de información con 3, 5 y 20 particiones en el conjunto de datos *RandomAdult*x3 en función del parámetro k.

*Adult*x10

			Tiempo (m) por particiones	
k	k^I	**AL final**	**5 part.**	**20 part.**
5	10	$(0,0,0,0,0,0,0,0,0,0)$	144	246
50	100	$(3,3,1,1,0,1,1,3,3,4)$	13	17
500	1000	$(3,2,1,1,1,1,1,3,3,3)$	14	23
1000	2000	$(3,2,1,1,1,1,1,3,3,3)$	14	23
5000	10000	$(3,2,1,1,1,1,1,3,3,4)$	12	18

Tabla 5.12: Resultados obtenidos con 5 y 20 particiones en el conjunto de datos *Adult*x10.

Como se muestra en la Figura 5.8, para la ejecución con 5 y 20 particiones, la pérdida de información obtenida fue la misma. Para el caso con 3 particiones SparkTPTDS presentó problemas de memoria dentro del entorno distribuido, motivo por el cuál no se pudo terminar de ejecutar dicho conjunto de pruebas. Además, una observación interesante es que para el parámetro de anonimización $k = 5$ no hubo ninguna pérdida de información, es decir, el conjunto anonimizado Δ resultó igual al conjunto original Θ. De esto podemos concluir que la elección del parámetro k es importante y no debe pasarse por alto.

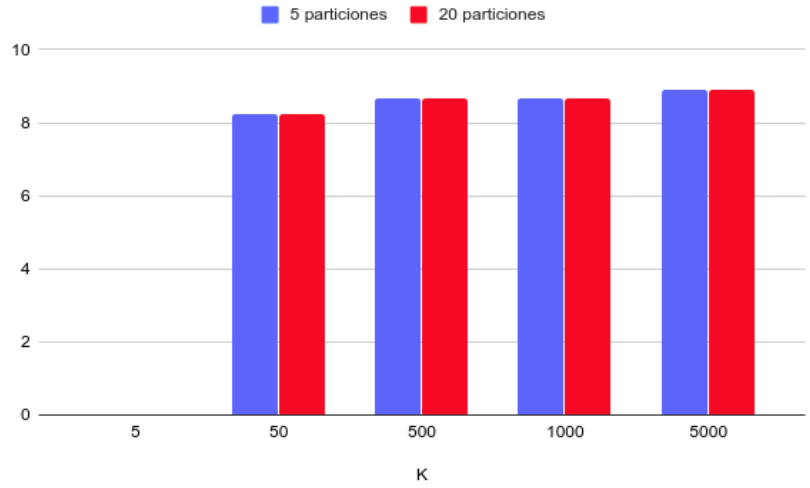


Figura 5.8: Pérdida de información con 5 y 20 particiones en el conjunto de datos *Adult*x10 en función del parámetro k.

Como última prueba se tomaron los tiempos de ejecución de SparkTPTDS con diferentes cantidades de registos en los conjunto de datos, partiendo como inicio del conjunto *Adult* y duplicandolo.

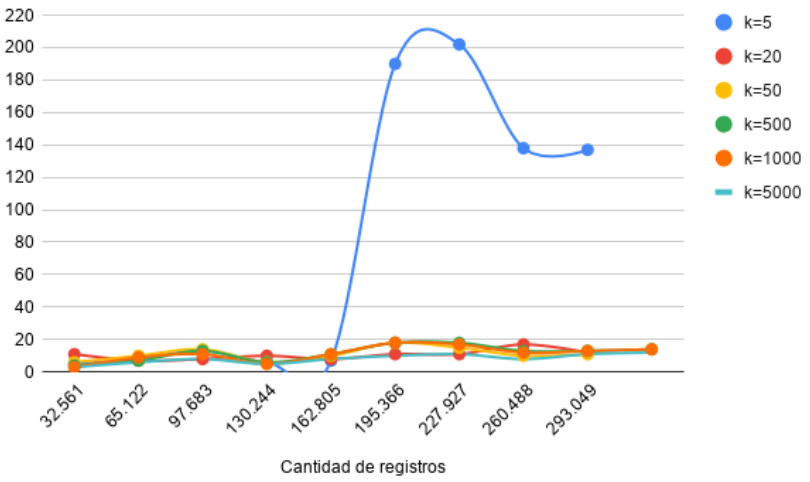


Figura 5.9: Tiempo en minutos de ejecuciones de SparkTPTDS en conjunto de datos *Adult* en función de la cantidad de registros para 5 particiones.

En la Figura 5.9 se observa que un aumento en la cantidad de registros del conjunto de datos, no implica el mismo crecimiento del tiempo de ejecución de SparkTPTDS. De la misma forma, los diferentes valores de k no implican tampoco un gran cambio. Para $k = 5$ se encuentra una excepción a lo anteriormente descrito, para grandes cantidades de registros se obtiene una demora

significativa en el tiempo de ejecución.

A partir de los resultados obtenidos hay dos conclusiones generales que podemos extraer. La primera refiere a la ejecución de SparkTPTDS, el mismo tiende a demorar más cuando el parámetro k es demasiado pequeño en comparación a la cantidad de registros del conjunto de datos, lo que es esperable, cuanto más pequeño es el valor de este parámetro, más especializaciones válidas se deberán considerar. La segunda conclusión es que la pérdida de información que presenta el conjunto de datos anonimizado se mantiene relativamente estable para los mismos valores del parámetro k, sin importar la cantidad de particiones utilizadas.

Capítulo 6

Conclusión y trabajo futuro

6.1. Conclusiones

A modo de síntesis, podemos concluir que se cumplió el objetivo planteado de implementar un algoritmo en Spark que transforme, de manera distribuida, el conjunto de datos de entrada en un conjunto de datos que cumpla la condición de *k-anonymity*. Para comprobar esta afirmación, se ejecutaron pruebas tanto en entornos centralizados como en entornos distribuidos y se realizaron observaciones generales a partir de ellas.

Como se mencionó y demostró a lo largo del documento, el proceso de anonimizar datos personales y mantener la mayor utilidad posible, no es una tarea trivial. Más difícil aún es comprobar que no es posible reidentificar a un individuo combinando el conjunto de datos anonimizado con otras fuentes de información, por ejemplo, bases de datos públicas, información que un potencial atacante posee sobre el contexto o algún individuo específico. Es por esto que es recomendable que el conjunto de datos anonimizado sea evaluado por alguien que tenga conocimiento del contexto y de los datos en sí, para poder determinar al nivel de riesgo al que se expondrán los datos al ser compartidos. Múltiples factores influyen en cual será el resultado de la anonimización, por ejemplo, la naturaleza de los datos, si se cuenta solamente atributos numéricos/categóricos o ambos, las características de los árboles taxonómicos, la cantidad de registros del conjunto de datos y los registros que forman cada partición.

Recordemos que *k-anonymity* brinda solamente protección para la divulgación de identidad, por lo cual la misma siempre debe ser complementada con técnicas que brinden protección ante la divulgación de atributos sensibles.

La mayor dificultad que presenta esta técnica es la elección de los parámetros del algoritmo, en particular del parámetro k. Utilizar un valor demasiado elevado aumenta la pérdida de información, mientras que un valor demasiado pequeño, causa que el riesgo de re-identificación aumente. Si bien la elección de este parámetro no es sencilla, es recomendable que esté a cargo de alguien con conocimiento del dominio debido a que podría conocer información relevante acerca de otros conjuntos de datos que hayan sido liberados previamente y, que en combinación con el conjunto de datos a liberar, puedan llevar a la re-identificación de individuos. Además de la elección de los parámetros de entrada, se debe considerar que las métricas utilizadas para cuantificar tanto la ganancia de información como la pérdida de anonimicidad, también influyen en el resultado final. Existen otras alternativas para cuantificar estos dos factores que pueden ser utilizadas y, dependiendo de la naturaleza de los datos, obtener mejores resultados según el criterio del usuario.

En resumen, si bien el proceso de anonimización presenta diversas dificultades y alternativas, debemos ser conscientes de su importancia y cuáles son las consecuencias de no realizarlo. Ejecutar por lo menos una técnica de anonimización sobre un conjunto de datos es mejor que no aplicar ninguna, ya que esto puede implicar que la privacidad de un individuo se vea afectada.

6.2. Trabajo futuro

A raíz de que la principal motivación del proyecto es contribuir con el Plan Ceibal en la anonimización de los datos que posee en su entorno distribuido, creemos pertinente destacar como primera posibilidad de trabajo futuro, ejecutar nuestra implementación en dicho entorno.

En otros aspectos, desde el comienzo del proyecto se consideró que la biblioteca de Python que contiene el algoritmo de anonimización, debía ser fácilmente extensible. Esto significa que agregar una nueva técnica que permita alcanzar la condición de *k-anonymity*, es tan sencillo como crear una nueva clase de

Python con sus respectivos métodos. Por lo tanto, una posibilidad de trabajo futuro refiere a agregar más implementaciones a la biblioteca, adicionando más técnicas de enmascaramiento sin perturbación o incluyendo alguna con perturbación. Asimismo, se puede flexibilizar la implementación actual permitiendo al usuario determinar algunos aspectos de la ejecución del algoritmo. Por ejemplo, en cuanto al preprocesamiento de datos, se podrían incorporar nuevas alternativas para completar los valores faltantes del conjunto de datos original. También se podría ampliar la implementación para que contemple no solo atributos numéricos enteros, sino que también contemple números reales. En el módulo de anonimización, se podrían utilizar métricas diferentes para definir cuál es la especialización a aplicar dentro del conjunto de especializaciones válidas.

Una dificultad que se plantea en todo proceso de anonimización es la definición de los quasi-identificadores, por lo tanto, una posible línea de trabajo futuro refiere a investigar y desarrollar mecanismos que dado un conjunto de datos, permita inferir cuáles son los atributos que deben ser considerados quasi-identificadores.

Otra posibilidad que sería tanto útil como interesante, sería desarrollar una técnica que anonimice los datos del conjunto basado en la información que potencialmente posea un atacante, por ejemplo, información extraída de fuentes públicas.

APÉNDICES

0.1. Manual de uso

A continuación se detalla el manual de uso de la biblioteca implementada, partiendo desde las dependencias necesarias hasta brindar un ejemplo de uso.

0.1.1. Configuración del entorno de ejecución

Las instalación de las dependencias varía según se esté ejecutando en un entorno local o en un entorno distribuido. Si se trabaja sobre un entorno local, se deben instalar las dependencias directamente en dicho entorno y establecer dos variables de entorno. Ésta instalación se puede realizar de forma global en el sistema operativo o en un entorno virtual aislado, utilizando herramientas como por ejemplo *virtualenv*[1] o *pipenv*[2]. Por otro lado, si se trabaja sobre un entorno distribuido que posea configurada de antemano la plataforma HDP, las dependencias se deben instalar en cada nodo del entorno (tanto en el maestro como en los nodos esclavos). Las dependencias necesarias se listan a continuación.

- AnyTree: 2.7.2
- JSONSchema: 3.1.1
- Numpy: 1.17.4
- PySpark: 2.3.1

Además, de utilizar una plataforma HDP, las versiones necesarias de cada componente son definidas en la siguiente lista.

- HDP: 3.0.1.0
- Hadoop: 3.1.1
- HDFS: 3.1.1
- YARN: 3.1.1

La instalación de la biblioteca implementada se realiza mediante el siguiente comando.

```
1 pip install -i https://test.pypi.org/simple/ k-anonymity
```

[1]https://virtualenv.pypa.io/en/latest/
[2]https://github.com/pypa/pipenv

0.1.2. Ejemplo de uso

A continuación se brinda un ejemplo de uso de la biblioteca implementada.

```
import os

from k_anonymity import DataPreprocess, TwoPhaseTDS


# Configuración de variables de entorno
# (solo necesario para ejecución local)
PYSPARK_PATH = <Ruta al directorio de instalacion de PySpark>
os.environ["PYSPARK_PYTHON"] = PYSPARK_PATH
os.environ["PYSPARK_DRIVER_PYTHON"] = PYSPARK_PATH

# Etapa de preprocesamiento
data_preprocess = DataPreprocess(
    data_information,
    categorical_trees
)

# Configuración de parámetros de anonimización
k = 500
k_i = 550

# Etapa de anonimización
tp_tds = TwoPhaseTDS(data_preprocess, k, k_i)
tp_tds.two_phase_tds()
```

En las pruebas realizadas sobre el algoritmo implementado, el parámetro `data_information` utilizado es el objeto JSON definido en el objeto 4.1 del Capítulo 4 y los árboles taxonómicos utilizados son ilustrados a continuación.

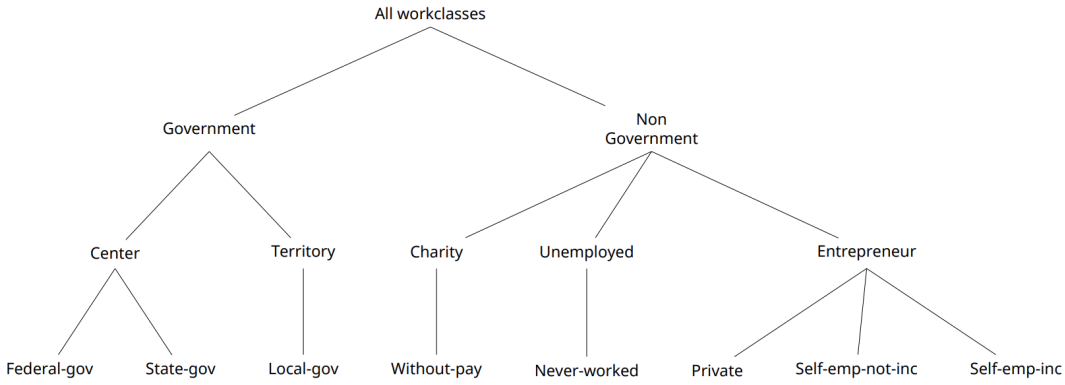


Figura 1: Árbol taxonómico utilizado para el atributo *workclass*.

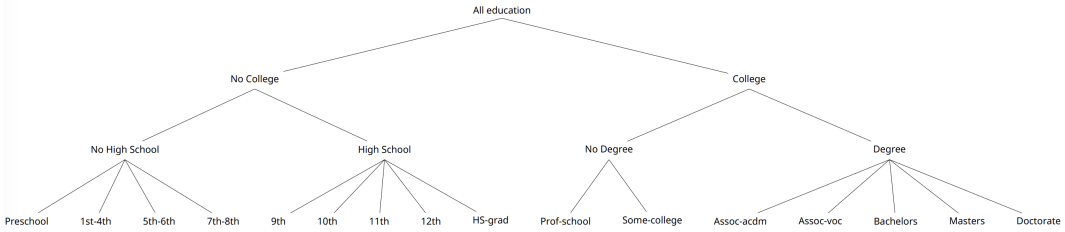


Figura 2: Árbol taxonómico utilizado para el atributo *education*.

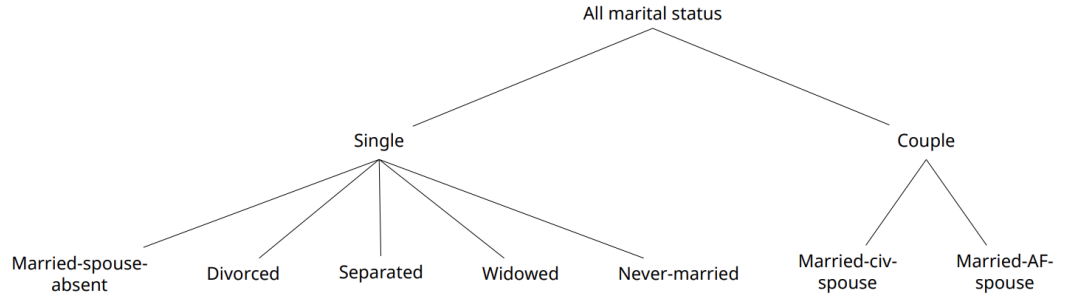


Figura 3: Árbol taxonómico utilizado para el atributo *maritalstatus*.

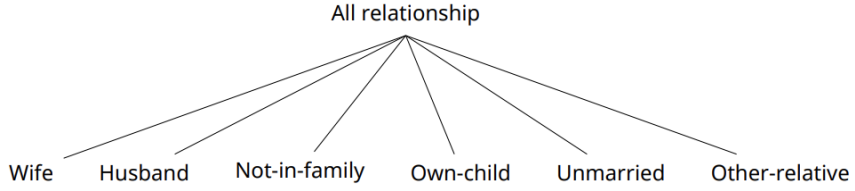


Figura 4: Árbol taxonómico utilizado para el atributo *relationship*.

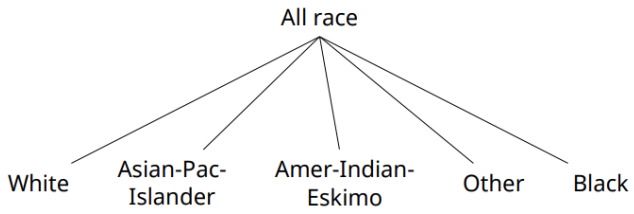


Figura 5: Árbol taxonómico utilizado para el atributo *race*.

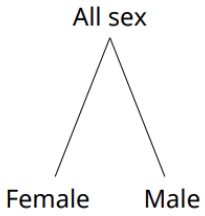


Figura 6: Árbol taxonómico utilizado para el atributo *sex*.

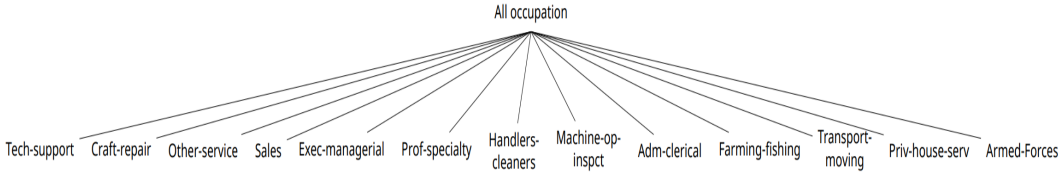


Figura 7: Árbol taxonómico utilizado para el atributo *occupation*.

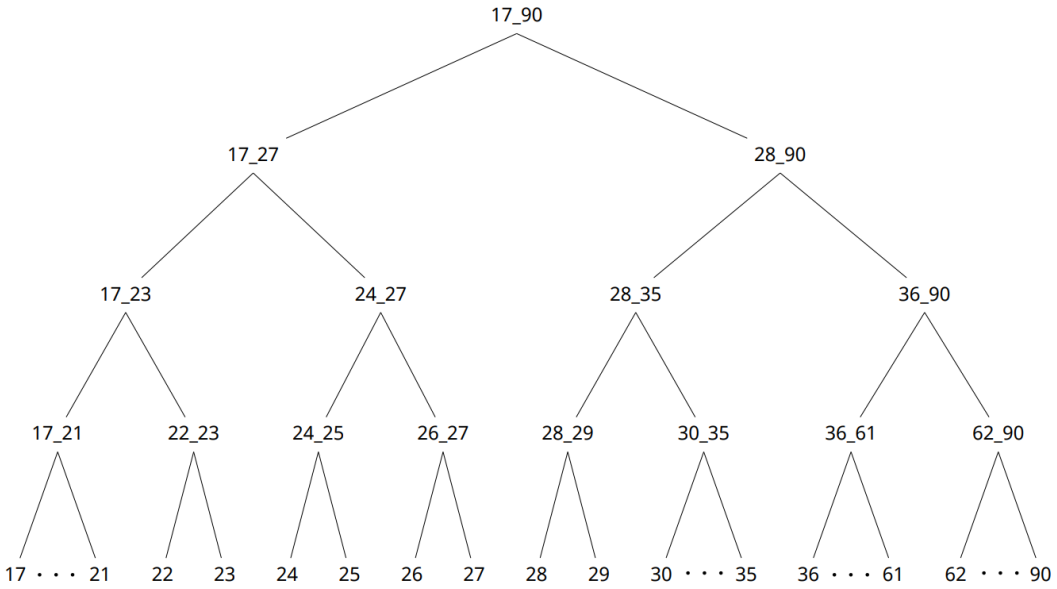


Figura 8: Árbol taxonómico generado para el atributo *age*.

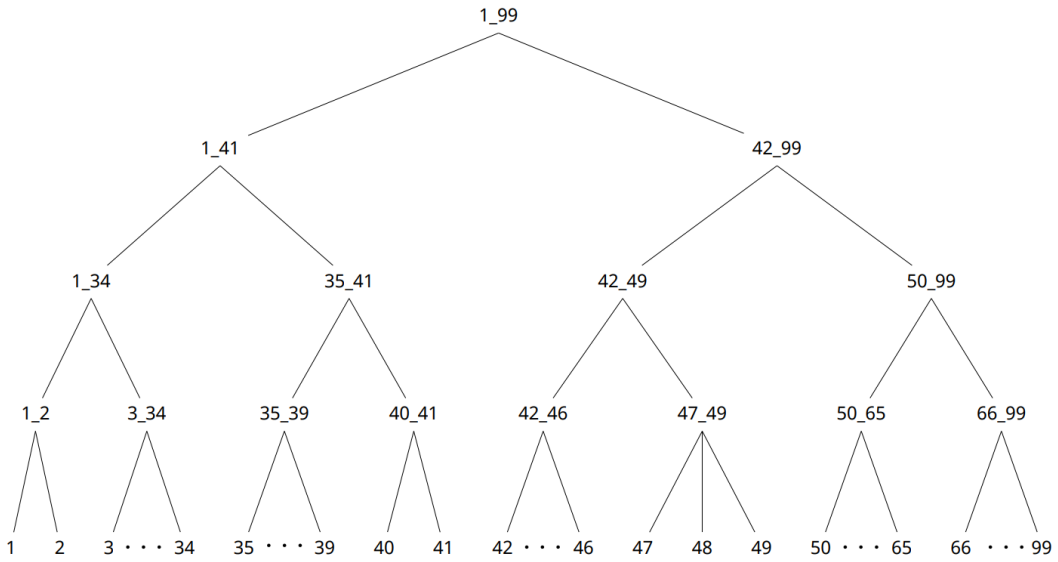


Figura 9: Árbol taxonómico generado para el atributo *hoursperweek*.

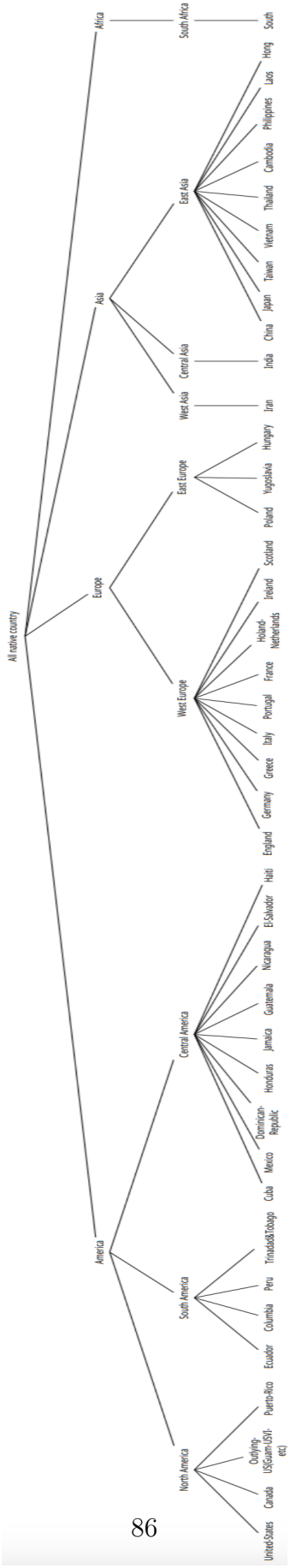


Figura 10: Árbol taxonómico utilizado para el atributo *nativecountry*.

Los árboles taxonómicos para atributos categóricos se definen mediante el objeto JSON definido a continuación.

```
{
    "workclass": {
        "value": "All workclasses",
        "children": [
            {
                "value": "Government",
                "children": [
                    {
                        "value": "Center",
                        "children": [
                            {"value": "Federal-gov"},
                            {"value": "State-gov"}
                        ]
                    },
                    {
                        "value": "Territory",
                        "children": [
                            {"value": "Local-gov"}
                        ]
                    }
                ]
            },
            {
                "value": "Non Government",
                "children": [
                    {
                        "value": "Charity",
                        "children": [
                            {"value": "Without-pay"}
                        ]
                    },
                    {
                        "value": "Unemployed",
                        "children": [
                            {"value": "Never-worked"}
                        ]
                    },
                    {
                        "value": "Entrepreneur",
                        "children": [
                            {"value": "Private"},
```

```
                            {"value": "Self-emp-not-inc"},
                            {"value": "Self-emp-inc"}
                        ]
                    }
                ]
            }
        ]
    },
    "education": {
        "value": "All education",
        "children": [
            {
                "value": "No College",
                "children": [
                    {
                        "value": "No High School",
                        "children": [
                            {"value": "Preschool"},
                            {"value": "1st-4th"},
                            {"value": "5th-6th"},
                            {"value": "7th-8th"}
                        ]
                    },
                    {
                        "value": "High School",
                        "children": [
                            {"value": "9th"},
                            {"value": "10th"},
                            {"value": "11th"},
                            {"value": "12th"},
                            {"value": "HS-grad"}
                        ]
                    }
                ]
            },
            {
                "value": "College",
                "children": [
                    {
                        "value": "No Degree",
                        "children": [
                            {"value": "Prof-school"},
                            {"value": "Some-college"}
```

```
                        ]
                    },
                    {
                        "value": "Degree",
                        "children": [
                            {"value": "Assoc-acdm"},
                            {"value": "Assoc-voc"},
                            {"value": "Bachelors"},
                            {"value": "Masters"},
                            {"value": "Doctorate"}
                        ]
                    }
                ]
            }
        ]

    },
    "marital-status": {
        "value": "All marital status",
        "children": [
            {
                "value": "Single",
                "children": [
                    {"value": "Married-spouse-absent"},
                    {"value": "Divorced"},
                    {"value": "Separated"},
                    {"value": "Widowed"},
                    {"value": "Never-married"}
                ]
            },
            {
                "value": "Couple",
                "children": [
                    {"value": "Married-civ-spouse"},
                    {"value": "Married-AF-spouse"}
                ]
            }
        ]

    },
    "occupation": {
        "value": "All occupation",
        "children": [
```

```
            {"value": "Tech-support"},
            {"value": "Craft-repair"},
            {"value": "Other-service"},
            {"value": "Sales"},
            {"value": "Exec-managerial"},
            {"value": "Prof-specialty"},
            {"value": "Handlers-cleaners"},
            {"value": "Machine-op-inspct"},
            {"value": "Adm-clerical"},
            {"value": "Farming-fishing"},
            {"value": "Transport-moving"},
            {"value": "Priv-house-serv"},
            {"value": "Protective-serv"},
            {"value": "Armed-Forces"}
        ]
    },
    "relationship": {
        "value": "All relationship",
        "children": [
            {"value": "Wife"},
            {"value": "Husband"},
            {"value": "Not-in-family"},
            {"value": "Own-child"},
            {"value": "Unmarried"},
            {"value": "Other-relative"}
        ]
    },
    "race": {
        "value": "All race",
        "children": [
            {"value": "White"},
            {"value": "Asian-Pac-Islander"},
            {"value": "Amer-Indian-Eskimo"},
            {"value": "Other"},
            {"value": "Black"}
        ]
    },
    "sex": {
        "value": "All sex",
        "children": [
            {"value": "Female"},
            {"value": "Male"}
        ]
```

```
    },
    "native-country": {
        "value": "All native country",
        "children": [
            {
                "value": "America",
                "children": [
                    {
                        "value": "North America",
                        "children": [
                            {"value": "United-States"},
                            {"value": "Canada"},
                            {"value": "Outlying-US"},
                            {"value": "Puerto-Rico"}
                        ]
                    },
                    {
                        "value": "South America",
                        "children": [
                            {"value": "Ecuador"},
                            {"value": "Columbia"},
                            {"value": "Peru"},
                            {"value": "Trinadad&Tobago"}
                        ]
                    },
                    {
                        "value": "Central America",
                        "children": [
                            {"value": "Cuba"},
                            {"value": "Mexico"},
                            {"value": "Dominican-Republic"},
                            {"value": "Honduras"},
                            {"value": "Jamaica"},
                            {"value": "Guatemala"},
                            {"value": "Nicaragua"},
                            {"value": "El-Salvador"},
                            {"value": "Haiti"}
                        ]
                    }
                ]
            },
            {
                "value": "Europe",
```

```
                "children": [
                    {
                        "value": "West Europe",
                        "children": [
                            {"value": "England"},
                            {"value": "Germany"},
                            {"value": "Greece"},
                            {"value": "Italy"},
                            {"value": "Portugal"},
                            {"value": "France"},
                            {"value": "Holand-Netherlands"},
                            {"value": "Ireland"},
                            {"value": "Scotland"}
                        ]
                    },
                    {
                        "value": "East Europe",
                        "children": [
                            {"value": "Poland"},
                            {"value": "Yugoslavia"},
                            {"value": "Hungary"}
                        ]
                    }
                ]
            },
            {
                "value": "Asia",
                "children": [
                    {
                        "value": "West Asia",
                        "children": [
                            {"value": "Iran"}
                        ]
                    },
                    {
                        "value": "Central Asia",
                        "children": [
                            {"value": "India"}
                        ]
                    },
                    {
                        "value": "East Asia",
                        "children": [
```

```
                                {"value": "China"},
                                {"value": "Japan"},
                                {"value": "Taiwan"},
                                {"value": "Vietnam"},
                                {"value": "Thailand"},
                                {"value": "Cambodia"},
                                {"value": "Philippines"},
                                {"value": "Laos"},
                                {"value": "Hong"}
                            ]
                        }
                    ]
                },
                {
                    "value": "Africa",
                    "children": [
                        {
                            "value": "South Africa",
                            "children": [
                                {"value": "South"}
                            ]
                        }
                    ]
                }
            ]
        }
}
```

Listing 1: Objeto JSON ejemplo para el parámetro `categorical_trees`.

Referencias bibliográficas

[1] Plan Ceibal. https://www.ceibal.edu.uy/es/institucional/, 2019. Accedido: Set 2019.

[2] Tom White. Hadoop: The Definitive Guide. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596521979, 9780596521974.

[3] Thomas M. Mitchell. Machine Learning. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

[4] Ley de protección de datos personales. https://www.impo.com.uy/bases/leyes/18331-2008, 2019. Accedido: Nov 2019.

[5] J. Domingo-Ferrer, D. Sanchez, and J. Soria-Comas. Database Anonymization: Privacy Models, Data Utility, and Microaggregation-based Inter-model Connections. Morgan Claypool, 2016. URL https://ieeexplore.ieee.org/document/7385400.

[6] Latanya Sweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[7] Conjunto de datos Adult. https://archive.ics.uci.edu/ml/datasets/Adult, 2019. Accedido: Set 2019.

[8] Rob Hall and Stephen E. Fienberg. Privacy-Preserving Record Linkage. In Proceedings of the 2010 International Conference on Privacy in Statistical Databases, PSD'10, page 269–283, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3642158374.

[9] SDC. https://sdctheory.readthedocs.io/en/latest/, 2020. Accedido: Mar 2020.

[10] Yang Xu, Tinghuai Ma, Meili Tang, and Wei Tian. A survey of privacy preserving data publishing using generalization and suppression. Applied Mathematics & Information Sciences, 8(3):1103, 2014.

[11] Adam Meyerson and Richard Williams. On the Complexity of Optimal K-Anonymity. Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 23:223–228, 01 2004. doi: 10.1145/1055558.1055591.

[12] X. Zhang, L. T. Yang, C. Liu, and J. Chen. A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud. IEEE Transactions on Parallel and Distributed Systems, 25(2):363–373, Feb 2014. doi: 10.1109/TPDS.2013.48.

[13] Yang Xu, Tinghuai Ma, Meili Tang, and Wei Tian. A Survey of Privacy Preserving Data Publishing using Generalization and Suppression. Applied Mathematics Information Sciences, 8:1103–1116, 05 2014. doi: 10.12785/amis/080321.

[14] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In 21st International Conference on Data Engineering (ICDE'05), pages 205–216, April 2005. doi: 10.1109/ICDE.2005.143.

[15] K. Wang B.C.M. Fung and P.S. Yu. Anonymizing Classification Data for Privacy Preservation. IEEE Trans. Knowledge and Data Eng., 19:711–725, May 2007.

[16] L. Wang B. Fung, K. Wang and P.C.K. Hung. Privacy Preserving Data Publishing for Cluster Analysis. Data and Knowledge Eng., 68:552–575, 2009.

[17] Benjamin C.M. Fung, Ke Wang, Ada Wai-Chee Fu, and Philip S. Yu. Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques. Chapman Hall/CRC, 1st edition, 2010. ISBN 1420091484.

[18] Ke Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: a data mining solution to privacy protection. In Fourth IEEE International Conference on Data Mining (ICDM'04), pages 249–256, Nov 2004. doi: 10.1109/ICDM.2004.10110.

[19] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient k-Anonymization Using Clustering Techniques. DASFAA 2007. LNCS, 4443: 188–200, 02 2007. doi: 10.1007/978-3-540-71703-4_18.

[20] T. Zhu, G. Li, W. Zhou, and P. S. Yu. Differentially private data publishing and analysis: A survey. IEEE Transactions on Knowledge and Data Engineering, 29(8):1619–1638, Aug 2017. doi: 10.1109/TKDE.2017.2697856.

[21] Guy Harrison. Next Generation Databases: NoSQL and Big Data. Apress, Berkely, CA, USA, 1st edition, 2015. ISBN 1484213300, 9781484213308.

[22] Hadoop. `https://hadoop.apache.org/`, 2019. Accedido: Set 2019.

[23] Apache. `https://www.apache.org/`, 2019. Accedido: Set 2019.

[24] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04: Sixth Symposium on Operating System Design and Implementation, pages 137–150, San Francisco, CA, 2004.