UNMDP - Facultad de Ingeniería - Departamento de Informática

DESCUBRIMIENTO DE CONOCIMIENTO PARA LA GESTIÓN EN EL ÁREA DE SALUD: APLICACIÓN A DATOS DE COVID-19

Autores: Ferraris Ignacio (ferrarisnacho@gmail.com),

Gabbanelli Lucia (luuucia3@gmail.com)

Directora: Dra. Seijas Leticia M.

Co-Director: Ing. Mileta Srecko Estanislao

Proyecto Final para optar al grado de Ingeniero en Informática

Mar del Plata, febrero del 2023

UNMDP - Facultad de Ingeniería - Departamento de Informática

DESCUBRIMIENTO DE CONOCIMIENTO PARA LA GESTIÓN EN EL ÁREA DE SALUD: APLICACIÓN A DATOS DE COVID-19

Autores: Ferraris Ignacio (ferrarisnacho@gmail.com),

Gabbanelli Lucia (luuucia3@gmail.com)

Directora: Dra. Seijas Leticia M.

Co-Director: Ing. Mileta Srecko Estanislao

Proyecto Final para optar al grado de Ingeniero en Informática

Mar del Plata, febrero del 2023

Índice general

Índice de gráficos y figuras

Índice de tablas

Agradecimientos

En primer lugar queremos agradecer a nuestros familiares y amigos por todo el amor y el apoyo brindado a lo largo de los años. Hicieron posible que no bajemos los brazos en los momentos difíciles y estuvieron junto a nosotros para celebrar nuestros logros.

Queremos mencionar, además, a nuestros amigos y compañeros de facultad, quienes alegraron cada cursada y nos ayudaron a seguir pese a las dificultades del camino. Gracias a ellos, nos llevamos los mejores recuerdos.

También, destacar el gran compromiso por parte de nuestros directores, la Dra. Leticia M. Seijas y el Ing. Srecko Estanislao Mileta por guiarnos y acompañarnos en este proyecto. Siempre dispuestos a ayudarnos y a brindarnos sus conocimientos.

Por último, queremos agradecer a todos los docentes de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata por su dedicación y aporte de conocimientos que hicieron posible nuestra formación como ingenieros informáticos.

Muchas gracias a todos.

Resumen

Actualmente, impulsados por los avances tecnológicos en la adquisición, el procesamiento, el almacenamiento y la difusión de datos, las organizaciones disponen de conjuntos de datos cada vez más grandes y complejos. Entonces, para encontrar la información requerida, descubrir patrones novedosos y realizar una categorización se hace uso de las técnicas de descubrimiento de conocimiento en bases de datos (*Knowledge Discovery in Databases* - KDD) y *data mining*.

En particular, la utilización de *data mining* en bases de datos hospitalarias tiene un gran potencial para explorar patrones ocultos en conjuntos de datos de dominio médico debido a su naturaleza voluminosa, heterogénea y distribuida. Estos patrones se pueden utilizar para el diagnóstico clínico y gestión de recursos, entre otros. Dada la situación de pandemia que se ha vivido recientemente, el descubrimiento de conocimiento para la eficiencia en la toma de decisiones se vuelve imprescindible.

El presente trabajo final tiene como objetivo hacer un aporte a la gestión en salud dentro de la comunidad de Mar del Plata haciendo uso de bases de datos vinculadas al COVID-19 de carácter público y privado. El uso de información privada fue posible gracias a la colaboración y buena predisposición del Hospital Privado de Comunidad (HPC).

A lo largo del proyecto se expone tanto la investigación, evaluación e implementación de métodos de *data mining* y descubrimiento de conocimiento como el análisis, documentación y visualización de los resultados obtenidos. Especialmente, se utilizan técnicas de *clustering* como k-means, k-prototypes y una red neuronal artificial en el paradigma del aprendizaje competitivo no supervisado, los mapas auto-organizados de Kohonen (SOM), obteniéndose mapas de alta calidad que permiten realizar un análisis profundo de un gran volumen de información en base de datos a nivel nacional y local.

Además, se ha aplicado una técnica híbrida que combina SOM y k-means con fines de refinamiento.

Los resultados de k-prototypes permitieron obtener un panorama general de la distribución de las muestras, mientras que en los mapas auto-organizados, con medidas de evaluación de modelos de gran calidad, posibilitaron un análisis más completo y profundo.

Finalmente, se presenta un software desarrollado en MATLAB© para que los expertos puedan hacer un mayor estudio de los resultados.

Palabras Clave: Descubrimiento de Conocimiento, Data Mining, Clustering, K-Means, K-Prototypes, Mapas Auto-Organizados (SOM), Bases de Datos, COVID-19, Patrones.

Abstract

Today, driven by technological advances in data acquisition, processing, storage and dissemination, organizations have increasingly large and complex data sets. Knowledge Discovery in Databases (KDD) and data mining techniques are used to find the required information, discover novel patterns and perform categorization.

In particular, the use of data mining in hospital databases has great potential to explore hidden patterns in medical domain datasets due to their voluminous, heterogeneous and distributed nature. These patterns can be used for clinical diagnosis and resource management, among others. Given the pandemic situation that has been experienced recently, knowledge discovery for efficiency in decision making becomes imperative.

This final work aims to make a contribution to health management in the community of Mar del Plata using public and private databases linked to COVID-19. The use of private information was possible thanks to the collaboration and willingness of the Hospital Privado de Comunidad (HPC).

Throughout the project, the research, evaluation and implementation of data mining and knowledge discovery methods as well as the analysis, documentation and visualization of the results obtained are presented. Specially, clustering techniques such as k-means, k-prototypes and an artificial neural network in the paradigm of unsupervised competitive learning, Kohonen's self-organizing maps, are used, obtaining high quality maps that allow a deep analysis of a large volume of information in a national and local database. In addition, a hybrid technique combining SOM and k-means has been applied for refinement purposes.

The results of k-prototypes allowed to obtain an overview of the distribution of the samples, while self-organized maps, with high quality model evaluation measures, made possible a more complete and deep analysis.

Finally, a software developed in MATLAB© is presented for further study of the results by experts.

Keywords: Knowledge Discovery, Data Mining, Clustering, K-Means, K-Prototypes, Self-Organizing Maps (SOM), Databases, COVID-19, Patterns.

Abreviaturas

KDD - **K**nowledge **D**iscovery in **D**atabases - Descubrimiento de conocimiento en bases de datos

SOM - **S**elf-**O**rganizing **M**ap - Mapa auto-organizado

HPC - **H**ospital **P**rivado de **C**omunidad

RNA - **R**edes **N**euronales **A**rtificiales

BMU - Best **M**atching **U**nit - Unidad de mejor correspondencia

U-Matrix - **U**nified **D**istance **M**atrix - Matriz de distancias unificadas

WCSS - **W**ithin-**C**luster **S**um of **S**quares - Suma de los cuadrados intra-cluster

FMSOM - **F**requency neuron **M**ixed **S**elf-**O**rganizing **M**ap - Mapa auto-organizado mixto de neuronas de frecuencia

GSOM - **G**eneralized **S**elf-**O**rganizing **M**ap - Mapa auto-organizado generalizado

IDE - Integrated **D**evelopment **E**nvironment - Entorno de desarrollo integrado

CSV - **C**omma **S**eparated **V**alues - Valores separados por comas

F - **F**emenino

M - **M**asculino

NR - **N**o **R**egistrado

QE - Quantization **E**rror - Error de cuantización

TE - Topographic **E**rror - Error topográfico

DB - **D**avies-**B**ouldin

CRISP-DM - **C**ross-**I**ndustry **S**tandard **P**rocess for **D**ata **M**ining - Proceso estándar trans-industrial para minería de datos

1. Introducción

1.1. Introducción general

El aumento del volumen y variedad de información que está digitalizada en bases de datos y otras fuentes creció exponencialmente en las últimas décadas. Gran parte de esta información es histórica, por lo tanto, cumple la función de "memoria de la organización". Esto es útil para explicar el pasado, entender el presente y predecir la información futura. La mayoría de las decisiones de empresas, organizaciones e instituciones se basan en información sobre experiencias pasadas.

En muchas situaciones, el método tradicional de transformar los datos en conocimiento consiste en un análisis e interpretación realizada de forma manual. Esta forma tiene como desventaja ser lenta, cara y altamente subjetiva. Además, el análisis manual es impracticable en dominios donde el volumen de los datos crece exponencialmente ya que desborda la capacidad humana de comprenderlos. Estos problemas y limitaciones hicieron surgir la necesidad de utilizar herramientas y técnicas digitales para soportar la extracción de conocimiento útil desde la información disponible y que se engloban bajo la denominación de minería de datos.

La minería de datos o *data mining* forma parte del proceso de KDD y se define como el procesamiento de los datos, a través de distintas tecnologías, para encontrar patrones de comportamiento que sean de utilidad para la toma de decisiones. Es importante aclarar que, si bien en ocasiones también se la referencia incorrectamente como descubrimiento de conocimiento en bases de datos (*knowledge discovery in databases*), el conocimiento será el que el experto del dominio pueda descubrir e interpretar a partir de los patrones encontrados.

A pesar de que los objetivos de aplicar minería sobre las bases de datos varían con las distintas áreas (Médica, Marketing, Comercial, etc.) en general puede decirse que están

orientados a determinar qué o quién realiza o posee qué cosa bajo qué circunstancia, para luego poder llevar a cabo una acción correctiva o beneficiosa al respecto [1, 2].

1.2. Objetivos del proyecto

El objetivo principal del proyecto es poder hacer un aporte a los expertos del área de Salud de Mar del Plata a través de la presentación detallada de un proceso completo para la potencial obtención de conocimiento sobre bases de datos vinculadas a COVID-19 siendo este replicable con otros conjuntos de datos de interés. Se suma a lo anterior la posibilidad de obtener información útil, para la posterior toma de decisiones, utilizando el prototipo desarrollado para el análisis de los resultados emergentes.

Para lograrlo, se utilizaron técnicas específicas sobre la base de datos del Hospital Privado de Comunidad y la publicada por el Ministerio de Salud sobre casos de COVID-19 a nivel nacional.

Se siguieron todas las etapas del proceso KDD para la extracción de conocimiento utilizando específicamente k-prototypes [12], k-means [37, 38, 39, 40] y self-organizing map [13] en la fase de *data mining*.

Los objetivos secundarios son:

- Incorporación de metodología para la gestión y desarrollo de un proyecto.
- Obtención de un software en versión prototipo para el análisis de resultados.
- Estudio de la problemática actual vinculada a disponibilidad de grandes volúmenes de datos y necesidad de descubrimiento de conocimiento para toma de decisiones orientado a la situación de pandemia y el contexto de gestión en salud.

- Estudio y aplicación de técnicas representativas de *clustering* como k-means y self-organizing map vinculadas a *data mining*.
- Aplicación de las técnicas sobre bases de datos públicas de complejidad simple y media.
- Elección de software para el uso de las técnicas anteriormente mencionadas.
- Puesta a punto de algoritmos, análisis y comparación de resultados.
- Aplicación de las técnicas estudiadas a un problema más complejo (estructura de la base de datos y volumen de datos) para una institución de salud local.
- Obtención y presentación adecuada de resultados.

2. Metodología

2.1. Metodología de trabajo y comunicación

La comunicación de manera virtual se mantuvo constante a lo largo del proyecto, por lo tanto, permitió sortear las barreras impuestas por un contexto de pandemia. Se mantuvieron reuniones periódicas para analizar el progreso, futuras tareas a realizar y para la resolución de problemas. La frecuencia de los encuentros variaba conforme la complejidad del proyecto avanzaba.

El Ing. Srecko Mileta actuó como nexo entre el hospital y la facultad proporcionando los datos necesarios para el progreso del trabajo.

2.2. Elección y desarrollo del software

A continuación se detallarán las distintas alternativas elegidas de software para las etapas del proceso de KDD (preparación de los datos, minería de datos y evaluación, interpretación y visualización de resultados) y restantes fases del proyecto.

2.2.1. Etapas de limpieza, selección y transformación

Existen varias opciones para llevar a cabo las primeras fases del KDD. Entre ellas, se encuentran algunas muy conocidas en el ámbito de la minería de datos, tales como Weka [33], RapidMiner [35] o KNIME [34]. Sin embargo, el software optado para esta etapa fue **Orange Data Mining** [3].

Orange es una plataforma gratuita de código abierto desarrollada en la Universidad de Liubliana utilizada principalmente en tareas de visualización, exploración, minería y análisis

de datos. Está desarrollada en gran parte en los lenguajes C, C + + y Python y se encuentra disponible para los sistemas operativos Windows, Linux y MacOS.

Figura 1

Interfaz de Orange con ejemplo de workflow para el procesamiento de un archivo .csv

El software se destaca principalmente por ser muy intuitivo de usar. Se trata de una herramienta de programación visual, en la que se construye un flujo de trabajo (*workflow*) arrastrando y soltando componentes predefinidos llamados *widgets* (ver **Figura 1**). Estos representan distintas técnicas, herramientas y algoritmos conocidos y utilizados en el ámbito.

Además de los *widgets* que vienen por defecto, se pueden descargar paquetes que contienen nuevos componentes orientados a áreas específicas como la Bioinformática o la minería de textos. Si una técnica o algoritmo no se encuentra disponible, es posible

implementarla mediante un componente particular que permite incluir scripts desarrollados en Python.

Orange tiene un sitio propio en donde se pueden encontrar, además del software, documentación, tutoriales, plantillas de *workflows* ya creadas , blogs, soporte y muchas otras utilidades [3].

La razón de su elección se debe principalmente a su alto grado de flexibilidad durante el tratamiento de los datos. Poder visualizar en tiempo real el flujo de datos, mientras se los manipula, facilitó en mayor medida su análisis. Además, es capaz de procesar el gran volumen de datos con los que se tuvo que trabajar, tarea que otros softwares eran incapaces de hacer por limitaciones de memoria. Finalmente, se tuvo preferencia por Orange por su compatibilidad con Python.

2.2.2. Etapa de minería de datos

En primer lugar, se utilizó la librería "**kmodes**" [4], implementada en **Python**, para la ejecución del algoritmo de k-prototypes. La misma es de libre acceso con licencia MIT y ofrece implementaciones tanto de k-prototypes como de k-modes. Además, está correctamente documentada y cuenta con varias opciones de parametrización de los algoritmos. Este paquete es de público conocimiento y ha sido utilizado anteriormente en otros proyectos e investigaciones tal como *Unsupervised learning with mixed type data for detecting money laundering* [5].

En segundo lugar, para la elección del entorno en dónde se ejecutaría el SOM, se realizaron experimentaciones con software y librerías de Java, Python y MATLAB© [53]. Finalmente, se terminó optando por la librería de uso libre **SOMToolbox** [6, 7] implementada en **MATLAB©** para el entrenamiento de los mapas. Esto es porque esta herramienta cuenta

con las opciones necesarias para tener mapas de mejor calidad. Más adelante (a partir de la **Sección 4.2.9.**) se detallarán los motivos de esta elección.

SOMToolbox es un paquete de uso libre de funciones para MATLAB© que implementa el algoritmo self-organizing map. Fue desarrollado en la Universidad de Helsinki, mismo lugar donde el profesor Teuvo Kohonen presentó por primera vez los mapas auto-organizados. SOMToolbox permite entrenar un SOM con diferentes topologías de red y parámetros de aprendizaje. Además, se pueden calcular para el mapa distintos tipos de errores, calidad y medidas. Por último, cuenta con varias opciones de visualización muy útiles a la hora de realizar análisis..

2.2.3. Etapa de análisis y gráfico de resultados

Para la confección de los distintos gráficos sobre los resultados obtenidos en la fase de *data mining* se usó la librería **plotly** de **Python** debido a su fácil aplicación, flexibilidad y gran documentación disponible en su sitio web [8].

La biblioteca es de código abierto y cuenta con una amplia variedad de gráficos interactivos con calidad de publicación; desde gráficos básicos, estadísticos, financieros y científicos hasta mapas, diagramas para bioingeniería e inteligencia artificial. Además, es muy versátil y permite personalizar ampliamente sus componentes.

Los gráficos correspondientes a las representaciones del SOM y sus variantes se realizaron en **SOMToolbox** de **MATLAB©**.

2.2.4. Construcción del software prototipo

El software propio para el análisis de los resultados se desarrolló en **MATLAB©** porque los cálculos y algoritmos necesarios ya habían sido implementados con la librería **SOMToolbox**.

2.2.5. Documentación

Se utilizó **Google Docs** [54] al ser este gratuito y de fácil utilización. Además, permite editar documentos de forma simultánea, por lo que se pudo trabajar sobre un archivo en conjunto.

2.2.6. Repositorio y control de versiones

Como repositorio para documentos y recursos generados se hizo uso de **Google Drive** [55].

También, se utilizó **Git** [9] mediante un repositorio en **Github** [56], para llevar un control de versiones sobre el software desarrollado en Python en la etapa de *data mining*.

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar distintos tipos de proyectos de una manera eficiente. Como ventajas de su utilización se pueden mencionar:

- Realización de comparaciones entre las distintas versiones del programa.
- Contar con una copia del código fuente en cada una de las versiones para poder volver a una anterior en caso de algún error o imprevisto en la versión actual.
- Tener un historial detallado de todas las modificaciones realizadas a lo largo del tiempo y ver quién fue el responsable de cada una al mismo tiempo que se puede apreciar toda la evolución del software.

3. Marco teórico

3.1. Knowledge discovery in databases (KDD)

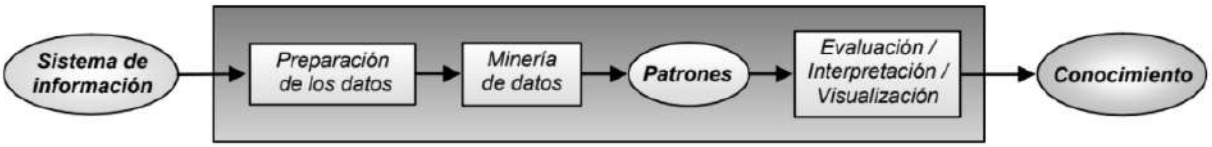
3.1.1. Definición

Se puede definir al KDD como un "proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos" [58]. Este proceso se ilustra en la **Figura 2**.

- **Válido:** hace referencia a que los patrones deben seguir siendo precisos para datos nuevos.
- **Novedoso:** que aporte algo desconocido tanto para el sistema y preferiblemente para el usuario.
- **Potencialmente útil:** la información debe conducir a acciones que reporten algún tipo de beneficio para el usuario.
- **Comprensible:** la extracción de patrones no comprensibles dificulta o imposibilita su interpretación, revisión, validación y uso en la toma de decisiones.

Figura 2

Etapas del proceso de KDD



Nota. Tomado de *Introducción a la Minería de Datos* (p. 13), por J. Hernandéz Orallo, M. J. Ramírez Quintana y C. Ferris Ramírez, 2004, Pearson Prentice Hall [2].

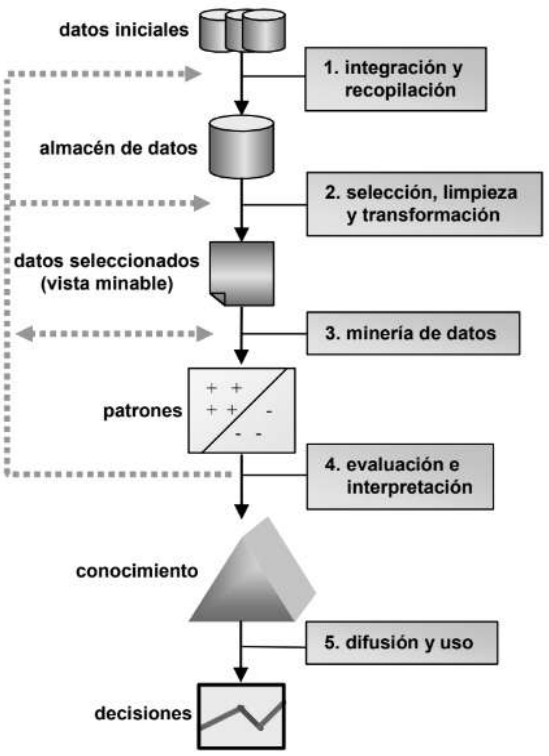
La diferencia entre KDD y la minería de datos es que el primero es el proceso global de descubrir conocimiento útil desde las bases de datos mientras que la minería de datos se refiere a la aplicación de los métodos de aprendizaje y estadísticos para la obtención de patrones y modelos.

3.1.2. Fases de KDD

En la **Figura 3** se muestra que KDD es un proceso iterativo e interactivo. Es iterativo ya que la salida de alguna de las fases puede hacer volver a pasos anteriores y porque generalmente son necesarias varias iteraciones para extraer conocimiento de alta calidad. Es interactivo porque el usuario, o un experto del dominio, debe ayudar en la preparación de los datos, validación del conocimiento extraído, etc.

Figura 3

Fases del proceso de descubrimiento de conocimiento en bases de datos, KDD.



Nota. Tomado de *Introducción a la Minería de Datos* (p. 20), por J. Hernandéz Orallo, M. J. Ramírez Quintana y C. Ferris Ramírez, 2004, Pearson Prentice Hall [2].

1- **Integración y recopilación:** lo normal es que los datos necesarios para poder llevar a cabo un proceso de KDD pertenezcan a diferentes lugares. Lo primero en realizarse, por lo tanto, es la integración de todos estos datos, lo que también da lugar a *data warehousing*.

2- **Limpieza y transformación y selección:** constituyen lo que se conoce como vista minable (subconjunto de datos a minar). Esta fase es necesaria, ya que existen problemas que afectan a la calidad de los datos y es necesario erradicarlos:

   - Datos irrelevantes o innecesarios.
   - Datos anómalos, pueden representar errores en los datos o pueden ser valores correctos que son simplemente diferentes a los demás. No siempre conviene eliminarlos.
   - Datos faltantes o perdidos.

   Otra tarea de preparación de los datos es la construcción de atributos, la cual consiste en construir automáticamente nuevos atributos aplicando alguna operación o función a los atributos originales. El objetivo de estos nuevos atributos es hacer más fácil el proceso de minería.

3- **Fase de minería de datos:** es la más característica del KDD y tiene como objetivo principal construir un modelo que genere nuevo conocimiento para el usuario. Se deben tomar las siguientes decisiones:

   1) Determinar qué tipo de tarea de minería es el más apropiado (ejemplo: clasificación).
   2) Elegir el tipo de método (ejemplo: RNA).

3) Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que se está buscando.

4- **Fase de evaluación e interpretación:** en esta etapa principalmente se analiza y aplica las medidas de calidad adecuadas para los modelos y técnicas utilizados.

5- **Fase de difusión, uso y monitorización:** el modelo puede usarse principalmente con dos finalidades: para que un analista recomiende acciones basándose en el modelo y en sus resultados o bien para aplicar el modelo a diferentes conjuntos de datos. Es necesario su difusión y medir lo bien que el modelo evoluciona.

3.1.3. Tareas de la minería de datos

Dentro de la minería de datos existen distintos tipos de tareas y cada una puede considerarse como un tipo de problema a ser resuelto por un algoritmo de minería de datos.

Las tareas pueden ser predictivas o descriptivas. Entre las tareas predictivas encontramos la clasificación y la regresión, mientras que el agrupamiento (*clustering*), las reglas de asociación, las reglas de asociación secuenciales y las correlaciones son tareas descriptivas [2]. La tarea de interés para el trabajo es el *clustering*, que se describe a continuación.

3.2. *Clustering*

El objetivo de esta tarea es obtener grupos o conjuntos entre los elementos de entrada de tal manera que los elementos asignados al mismo grupo sean similares. A priori, no se sabe ni cómo son los grupos ni cuántos hay. En algunos casos se puede proporcionar el número de grupos que se desea obtener. Otras veces este número se determina por el algoritmo de agrupamiento según las características de los datos.

Averiguar que las instancias se agrupan en varios segmentos es útil porque se puede determinar el comportamiento de una nueva viendo a qué grupo pertenece. Generalmente, posterior al agrupamiento, se toman decisiones diferentes para cada grupo.

Se pueden definir variantes para realizar un agrupamiento suave u obtener estimadores de probabilidad de agrupamiento que proporcionan más flexibilidad y posibilidades a la hora de interpretar y trabajar con los grupos formados.

Otro posible uso es utilizar el agrupamiento para reducir la gran cantidad de ejemplos en unos pocos grupos para posteriormente analizarlos y así entender mejor los datos originales, utilizando dichos grupos a modo de resumen. Esto también se conoce como sumarización [2].

3.3. Técnicas de *clustering*

3.3.1. K-means

3.3.1.1. Definición y ventajas

El algoritmo k-means [37, 38, 39, 40] es el método de agrupamiento en *clusters* no jerárquico más ampliamente utilizado que busca minimizar la distancia cuadrática promedio entre puntos en el mismo grupo. Aunque no ofrece garantías de precisión, su sencillez y rapidez resultan muy atractivas en la práctica. Se puede decir que es el algoritmo de agrupación en *clusters* más popular utilizado en aplicaciones científicas e industriales [2].

K-means es versátil, fácil de modificar en cada etapa del proceso y simple en la función de cálculo de distancia.

Se mencionó que k-means es un método no jerárquico, esto es porque es necesario a priori especificar el número de *clusters*. En contraparte, los métodos jerárquicos van

generando grupos a medida que las fases del proceso elegido avanzan para encontrar así el número de grupos óptimo.

3.3.1.2. Algoritmo

La idea del algoritmo es clasificar un conjunto de datos dado en *k* números de clases disjuntas y consta de dos fases.

La primera fase consiste en definir *k* centroides, uno para cada grupo establecido.

La siguiente fase es tomar cada punto de datos del conjunto de datos con el que se está trabajando y asociarlo al centroide más cercano. Se considera generalmente la distancia euclidiana para determinar la distancia entre los datos y los centroides. Cuando todos los datos son incluidos en algún grupo, se completa el primer paso y se realiza la agrupación temprana. En este momento se recalculan los nuevos centroides, ya que la inclusión de nuevos puntos de datos puede conducir a un cambio en los centros del grupo. Una vez que se encuentran los nuevos centroides, los puntos de datos se reasignan a su centro más cercano. Nuevamente, se recalculan los centroides y es así cómo se genera un ciclo. Eventualmente, llegará el punto en donde los centroides ya no se moverán. Esto es lo que se conoce como el criterio de convergencia del algoritmo para la agrupación [10].

A continuación se detalla, con sus respectivas fórmulas matemáticas [2], el algoritmo:

1) Se calcula para cada punto de dato x_p presentado, el centroide más próximo A_g y se incluye en la lista de datos de dicho centroide como se observa en la **Ecuación 1**.

$$A_g = arg\ min_{A_i} \{ d(x_p, A_i) \}\ \forall i = 1..k \qquad \textbf{(Ec. 1)}$$

2) Después de haber introducido todos los datos, cada centroide A_i tendrá un conjunto de *m* datos x_i a los que representa ilustrado en la **Ecuación 2**.

$$I(A_i) = \{ x_{i_1} , x_{i_2} , x_{i_3} , \dots , x_{i_m} \} \qquad \textbf{(Ec. 2)}$$

3) Se desplaza el centroide A_i hacia el centro de masas de su conjunto de *m* datos x_i como se indica en la **Ecuación 3**.

$$A_i = \frac{\sum_{j=1}^{m} x_{i_j}}{m} \qquad \textbf{(Ec. 3)}$$

4) Se repite el procedimiento hasta que ya no se desplazan los centroides.

Mediante este algoritmo el espacio de datos de entrada se divide en *k* clases o regiones y el centroide de cada clase estará en el centro de la misma. Dichos centros se determinan con el objetivo de minimizar las distancias cuadráticas euclídeas entre los datos de entrada y el centro más cercano, es decir minimizando el valor *J* como se muestra en la **Ecuación 4**.

$$J = \sum_{i=1}^{k} \sum_{p=1}^{n} M_{i,p} \, d_{EUCL} \, (x_p - A_i)^2 \qquad \textbf{(Ec. 4)}$$

Siendo:

- *n* el conjunto total de datos
- d_{EUCL} la distancia euclidiana
- x_p el dato de entrada *p*

- A_i el centroide de la clase *i*

- $M_{i,p}$ es la función de pertenencia del dato p de la región i de forma que vale 1 si el centroide A_i es el más cercano al dato x_p y 0 en caso contrario. Esto se muestra en la **Ecuación 5**:

$$M_{i,p} = \begin{cases} 1 \ si \ d_{EUCL}(x_p - A_i) < d_{EUCL}(x_p - A_s) \ \forall s \neq i, s = 1, 2, \dots, k \\ 0 \ en \ caso \ contrario \end{cases} \quad \textbf{(Ec. 5)}$$

3.3.1.3. Consideraciones

Que el algoritmo requiera con anterioridad el número de clases es una de las cuestiones más importantes a tener en cuenta, ya que implica una de las principales debilidades de k-means. Por ejemplo, si suponemos que existen aparentemente cinco grupos y se elige un k = 4 al menos dos grupos terminarían unidos. No obstante, si se incluyen demasiados *clusters* el resultado obtenido puede no ser mejor, ya que esto generaría grupos que serían divididos de forma artificial e ineficiente.

Existe otra desventaja y es que el algoritmo produce diferentes grupos para diferentes valores de centroides iniciales. La calidad y precisión de los conglomerados finales depende en gran medida de la inicialización de estos centroides. Además, en término de debilidades de k-means se puede destacar la complejidad computacional que presenta debido a la necesidad de reasignar los datos varias veces, durante cada iteración del ciclo.

Por último, una limitación clave es su modelo de agrupamiento. El concepto se basa en grupos esféricos que son separables de una forma en que el valor de la media converge hacia el centro del grupo. Se espera que los grupos tengan igual tamaño. Esto se puede demostrar en el siguiente ejemplo desarrollado en Orange:

Se utilizó el Mouse Dataset. Este *dataset* de 500 puntos de datos cuenta con 3 tipos de etiquetas correspondientes a la cabeza, oreja derecha y oreja izquierda de un ratón (**Figura 4**). Además, tiene un etiquetado adicional que se corresponde con ruido.

Al conjunto de datos se le aplicó k-means con un $k = 3$ y se lo inicializó usando k-means + + (método que se detalla en la **Sección 4.1.2.**). El resultado es el siguiente:

Figura 4

Visualización del dataset Mouse utilizando Scatter Plot

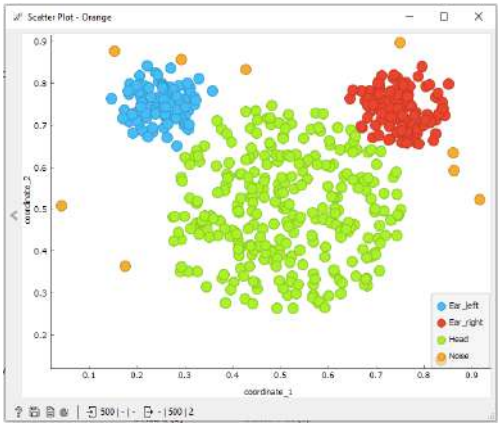
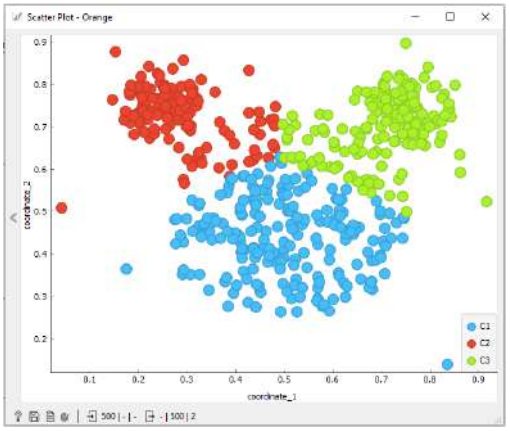


Figura 5

Visualización, con Scatter Plot, del resultado de procesar el Mouse Dataset usando k-means



Se puede evidenciar con esto cómo esta tendencia a formar grupos de similar tamaño llevan al método a realizar un mal etiquetado en ciertos puntos de datos (ver **Figura 5**).

Como sucede con cualquier otro algoritmo de *clustering*, el resultado de k-means no solo depende de los parámetros de entrada, sino también del *dataset*. Esto quiere decir que simplemente trabaja bien en algunos conjuntos de datos mientras que en otros no realiza una agrupación de calidad.

3.3.2. K-Prototypes

3.3.2.1. Definición

K-prototypes surge como una variante del algoritmo k-means para tratar el inconveniente de no poder trabajar con datos no numéricos. Si bien existe la alternativa de

realizar una conversión de datos categóricos a numéricos, esta opción implica normalmente un incremento notable de la cantidad de atributos. Esto impacta directamente sobre cuestiones tales como el almacenamiento requerido, problemas derivados de una alta dimensionalidad y la claridad y comprensión de la información, entre otras. Además, esta conversión "forzada" puede no reflejar distancias reales entre los distintos valores por lo que el modelo resultante puede ser impreciso [12].

3.3.2.2. Cambios en el algoritmo frente a k-means

La base del algoritmo se mantiene casi idéntica a la del k-means, siendo la diferencia más notable el cambio en la medida de distancia utilizada. Esta nueva medida de disimilitud entre dos elementos se obtiene sumando dos partes, correspondientes a la distancia numérica y categórica entre los mismos. Esto puede verse en la **Ecuación 6**:

$$d(X_i, Q_l) = \sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c) \qquad \textbf{(Ec. 6)}$$

Siendo:

- $d(X_i, Q_l)$ la distancia entre un objeto X_i y su prototipo correspondiente Q_l con $1 \leq i \leq n$ y $1 \leq l \leq m$, siendo n la cantidad de objetos o casos y m la cantidad de prototipos (centroides)
- m_r la cantidad de atributos numéricos
- m_c la cantidad de atributos categóricos
- γ el factor de ponderación *gamma*
- δ la medida de disimilitud categórica

La parte de la distancia numérica se obtiene sumando las distancias individuales de cada atributo numérico mediante el cálculo de la distancia euclidiana al cuadrado. La otra parte se calcula sumando las disimilitudes entre los atributos categóricos. Para esto, el algoritmo utiliza una medida de disimilitud denominada *simple matching dissimilarity* [57]. Básicamente, lo que esta métrica refleja es la cantidad de no coincidencias entre los atributos categóricos de dos objetos distintos. Si dos categorías coinciden en un mismo valor, la disimilitud entre ambas tomará el valor cero y de ocurrir lo contrario tomará el valor uno (**Ecuación 7**).

$$\delta(x_j, y_j) = \begin{cases} 0 \ (x_j = y_j) \\ 1 \ (x_j \neq y_j) \end{cases} \qquad \textbf{(Ec. 7)}$$

El término correspondiente a la disimilitud categórica también se ve afectado por un parámetro llamado *gamma* (γ). Este coeficiente actúa como un factor de ponderación y sirve para favorecer (o no) atributos de un cierto tipo, por lo que su valor cambiará según se tengan más o menos atributos categóricos o si se les quiere dar más o menos importancia a unos que a otros. Si este valor fuese cero, la técnica sería equivalente al k-means clásico (no habría peso en las componentes categóricas). El autor de este algoritmo sugiere utilizar valores pequeños de *gamma* en general (suelen utilizarse valores comprendidos entre 0 y 2 aunque podría darse el caso de necesitar valores más grandes).

A la hora de definir el objeto más representativo de cada *cluster* (es decir, su centroide) nuevamente se recurre a dos procedimientos distintos según se trate de atributos numéricos o categóricos. Para los primeros, al igual que el k-means clásico, se utiliza la media para determinar el valor representativo de cada atributo. En el caso de los categóricos, se obtiene la moda de cada uno. En definitiva, cada representante de clase será un vector de atributos tanto numéricos como categóricos [12].

3.3.3. Self-organizing maps (SOM)

3.3.3.1. Definición

Los mapas auto-organizados son un tipo de red neuronal artificial que se entrenan utilizando técnicas de aprendizaje no supervisado competitivo. Como resultado de este entrenamiento se obtiene una representación discreta de baja dimensión del espacio de las muestras de entrada, llamado mapa. Fueron presentados por el profesor finlandés Teuvo Kohonen en 1982 [13] .

Existen muchas evidencias empíricas de la utilidad de los mapas auto-organizados, y sus resultados se han utilizado en una amplia gama de disciplinas: modelado neurobiológico, economía, sociología, minería de textos, monitoreo de procesos, entre otras. Además, esta técnica tiene futuro en el contexto de *big data*, puesto que su complejidad computacional es baja en proporción a la cantidad de datos procesados [52].

Estas redes neuronales artificiales al utilizar el aprendizaje no supervisado no requieren que se les proporcione un conjunto de muestras de entrada con la correspondiente etiqueta (clase) a la que pertenecen. Es suficiente con facilitarles un grupo de datos de entrada sin respuestas para ser capaces de hallar patrones ocultos en ellos.

Como ya se mencionó, difieren de otras redes neuronales artificiales, ya que aplican el aprendizaje competitivo en oposición al aprendizaje de corrección de errores. Cuando se presenta a la red un dato de entrada se pretende que una neurona se active. Por lo tanto, las neuronas compiten entre sí por activarse para quedar finalmente como neurona ganadora mientras que el resto son forzadas a sus valores de respuesta mínimos.

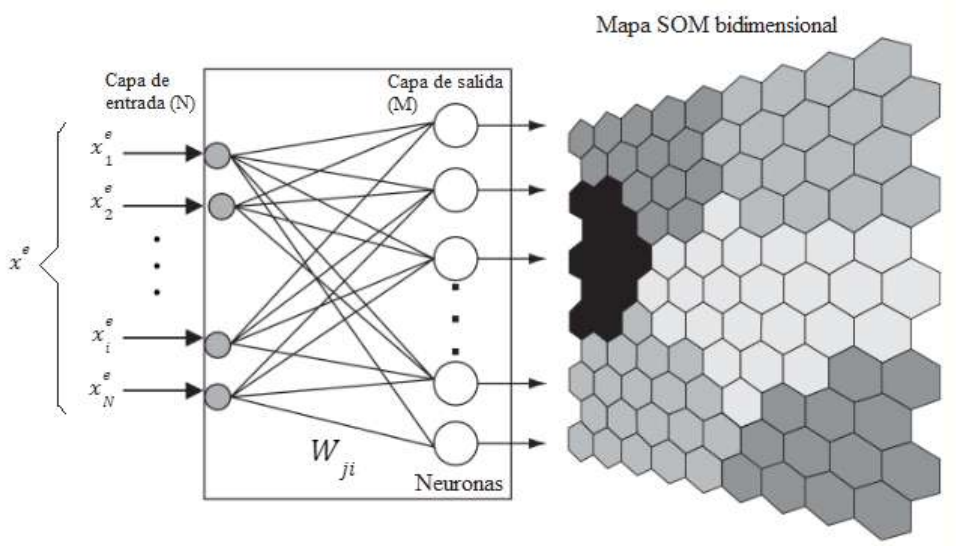
El objetivo de este aprendizaje es categorizar los datos que se introducen en la red. Se clasifican valores similares en la misma categoría y, por lo tanto, deben activar la misma

neurona de salida. Las clases o categorías son creadas por la propia red a través de las similitudes entre los datos de entrada.

En cuanto a la arquitectura, el modelo SOM está compuesto por dos capas de neuronas, como se muestra en la **Figura 6**:

Figura 6

Arquitectura de SOM



Nota. Inspirado de *TeX-LateX Stack Exchange*, 2017, https://tex.stackexchange.com/questions/144366/draw-a-kohonen-som-feature-map

- **La capa de entrada:** formada por *N* neuronas, una por cada atributo de la entrada. Encargada de recibir y transmitir a la capa de salida la información del exterior.
- **La capa de salida:** formada por *M* neuronas. Encargada de procesar la información y formar el mapa.

El algoritmo de entrenamiento de los mapas autoorganizados está compuesto por tres partes [14]:

- Competencia entre neuronas.
- Determinación de las neuronas que pertenecen a la vecindad de la neurona ganadora.

- Adaptación de los pesos.

3.3.3.2. Competencia entre neuronas

A cada neurona de la capa de salida *j* se le asigna un vector de pesos con la misma dimensionalidad que el espacio de entrada.

$$W_j = (W_{j1}, \dots, W_{ji}, \dots, W_{jN})$$

Se calculan las distancias entre cada neurona de la capa de salida con las neuronas de entrada. La neurona de salida con la menor distancia será la neurona ganadora *c*. A esta neurona se la denomina BMU (*Best Matching Unit*), como se muestra en la **Ecuación 8**.

$$A_c = min_j \{ | x^e - W_j | \} \qquad \textbf{(Ec. 8)}$$

Siendo $x^e = (x_1^e, \dots, x_i^e, \dots, x_N^e)$ una muestra de la base de entrenamiento X^e.

Normalmente para calcular la distancia entre las neuronas se utiliza la distancia euclidiana (ver **Ecuación 9**).

$$d^2(x^e, W_j) = \sum_{i=1}^{n} (x_i^e - W_{j_i})^2 \qquad \textbf{(Ec. 9)}$$

3.3.3.3. Determinación de las neuronas pertenecientes a la vecindad

Antes de actualizar los pesos se determina qué neuronas pertenecen a la vecindad de la neurona ganadora. Para esto se siguen los siguientes pasos:

1) Se calcula el tamaño del radio de vecindad centrado en la neurona ganadora.

2) Se determina qué neuronas están dentro de dicho radio.

Para el paso número uno se utiliza la función de decaimiento exponencial (**Ecuación 10**) que calcula para cada iteración $t = t_0, t_1, \dots, t_T$ la dimensión del radio de vecindad (ver **Figura 7**) :

$$\sigma(t) = \sigma_0 e^{-(\frac{t}{\tau})} \qquad \textbf{(Ec. 10)}$$

donde σ_0 corresponde al radio de vecindad para la iteración t_0 , siendo las *filas* y *columnas* la cantidad de filas y columnas pertenecientes a la grilla del mapa (**Ecuación 11**):

$$\sigma_0 = \frac{max\,(filas, columnas)}{2} \qquad \textbf{(Ec. 11)}$$

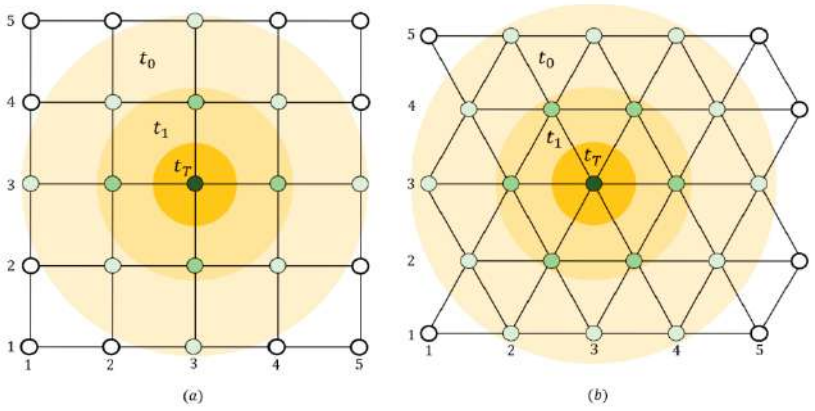
Y donde τ es una constante denominada vida media que se define en la **Ecuación 12** como:

$$\tau = \frac{T}{log\,\sigma_0} \qquad \textbf{(Ec. 12)}$$

siendo *T* el número total de iteraciones que realiza el algoritmo de entrenamiento.

Figura 7

Actualización del radio de vecindad para una topología rectangular (a) y hexagonal (b) suponiendo que la neurona del centro es la neurona ganadora (BMU).



Nota. Tomado de *Mapas Autoorganizados para Clasificación en Tiempo Real: Implementación digital sobre FPGAs* (p. 27), por M. D. Rodríguez, 2020.

Una vez obtenido el radio de vecindad para cada iteración se determina qué neuronas se encuentran dentro del radio (paso dos). Para eso se calcula la distancia entre la neurona ganadora y todas las neuronas de la red y se comprueba si están dentro del radio de vecindad. Esto se muestra en la **Ecuación 13**:

$$d^2(j,c) = \sum_{i=1}^{N} (j_i, c_i)^2 = \begin{cases} \leq \sigma(t) & j \in N_c(t) \\ > \sigma(t) & j \notin N_c(t) \end{cases} \qquad \textbf{(Ec. 13)}$$

Siendo:

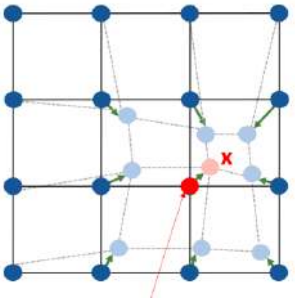
- j la neurona de salida, donde $j \neq c$
- c la neurona ganadora
- $N_c(t)$ neuronas que pertenecen a la vecindad de la neurona ganadora

3.3.3.4. Adaptación de pesos

Una vez elegida la neurona ganadora c y las neuronas pertenecientes a su vecindad $N_c(t)$, se deben actualizar los pesos para que las neuronas se acerquen a la observación de entrada x^e , aunque no de la misma manera (ver **Figura 8**). Cuanto más alejadas estén las neuronas de las muestras de entrada x^e, el ajuste de los pesos será menor.

Figura 8

Actualización de los pesos de las neuronas de la capa de salida para una topología rectangular



Nota. Tomado de *Mapas Autoorganizados para Clasificación en Tiempo Real: Implementación digital sobre FPGAs* (p. 28), por M. D. Rodríguez, 2020.

Los pesos de la neurona ganadora y la de sus vecinas se actualizan mediante la siguiente fórmula (ver **Ecuación 14**):

$$W_{ji}(t+1) = W_{ji}(t) + \alpha(t)\, h_{jc}(t)\, (x_i^e - W_{ji}(t)),\ j \in N_c(t) \quad \textbf{(Ec. 14)}$$

donde $\alpha(t)$ es la tasa de aprendizaje y se define en la **Ecuación 15** y puede verse su comportamiento en la **Figura 9**:

$$\alpha(t) = \alpha_0\, e^{(-\frac{t}{T})} \quad \textbf{(Ec. 15)}$$

siendo α_0 la tasa de aprendizaje inicial y donde $h_{jc}(t)$ es la función de núcleo de vecindad definida en la **Ecuación (16)**:

$$h_{jc}(t) = exp(\frac{d_{jc}^2}{2\sigma^2(t)}) \quad \textbf{(Ec. 16)}$$

Esta función tiene como objetivo hacer que el efecto del aprendizaje sea proporcional a la distancia entre la neurona ganadora y las neuronas pertenecientes a su vecindad.

Cabe destacar que en este caso la actualización de los pesos de las neuronas es realizada de forma secuencial, es decir, luego de procesar cada dato de entrada individualmente. Existe otra variante, llamada *Batch SOM* (SOM por lotes), en donde los pesos se actualizan una vez procesados todos los datos de entrada [13]. El SOM por lotes es utilizado por otra variante de SOM detallada más adelante en la **Sección 4.1.3.1**.

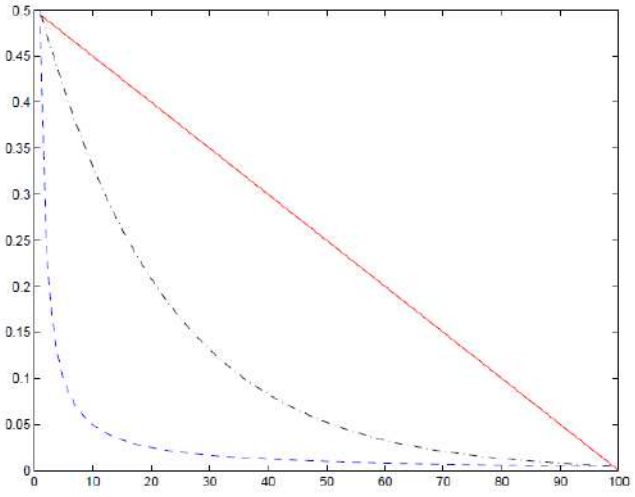
3.3.3.5. Funciones de velocidad de aprendizaje

Uno de los factores a tener en cuenta a la hora de entrenar un mapa auto-organizado es la definición de su función de tasa o velocidad de aprendizaje. La **Ecuación 15**, de carácter

exponencial, fue una de las más comunes que se encontró en la bibliografía consultada. Sin embargo, no es el único tipo de función de aprendizaje que existe. Por ejemplo, en la **Figura 9** se puede apreciar el comportamiento de otros tipos de funciones:

Figura 9

Distintos tipos de funciones de aprendizaje: lineal (color rojo), exponencial (color negro) e inversamente proporcional a la cantidad de iteraciones (color azul)



Nota. Tomado de *SOM Toolbox for Matlab 5,* (p.10), 2000 [15].

Las funciones de aprendizaje recién mencionadas se definen en general de la siguiente manera [15], en las **Ecuaciones 17, 18** y **19**:

1. Lineal: $\alpha(t,T) = \alpha_0.(1 - \frac{t}{T})$ **(Ec. 17)**

2. Exponencial: $\alpha(t,T) = \alpha_0.e^{\frac{-t}{T}}$ **(Ec. 18)**

3. Inversamente proporcional: $\alpha(t) = \alpha_0.\frac{1}{t}$ **(Ec. 19)**

También se utilizó otra función que consiste en una forma particular de la función de velocidad de aprendizaje de tipo exponencial [16] (**Ecuación 20**):

$$\alpha(t) = \mu.\alpha(t-1) \quad \text{para } t > 0 \qquad \textbf{(Ec. 20)}$$

En donde μ es el factor de variación de la velocidad de aprendizaje durante el entrenamiento, y toma valores entre 0 y l. El valor de μ se calcula en función del total de iteraciones y del valor de velocidad inicial (α_0) y velocidad final (α_f) deseados para la primera y última época (ver **Ecuaciones 21** y **22**):

$$\alpha_f = \mu^T . \alpha_0 \quad \textbf{(Ec. 21)}$$

$$\mu = e^{ln(\alpha_f/\alpha_0)/T} \quad \textbf{(Ec. 22)}$$

El radio de la vecindad es definido de forma análoga y se calcula en cada iteración a partir de los valores de la distancia inicial (σ_0), final (σ_f) y la cantidad total de iteraciones.

Habiendo definido los parámetros de entrada y las funciones de tasa de aprendizaje y vecindad, se presenta el siguiente pseudocódigo del algoritmo de SOM utilizado:

Entrada: conjunto de datos $X = \{x_1, \dots, x_N\}$
Salida: conjunto de neuronas $Y = \{y_1, \dots, y_M\}$
Inicio
    inicializar Y de manera aleatoria
    repetir
        seleccionar $x \in X$ aleatóriamente
        calcular $y \in Y$ tal que $BMU(x) = y$
        actualizar todos los pesos de Y en función de x, y, $\alpha_{(t)}$ y $h_{(t)}$
        actualizar valores de $\alpha_{(t)}$ y $\sigma_{(t)}$
    hasta llegar al número de iteraciones fijado
Fin

3.3.3.6. Representación del SOM: matriz-U

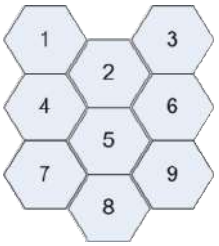
Una vez entrenado el mapa, es necesaria una herramienta para poder observar los posibles clusters o clases formadas. La manera más común de realizar esto es creando una matriz llamada matriz-U (matriz de distancias unificadas) [17].

Por cada neurona de salida j, se calcula la distancia entre ella y sus neuronas vecinas más próximas. El resultado es una matriz que indica cuán cerca está cada neurona con sus vecinas más próximas.

Para ilustrar lo anterior se propone el siguiente ejemplo. Suponiendo que se tiene una grilla de 9 neuronas (ver **Figura 10**).

Figura 10

Grilla de SOM de 3x3

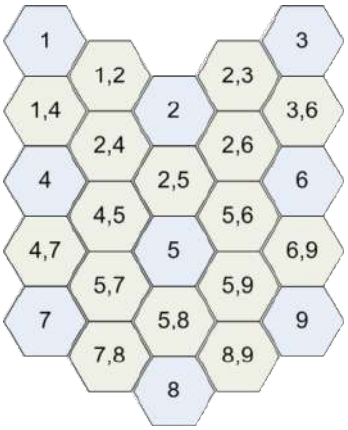


Nota. Tomado de StackOverflow, 2012, https://stackoverflow.com/questions/13631673/how-do-i-make-a-u-matrix

La matriz-U resultante de esta grilla es la de la **Figura 11**:

Figura 11

Matriz-U de SOM de 3x3



Nota. Tomado de StackOverflow, 2012, https://stackoverflow.com/questions/13631673/how-do-i-make-a-u-matrix

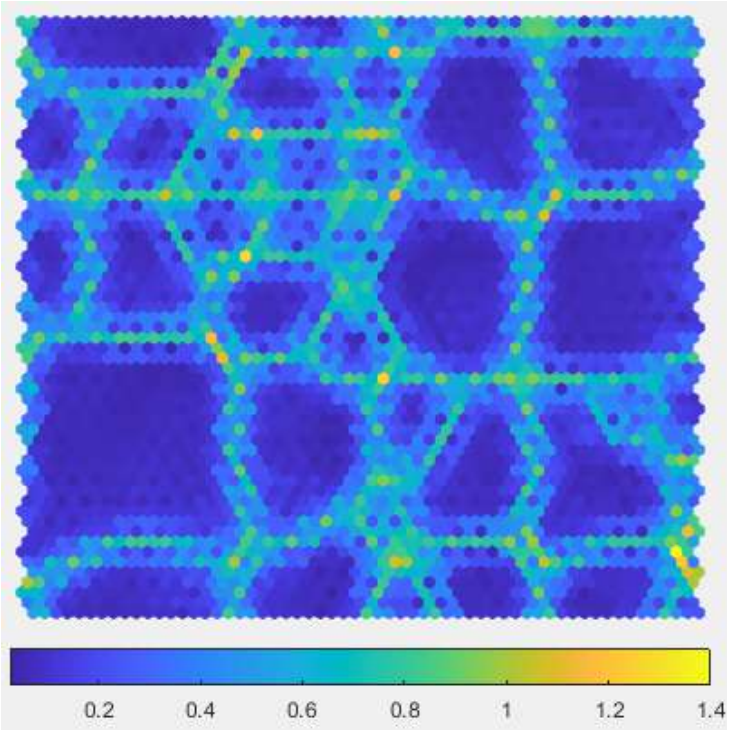
Esas nuevas celdas que se agregan representan la distancia entre una neurona y otra según corresponda. Por ejemplo: la "1,4" representa la distancia entre la neurona 1 y la 4.

Luego de que se calculan estas distancias, la matriz-U se puede representar gráficamente coloreando las celdas según su valor.

A continuación (**Figura 12**) se muestra un ejemplo de la representación gráfica de una matriz-U para una grilla de 30x30:

Figura 12

Matriz de distancias unificadas para SOM de 30x30



En la barra inferior se muestra que para las distancias más grandes las celdas se colorean en amarillo y en caso contrario de azul.

Esta técnica permite visualizar agrupaciones (los espacios azules que se forman) y sus fronteras (límites que se forman por las celdas coloreadas de verde y amarillo).

3.3.3.7. K-means sobre SOM

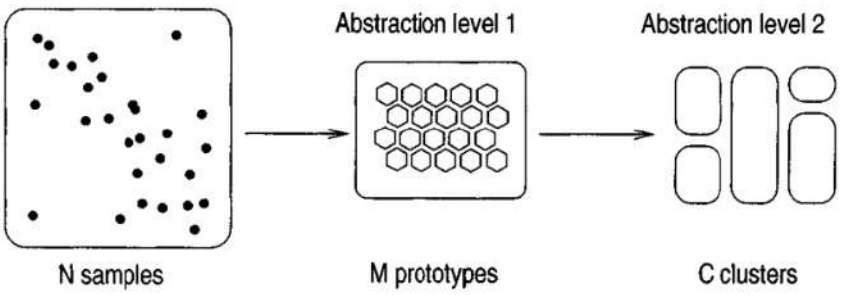
Los mapas auto-organizados son redes neuronales eficaces en la visualización de datos de alta dimensión. Sin embargo, para poder comprender completamente el contenido de un conjunto de datos es vital averiguar si los datos tienen estructura de *cluster*. Si este es el

caso, los grupos se extraen para poder explorar completamente las propiedades del *dataset*. Esto facilita el análisis en el caso de mapas compuestos por numerosas neuronas.

El enfoque de dos niveles trata de la aplicación de un método de *clustering* sobre SOM en lugar de realizarlo directamente en los datos [11]. Este proceso está ilustrado en la **Figura 13** y es el siguiente: primero, se obtiene un conjunto de neuronas a partir del procesamiento de los datos con los mapas autoorganizados. Estas neuronas, o prototipos, se pueden interpretar como "protoclusters" los cuales en el siguiente paso serán combinados para formar los grupos finales. Cada punto de datos del *dataset* original pertenece al mismo *cluster* que su neurona más cercana, por lo tanto y ya teniendo los grupos finales, se procesa cada punto de datos, se lo asocia a su BMU correspondiente y se lo etiqueta con el mismo *cluster* al que pertenece dicho BMU. Es así como el *dataset* original queda clasificado de manera indirecta a través del *clustering* de las neuronas del SOM previamente entrenado.

Figura 13

Proceso de obtención de clusters utilizando el enfoque de dos niveles. El primer nivel de abstracción se obtiene creando un conjunto de neuronas usando, por ejemplo, SOM. El clustering sobre el SOM crea el segundo nivel de abstracción.



Nota. Tomado de *IEEE Transactions on Neural Networks*: *Clustering of the Self-Organizing Map* (p. 586), por Juha Vesanto y Esa Alhoniemi, 2000, IEEE, Vol. 11, No. 3 [11].

La aplicación de este enfoque presenta ventajas. Primero, el conjunto de datos original se representa usando un conjunto más pequeño de vectores prototipo, lo que permite un uso eficiente de algoritmos de agrupamiento para dividir los prototipos/neuronas en grupos. Esto implica una reducción del costo computacional, especialmente importante para algoritmos

jerárquicos. En segundo lugar, la grilla 2D permite una presentación e interpretación visual aproximada de los *clusters*. Otro beneficio es la reducción de ruido. Los prototipos son promedios locales de los datos y, por lo tanto, son menos sensibles a las variaciones aleatorias que los datos originales [11].

Siguiendo estos conceptos y teniendo en cuenta las virtudes que presenta, en el presente trabajo se ha aplicado el algoritmo k-means sobre un SOM ya entrenado.

3.4. Métricas de evaluación

A continuación se profundizará sobre cada una de las medidas utilizadas para evaluar los resultados obtenidos de las técnicas de *clustering*. Las primeras tres se aplican a k-means (y también k-prototypes en el caso de silhouette score) mientras que las dos últimas se usan para analizar SOMs.

3.4.1. Elbow method

El método del codo es una heurística utilizada para determinar el número óptimo de *clusters* presentes en un *dataset* [18]. Esto se logra graficando la suma de los cuadrados intra-cluster (WCSS) de todos los puntos una vez entrenado el modelo haciendo variar el valor de k (cantidad deseada de *clusters*) hasta un valor máximo razonable. El WCSS es una medida de la varianza del agrupamiento obtenido y se calcula como se indica en la **Ecuación 23**:

$$WCSS(k) = \sum_{j=1}^{k} \sum_{x_i \in cluster\ j} \left|\left| x_i - \overline{x}_j \right|\right|^2 \qquad \textbf{(Ec. 23)}$$

Siendo:

- k la cantidad total de *clusters*

- x_i un objeto perteneciente al *cluster j*

- $\bar{x}_j$ el centroide correspondiente al *cluster j*

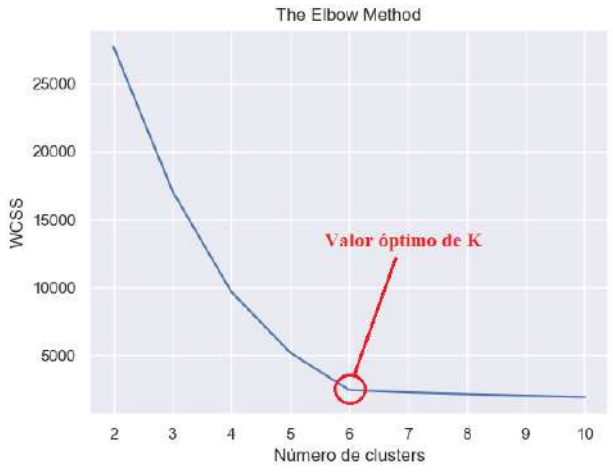
- $\left\|x_i - \bar{x}_j\right\|^2$ la distancia euclidiana al cuadrado entre x_i y $\bar{x}_j$

Una vez realizado el gráfico se identifica el "codo" de la curva (de ahí el nombre de la técnica). Este punto indica que el modelo está en su valor óptimo de ajuste (ver **Figura 14**). Esto significa que no se obtendrá un mejor modelado si se sigue aumentando el número de *clusters*. Los valores inferiores a este punto indican un infra-ajuste del modelo (*underfitting*), mientras que los superiores muestran un sobreajuste del mismo (*overfitting*).

Es común que esta técnica se utilice en conjunto con algoritmos de *clustering* que necesiten el número total de *clusters* como un parámetro de entrada y se desconozca su posible valor como es el caso de k-means y k-prototypes.

Figura 14

Valor óptimo de k para el dataset 10 Clusters utilizando el elbow method



Es importante destacar que este método es de carácter visual y que no siempre se obtendrán gráficos en donde el punto de quiebre se encuentre fácilmente marcado. Aún así, la

técnica puede servir para indicar posibles valores de k que sean buenos para un conjunto de datos dado.

3.4.2. Silhouette score

Silhouette score [19] es una medida que representa qué tan bien fue agrupado un objeto. Se calcula utilizando la distancia media del objeto con los elementos de su mismo *cluster* ($a(i)$) y la distancia media al *cluster* más próximo ($b(i)$) (ver **Figura 15 y Ecuaciones 24, 25** y **26**).

$$s(i) = \frac{b(i) - a(i)}{max\{a(i)\,, b(i)\}} \qquad \textbf{(Ec. 24)}$$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i,\, i \neq j} d(i,j) \qquad \textbf{(Ec. 25)}$$

$$b(i) = min_{k \neq i} \left(\frac{1}{|C_k|} \sum_{j \in C_k} d(i,j) \right) \qquad \textbf{(Ec. 26)}$$

Siendo:

- $s(i)$ el valor de Silhouette Score para el objeto i
- $a(i)$ la distancia media del objeto i con los elementos de su mismo *cluster*
- $b(i)$ la distancia media del objeto i con el *cluster* más próximo
- C_i el *cluster* correspondiente al objeto i
- C_k el *cluster* correspondiente al objeto j, con $k \neq i$
- $d(i,j)$ la distancia entre el objeto i y el objeto j

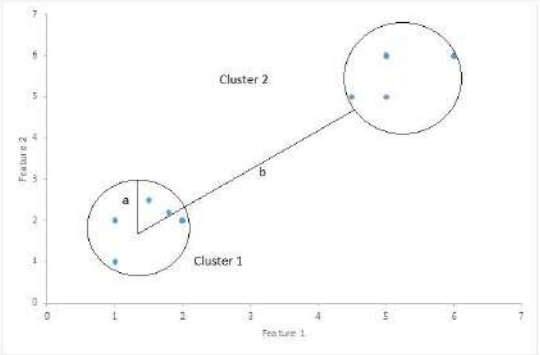
Esta medida toma valores contínuos dentro del rango $[-1, 1]$. Los valores extremos y central tienen la siguiente interpretación:

- El valor **1** indica que el objeto fue agrupado de manera perfecta.
- El valor **-1**, por el contrario, indica que la agrupación fue totalmente errónea. Esto significa que el objeto en cuestión debería haber sido agrupado en un *cluster* diferente.
- El valor **0** indica solapamiento. Un ejemplo de esto sería un objeto que se encuentra en el medio de dos grupos.

Silhouette score puede ser calculada ya sea para un objeto individual como para un *cluster* específico o para el modelo entero.

Figura 15

Ilustración de las distancias utilizadas para el cálculo de silhouette score



Nota. Tomado de *Coeficiente de silueta: Validación de técnicas de agrupamiento,* por Ashutosh Bhardwaj, 2020, https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c

Esta métrica presenta las siguientes ventajas:

- Puede ser aplicada independientemente de la medida de distancia utilizada, mediante el cálculo previo de una matriz de distancias.

- Puede ser calculada ya sea para un objeto individual como para un *cluster* específico o para el modelo entero.
- Tiene en cuenta tanto la cohesión (aglutinamiento) de los *clusters* como la separación entre los mismos para su cálculo.
- Puede ser representada gráficamente (silhouette plot) para una mejor interpretación de ser necesario.

Como desventaja se puede mencionar que el cálculo de esta medida no es muy escalable, puesto que debe calcularse una matriz de nxn, siendo n la cantidad total de observaciones. Sin embargo, si se guarda la matriz para reutilizarse los tiempos pueden acortarse de manera significativa.

3.4.3. Davies-Bouldin index

El índice de Davies-Bouldin [20] es un número adimensional que indica la similitud promedio entre los *clusters*, siendo esta similitud una medida que compara la distancia entre los *clusters* con su tamaño. Cuanto más bajo sea el valor del índice, mayor separación tendrán los grupos que lo componen. El valor mínimo que esta métrica puede tomar es cero.

El índice DB se calcula como se muestra en las **Ecuaciones 27** y **28**:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \qquad \textbf{(Ec. 27)}$$

$$DB = \frac{1}{k}\sum_{i=1}^{k} max_{i \neq j}\ R_{ij} \qquad \textbf{(Ec. 28)}$$

Siendo:

- R_{ij} la similitud entre los *clusters* i y j

- s_i la distancia promedio entre cada objeto perteneciente al *cluster* i y el centroide de dicho *cluster* (a esto se lo conoce también como diámetro del *cluster*)
- d_{ij} la distancia entre los centroides del *cluster* i y el *cluster* j
- k el número total de *clusters*

Esta métrica tiene la ventaja de ser mucho más simple y rápida de calcular que silhouette score pero tiene también la desventaja de ser aplicable sólo cuando la distancia utilizada es euclidiana.

3.4.4. Error de cuantización

Para que un mapa auto-organizado sea un modelo preciso debe "ajustarse" a los datos y al mismo tiempo preservar la topología y las vecindades de los datos de entrada. La primera condición es medida por el error de cuantización (QE). Esta métrica dimensiona la distancia promedio entre los puntos de datos y sus correspondientes BMUs. Kohonen la sugiere como una medida básica de calidad para evaluar los mapas autoorganizados [13].

Valores pequeños de QE indican un mejor resultado siendo cero el menor valor posible. El error de cuantización se calcula de la siguiente manera (**Ecuación 29**) [21]:

$$QE(M) = \frac{1}{n}\sum_{i=1}^{n}\left|\left|\phi(x_i) - x_i\right|\right| \qquad \textbf{(Ec. 29)}$$

Siendo:

- $QE(M)$ el error de cuantización correspondiente al mapa M
- n la cantidad de objetos

- $\phi(x_i)$ el peso de la BMU del dato de entrada x_i
- $\left|\left|\phi(x_i) - x_i\right|\right|$ la distancia euclidiana entre $\phi(x_i)$ y x_i

Es importante destacar que el error de cuantización solamente se puede usar para comparar mapas entre sí, no como una evaluación independiente de calidad. Además, debe utilizarse únicamente para comparar mapas del mismo tamaño. Esto se debe a que el valor de QE es proporcional al tamaño de la grilla del SOM. Un mapa que tenga mayor cantidad de neuronas resultará en un valor de QE más pequeño y viceversa.

Esta métrica solo tiene en cuenta la relación entre los objetos o datos de entrada y las neuronas que les son asignados, no cómo se organizan las neuronas entre sí. Esto significa que el QE solo evalúa la estructura local de los datos, es decir, no detecta problemas en el mapa como giros o pliegues en el espacio.

3.4.5. Error topográfico

Como se mencionó anteriormente, uno de los objetivos principales del algoritmo SOM es preservar las características topológicas del espacio de entrada en el espacio de salida de baja dimensión. El error topográfico (TE) es una medida de qué tan bien la estructura del espacio de entrada es modelada por el mapa [21].

El TE se calcula encontrando la neurona de mejor coincidencia y la segunda de mejor coincidencia en el mapa para cada punto de entrada y luego se evalúan sus posiciones. Si los nodos son adyacentes, se dice que la topología se ha conservado para esta entrada. Si esto no ocurre, se cuenta como un error. El número total de errores dividido por el número total de puntos de datos da como resultado el error topográfico del mapa (**Ecuación 30**).

$$TE(M) = \frac{1}{n}\sum_{i=1}^{n} t(x_i) \qquad \textbf{(Ec. 30)}$$

Siendo:

- $TE(M)$ el error topográfico correspondiente al mapa M
- n la cantidad de objetos
- x_i el objeto i
- $t(x) = \begin{cases} 0 \; si \; \mu(x) \; y \; \mu'(x) \; son \; adyacentes \\ 1 \; en \; caso \; contrario \end{cases}$
- $\mu(x)$ la BMU y $\mu'(x)$ la segunda BMU de x

El error topográfico es una medida que sólo evalúa qué tan bien se mapean los puntos de datos individuales a los nodos y no tiene en cuenta, al igual que el error de cuantización, problemas con el mapa como giros, pliegues, etc.

3.5. Numerización de atributos

La numerización es el proceso inverso a la discretización (transformación de un atributo categórico a uno numérico). Suele ser de mucha utilidad cuando el método de *data mining* que se va a utilizar no admite datos nominales.

Un enfoque común que suele utilizarse es lo que se denomina numerización "1 a n", que es la creación de varias variables indicadoras bandera o *flag*. Si una variable nominal x tiene posibles valores $\{a_1, a_2, ..., a_n\}$ se crean n variables numéricas, con valores 0 o 1 dependiendo de si la variable nominal toma ese valor o no. Este proceso también recibe el nombre de binarización.

Una evidente consecuencia de aplicar esta técnica es el aumento notorio de la dimensionalidad del juego de datos, dado que esta se incrementará en uno por cada valor posible que el atributo categórico pueda tomar [2].

3.6. Normalización de atributos

La normalización de atributos es muy frecuente en la mayoría de los algoritmos basados en distancias o cuando es necesario realizar escalados no lineales por la presencia de datos anómalos.

La normalización más común es la normalización lineal uniforme y se normaliza a una escala genérica entre cero y uno utilizando la siguiente fórmula (**Ecuación 31**).

$$v' = \frac{v - min}{max - min} \qquad \textbf{(Ec. 31)}$$

Siendo:

- v' el valor escalado del atributo para un objeto
- v el valor original del atributo para un objeto
- min el valor mínimo registrado del atributo para todos los objetos
- max el valor máximo registrado del atributo para todos los objetos

El resultado de esta normalización es que la relación (el cociente) entre valores se mantiene. Para realizar esto sólo es necesario conocer el máximo y mínimo de los valores dados para ese atributo.

Se ha recurrido a este método de normalización puesto que las variables categóricas se ven forzadas a pertenecer al conjunto $\{0, 1\}$ luego de aplicarles binarización. Esto restringe el conjunto de valores que puedan tomar los atributos numéricos, ya que si éstos pertenecieran a un rango distinto al $[0, 1]$ los cálculos basados en distancias se verían perjudicados [2].

4. Trabajo realizado

En este capítulo se detalla en orden cronológico las etapas que constituyen el desarrollo del trabajo. Se documenta la investigación, selección, pruebas e implementación de las técnicas junto con los resultados.

4.1. Etapas de investigación, análisis y selección

4.1.1. Variantes de k-means

K-means es utilizado para datos de tipo numérico y si bien se lo puede forzar para ser usado con datos categóricos, por ejemplo a través de la binarización, la precisión del agrupamiento resulta afectada. Es por eso que existen variaciones pensadas para diferentes tipos de datos. Una de ellas es **k-modes** [12] para tratar con datos categóricos. En lugar de la distancia euclidiana utiliza una medida de disimilitud de coincidencia simple, conocida como distancia de Hamming (abarcada también en la **Sección 3.3.2.2.**).

Otra de las variantes, previamente ya detallada, es **k-prototypes** [12] que mediante la definición de una nueva medida de distancia, integra los algoritmos k-means y k-modes para permitir la agrupación de datos mixtos. Diversos estudios han demostrado que ambos algoritmos son eficientes al agrupar grandes conjuntos de datos, lo cual es fundamental para las aplicaciones de minería de datos.

K-means es un algoritmo de agrupamiento duro, es decir, no difuso. Algo significativo sobre este tipo de algoritmos es el defecto que surge del modelo en sí en el que cada dato está agrupado inequívocamente con otros miembros de "su" grupo y sólo pertenece a él. Por lo tanto, no tiene similitud aparente con otros miembros del conjunto de datos. De esto surge una variante de k-means conocida como **fuzzy c-means clustering** [22]. La idea de fuzzy c-means *clustering* es representar la similitud que un dato comparte con cada grupo

con una función, denominada función de membresía, cuyos valores, denominados membresías, están entre cero y uno. En este algoritmo, cada dato tendrá una membresía en cada grupo, las membresías cercanas a la unidad significan un alto grado de similitud entre el dato y un grupo, mientras que las membresías cercanas a cero implican poca similitud entre el dato y ese grupo. El efecto neto de dicha función para la agrupación es producir particiones *C* difusas de un conjunto de datos dado. En resumen, una partición *C* difusa de un *dataset* es aquella que caracteriza la pertenencia de cada dato en todos los conglomerados mediante una función de pertenencia. Además, la suma de los miembros para cada dato debe ser la unidad.

Con el mismo razonamiento, existen las variantes **fuzzy k-modes** [23] para datos categóricos y **fuzzy k-prototypes** [24] para datos de tipo numérico y categórico mixtos.

Otra de las variaciones es **k-means ++** [25], pero esta es utilizada en combinación con k-means para la inicialización de los centroides. K-means + + es un método utilizado como selección de semillas iniciales para intentar resolver las agrupaciones pobres que pueden resultar luego de aplicar k-means a un conjunto de datos. Lo que intenta este método es minimizar la varianza intra-grupo, es decir, minimizar la suma de las distancias al cuadrado de cada punto de dato al centroide más cercano a él. Aunque utilizar k-means + + para la selección de los centroides iniciales tome tiempo extra, k-means converge muy rápidamente y se reduce, por lo tanto, el tiempo de cálculo.

Además, existen variantes como **k-medoids** [26] que, en contraparte a k-means, elige puntos reales como centros (medoides) y se puede usar con medidas de disimilitud arbitrarias. También, **k-medians** [27] que en lugar de calcular la media de cada grupo para determinar su centroide calcula la mediana, **spherical k-means** [28] que difiere de la versión estándar al normalizar los centroides al final de cada paso de maximización. Otras como **x-means** [29], método de estimación del número de conglomerados que entra en acción después de cada ejecución de k-means, tomando decisiones locales sobre qué subconjunto de los centroides

actuales deben dividirse para lograr un mejor ajuste. Por último, **g-means** [30] es otro k-means extendido que intenta determinar automáticamente el número de *clusters* mediante una prueba de normalidad. Se basa en una prueba estadística para la hipótesis de que un subconjunto de datos sigue una distribución gaussiana. Ejecuta k-means aumentando *k* de forma jerárquica hasta que la prueba acepta la hipótesis de que los datos asignados a cada centro de k-means son gaussianos**.**

Teniendo en cuenta factores como la naturaleza de los datos a utilizar, tiempo requerido para entender el lenguaje de programación, cantidad de información disponible y la posibilidad de acceder a la implementación del algoritmo a través de algún software o librería, se optó por utilizar **k-prototypes**.

4.1.3. Variantes de self-organizing map

4.1.3.1. FMSOM

El Mapa Autoorganizado Mixto de Neuronas de Frecuencia, **FMSOM** [31], crea un nuevo modelo para tratar las características categóricas, lo que mejora considerablemente el rendimiento. Preserva el algoritmo original para tratar la parte numérica, mientras que para los datos categóricos se amplían los vectores de peso de las neuronas con un vector de frecuencia, uno por característica categórica. Sin entrar en tanto detalle, FMSOM está compuesto por tres procesos al igual que en el SOM original (tanto por lote como secuencial): competitivo, cooperativo y adaptativo. El cooperativo es similar al SOM por lotes, en cambio, las diferencias están en los otros dos procesos. En el proceso competitivo, la medida de disimilitud para los atributos numéricos (D_n) es la misma que en el SOM original, es decir (**Ecuación 31**):

$$D_n(X_p, W_i) = \sqrt{\sum_{z=1}^{n} (X_{pz} - W_i)^2} \quad \textbf{(Ec. 31)}$$

siendo:

- X_p un vector de entrada
- W_i una neurona ganadora
- *P* cantidad de vectores de entrada, con $1 \leq p \leq P$
- *I* cantidad de neuronas del mapa, con $1 \leq i \leq I$
- *n* cantidad de atributos numéricos

En cambio, para los atributos de tipo categóricos la disimilitud se calcula en base a medidas de probabilidad, como se ve en la **Ecuación 32**:

$$D_c(X_p, W_i) = \sum_{z=n+1}^{k} (1 - W_{iz}[X_{pz}])^2 \quad \textbf{(Ec. 32)}$$

siendo *k* la cantidad de atributos categóricos. Los primeros *n* atributos del vector corresponden a atributos numéricos, mientras que los *k* restantes corresponden a los categóricos.

La ecuación anterior nos dice que la disimilitud categórica entre un vector de entrada X_p y una neurona W_i es igual a la sumatoria de las disimilitudes parciales para cada atributo categórico. Esta disimilitud parcial se obtiene calculando la probabilidad de que la neurona no contenga el valor presente del atributo categórico del vector de entrada ($1 - W_{iz}[X_{pz}]$).

Finalmente, la medida de disimilitud total es la siguiente (**Ecuación 33**):

$$d(X_p, W_i) = D_n(X_p, W_i) + D_c(X_p, W_i) \qquad \textbf{(Ec. 33)}$$

En cuanto al proceso adaptativo ya se pueden ver algunas diferencias observando el vector W_i en la ecuación de disimilitud de los atributos categóricos. Las componentes del vector de peso de las neuronas W_i son de dos tipos: para un atributo numérico la componente es de tipo numérico y para un atributo categórico la componente del vector es otro vector pero de frecuencias, llamado vector de frecuencias relativas.

Para que sea más fácil de visualizar la idea y a muy grades rasgos, se propone el siguiente ejemplo donde se tiene la estructura de un vector de pesos W para los atributos "edad", "altura", "sexo" y "color de ojos" como carecterísticas de una persona:

$$W = [\underset{\textit{valor numérico}}{\textbf{edad}}, \underset{\textit{valor numérico}}{\textbf{altura}}, \underbrace{[\underset{\textit{valor masculino}}{\textit{frecuencia relativa}}, \underset{\textit{valor femenino}}{\textit{frecuencia relativa}}]}_{\textbf{sexo}}, \underbrace{[\underset{\textit{valor café}}{\textit{frecuencia relativa}}, \underset{\textit{valor azul}}{\textit{frecuencia relativa}}, \underset{\textit{valor verde}}{\textit{frecuencia relativa}}]}_{\textbf{color\_de\_ojos}}]$$

Como se puede observar, la cantidad de componentes del vector de frecuencias relativas corresponde a la cantidad de valores posibles que puede tomar el atributo categórico, es decir; $\alpha_k^1, \alpha_k^2, \ldots, \alpha_k^r$ siendo $\alpha^1, \alpha^2, \ldots, \alpha^r$ cada uno de los r valores posibles que puede tomar el atributo categórico k. Además, el valor de cada una de estas componentes, $F(\alpha_k^r, W_{ik}(s))$, se calcula como se muestra en la **Ecuación 34**:

$$F(\alpha_k^r, W_{ik}(s)) = \frac{\sum_{p=1}^{P} h(s, c(X_p), i) \mid X_{pk} = \alpha^r}{\sum_{p=1}^{P} h(s, c(X_p), i)} \qquad \textbf{(Ec. 34)}$$

numerador: vecindad acumulada para el atributo categórico k con valor α^r

denominador: vecindad acumulada total

siendo

- $c(X_p)$ la neurona ganadora para el vector de entrada p.

- *s* la iteración actual.
- *h* función de vecindad gaussiana.
- $h\ (s,\ c\ (X_p),\ i)$ el valor en la iteración *s* de la función vecindad *h* que está centrada en la neurona ganadora $c\ (X_p)$ cuando el vector de entrada X_p es presentado a la red.

Por lo tanto, durante el proceso de adaptación, el vector de peso de la neurona *i* en la iteración *s+1* se define como la combinación de los vectores de peso de los atributos numéricos y categóricos (**Ecuación 35**):

$$W_i(s\ +\ 1)\ =\ \{\ W_{in}(s\ +\ 1),\ W_{ik}(s\ +\ 1)\ \},\ n\ =\ 1\ldots N,\ k\ =\ 1\ldots K$$

(Ec. 35)

siendo $W_{in}\ (s\ +\ 1)$ la misma ecuación adaptativa para atributos numéricos que se usa en el algoritmo de SOM por lotes y $W_{ik}\ (s\ +\ 1)$ siendo como se muestra en la **Ecuación 36**:

$$W_{ik}(s\ +\ 1)\ =\ \{F(\alpha_k^1,\ W_{ik}(s)),\ F(\alpha_k^2,\ W_{ik}(s)),\ \ldots,\ F(\alpha_k^r,\ W_{ik}(s))\}$$

(Ec. 36)

El principal problema con este tipo de SOM para variables mixtas es que no se especifica el cálculo de la matriz-U. Por otra parte, tampoco existe documentación que sustente la convergencia del método. Teniendo en cuenta esto y sumado a la complejidad que conlleva implementar dicho algoritmo con la posibilidad de no poder visualizar las distancias entre neuronas, la variante fue descartada.

4.1.3.2. GSOM

El Generalized Self-Organizing Map [32] sirve para mejorar la habilidad del SOM para procesar datos categóricos y datos mixtos de forma directa y reflejar fielmente su relación topológica.

Se utiliza el concepto de distancia jerárquica, la cual está compuesta por nodos y enlaces donde los nodos de nivel superior representan conceptos más generales mientras que los nodos de nivel inferior representan conceptos más específicos (ver **Figura 17**). Además, cada enlace tiene un peso que representa la distancia (el peso puede ser asignado por un experto). Por lo tanto, la distancia entre dos conceptos en los nodos hoja se define, entonces, como el peso total del enlace entre esos dos nodos hoja.

A continuación se muestran ejemplos de binarización (**Figura 16**) y de conceptos de jerarquía (**Figuras 17 y 18**).

Figura 16

El atributo categórico Favorite_Drink se transforma en cuatro atributos binarios según su dominio

Name	Favorite_Drink	Amount
Jane	Coke	7
Mary	Pepsi	7
Tom	Mocca	7
Helen	Nescafe	7

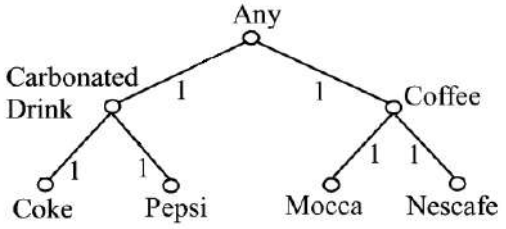
⇨

Name	Coke	Pepsi	Mocca	Nescafe	Amount
Jane	1	0	0	0	7
Mary	0	1	0	0	7
Tom	0	0	1	0	7
Helen	0	0	0	1	7

Nota. Tomado de *IEEE Transactions On Neural Networks,* Vol. 17, Nro. 2 (p.296), 2006.

Figura 17

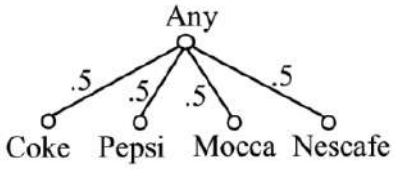
Distancia jerárquica con peso 1 en los enlaces



Nota. Tomado de *IEEE Transactions On Neural Networks,* Vol. 17, Nro. 2 (p.296), 2006.

Figura 18

Jerarquía de distancia de dos niveles para un enfoque de coincidencia simple



Nota. Tomado de *IEEE Transactions On Neural Networks,* Vol. 17, Nro. 2 (p.296), 2006.

Se pueden mencionar algunas diferencias entre ambas metodologías:

1) En la binarización no se puede determinar la información de similitud entre los valores categóricos. Por ejemplo, la relación transformada de la **Figura 18** no muestra que Coca-Cola sea más parecida a Pepsi que a Moca.

2) Cuando el dominio de un atributo categórico es grande, transformarlo en un conjunto de atributos binarios aumenta la dimensionalidad de la relación, lo que resulta en el desperdicio de espacio de almacenamiento y el aumento del entrenamiento.

3) Es difícil mantener el nuevo esquema binarizado. Cuando cambia el dominio de un atributo, es necesario cambiar el esquema de relación transformado. Por ejemplo, si se agrega "Juice" al dominio de Favorite_Drink, se debe incluir un atributo adicional Juice en el esquema de relación transformado.

4) Los nuevos atributos binarios no pueden reflejar la semántica del atributo original. Por ejemplo, después de la transformación, los cuatro atributos binarios no expresan el significado de Favorite_Drink.

No es posible aplicar esta variante al *dataset* de COVID-19 porque no existe una relación jerárquica en los atributos. Teniendo en cuenta esto y que este enfoque no se encuentra implementado, se descartó.

Si bien las variantes del SOM presentan muchas ventajas por sobre la binarización, la dificultad de desarrollarlas y de testearlas excedería los tiempos y la complejidad del proyecto de grado propuesto. Por lo tanto, se hizo uso de la binarización para aplicar el SOM estándar al ser este el enfoque más utilizado en la bibliografía para datos mixtos.

4.1.4. Software, lenguajes de programación e IDE

La primera alternativa fue **Weka** [33], un programa de minería de datos escrito en Java y desarrollado en la Universidad de Waikato, Nueva Zelanda. Weka es software libre y fue utilizado en el pasado por uno de los directores del proyecto. Pese a que ofrece una interfaz interesante para el análisis de atributos, fue desestimado principalmente por las siguientes razones:

1) El software tiene muchas dificultades para procesar *datasets* de gran tamaño (del orden de los GBs), requisito indispensable dadas las dimensiones del juego de datos publicado por el Ministerio de Salud de Argentina.

2) La opciones de personalización de k-means y SOM no son lo suficientemente flexibles y no permiten la modificación de ciertos parámetros considerados necesarios.

Existen otras alternativas como **KNIME**, **RapidMiner** y **SAS** [34, 35, 36], softwares muy conocidos en el entorno de la minería de datos. Sin embargo, estas opciones no fueron seleccionadas por temas de licencia, limitaciones con la cantidad de datos que permitían analizar o cuestiones técnicas de los algoritmos que disponían.

Finalmente, se decidió utilizar **Orange Data Mining** [3] por todas las características detalladas en la sección de metodología. Aunque este software también tiene la desventaja de

no poder parametrizar lo suficiente las técnicas de *clustering* requeridas, sí es adecuado para el tratamiento de los datos en las fases previas al *data mining*.

La elección de un lenguaje de programación estuvo fuertemente ligada a las implementaciones existentes de las técnicas de *clustering* elegidas. Los lenguajes más usados en este contexto son **Python** y **R**. Se prefirió el primero puesto que su sintaxis y funcionamiento son similares a lenguajes anteriormente utilizados, haciendo que la curva de aprendizaje tenga menor impacto sobre el tiempo y esfuerzo requeridos.

Por último, se optó por **Visual Studio Code** como entorno de desarrollo porque cuenta con plugins para adaptarse al lenguaje de programación elegido. Además, tiene soporte para integración con Github y permite que dos o más personas puedan modificar un archivo de forma remota y en tiempo real.

4.1.5. Bases de datos públicas de interés

Dado que al comienzo del proyecto la pandemia era un hecho relativamente reciente, los *datasets* publicados no eran abundantes y los atributos que los componían no tenían una descripción muy detallada o no resultaban interesantes para el análisis.

El Ministerio de Salud de la Nación llevaba registro y publicaba a diario los casos de COVID-19 registrados día a día en el país [48]. Este *dataset*, pese a su gran tamaño, era de fácil acceso y de sencilla comprensión. Además, sus atributos tenían potencial para realizar un análisis interesante y poder hacer una comparación a futuro con datos provenientes del HPC. Por estas razones se lo eligió para comenzar con el proceso de KDD.

4.2. Aplicación KDD

Se trabajó sobre el juego de datos público "Casos COVID-19" proporcionado por el Ministerio de Salud. Dicho *dataset* lleva registro de los casos de COVID-19 notificados en todo el país y era actualizado de forma diaria. La extensión del archivo de datos es del tipo ".csv" (*comma separated values*, es decir, valores separados por coma), un tipo de archivo utilizado muy frecuentemente en el ámbito de la minería de datos, entre otros. Al momento de su análisis, el juego de datos contaba con más de 15 millones de casos registrados. La **Tabla 1** presenta la lista de atributos del *dataset* [48]:

Tabla 1

Descripción de atributos dataset COVID-19 publicado por el Ministerio de Salud

Atributo	Tipo de dato	Descripción
id_evento_caso	Número entero (*integer*)	Número de caso
sexo	Texto (*string*)	Sexo
edad	Número entero (*integer*)	Edad
edad_años_meses	Texto (*string*)	Edad indicada en meses o años
residencia_pais_nombre	Texto (*string*)	País de residencia
residencia_provincia_nombre	Texto (*string*)	Provincia de residencia
residencia_departamento_nombre	Texto (*string*)	Departamento de residencia
carga_provincia_nombre	Texto (*string*)	Provincia de establecimiento de carga
fecha_inicio_sintomas	Fecha ISO-8601 (*date*)	Fecha de inicio de síntomas
fecha_apertura	Fecha ISO-8601 (*date*)	Fecha de apertura del caso
sepi_apertura	Número entero (*integer*)	Semana Epidemiológica de fecha de apertura
fecha_internacion	Fecha ISO-8601	Fecha de internación

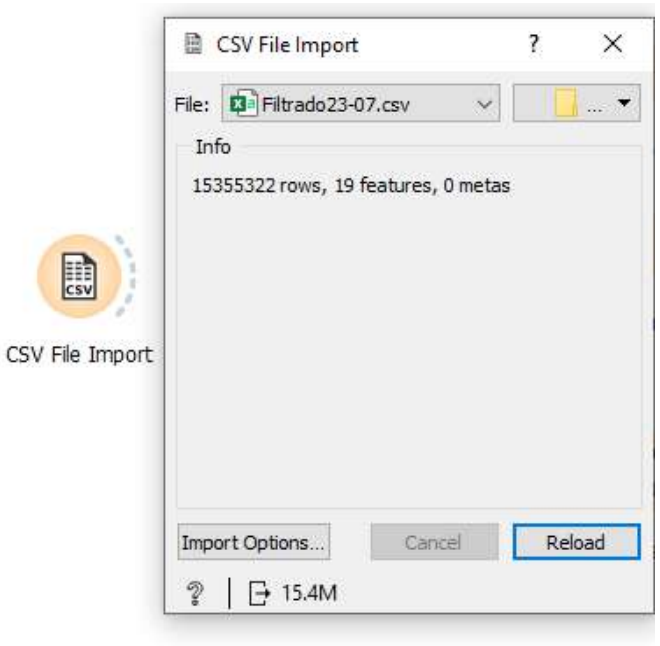
	(*date*)	
cuidado_intensivo	Texto (*string*)	Indicación si estuvo en cuidado intensivo
fecha_cui_intensivo	Fecha ISO-8601 (*date*)	Fecha de ingreso a cuidado intensivo en el caso de corresponder
fallecido	Texto (*string*)	Indicación de fallecido
fecha_fallecimiento	Tiempo ISO-8601 (*time*)	Fecha de fallecimiento en el caso de corresponder
asistencia_respiratoria_mecanica	Texto (*string*)	Indicación si requirió asistencia respiratoria mecánica
carga_provincia_id	Número entero (*integer*)	Código de Provincia de carga
origen_financiamiento	Texto (*string*)	Origen de financiamiento
clasificacion	Texto (*string*)	Clasificación manual del registro
clasificacion_resumen	Texto (*string*)	Clasificación del caso
residencia_provincia_id	Número entero (*integer*)	Código de Provincia de residencia
fecha_diagnostico	Tiempo ISO-8601 (*time*)	Fecha de diagnóstico
residencia_departamento_id	Número entero (*integer*)	Código de Departamento de residencia
ultima_actualizacion	Fecha ISO-8601 (*date*)	Última actualización

4.2.1. Exploración previa de los datos

Ya que los datos provienen de una única fuente no fue necesario realizar una integración previa de los mismos. Utilizando el *widget* llamado “CSV File Import” (ver **Figura 19**) se puede importar un archivo de extensión “.csv” y cargarlo en Orange.

Figura 19

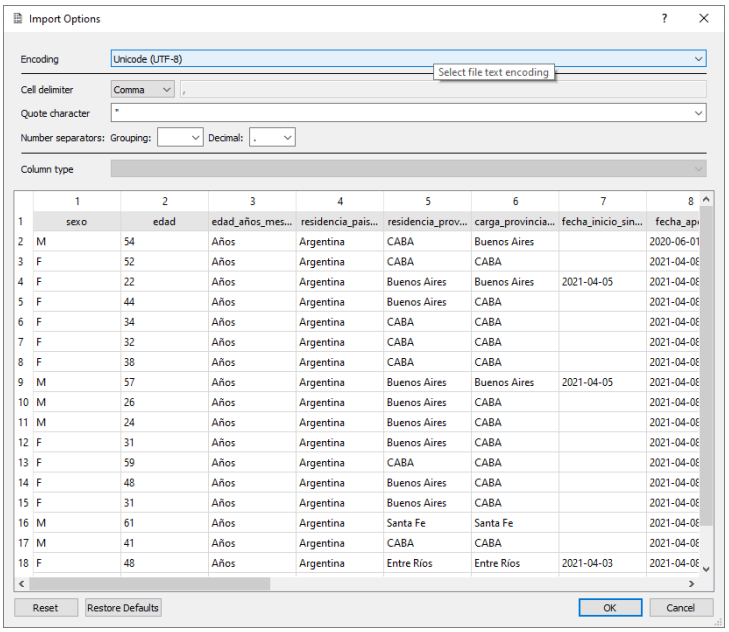
Carga de archivo .csv en Orange utilizando el widget CSV File Import



Al abrir este componente, aparece una ventana que permite cargar el archivo seleccionado. Una vez cargado, muestra la cantidad de filas y de atributos que lo componen. Además, pulsando el botón "Import Options" se puede tener una vista previa del archivo y realizar algunos ajustes si fuera necesario (ver **Figura 20**). Por ejemplo, se puede seleccionar el tipo de *encoding* que posee el archivo (útil cuando aparecen caracteres especiales, como lo son la letra "ñ" y las tildes), el caracter utilizado para delimitar campos (es muy común que, además de la coma, se utilice el tabulador), o de qué manera se indica el separador decimal en un número, entre otras opciones. Es posible además asignar manualmente a qué tipo de dato pertenece cada atributo (si no se hace, Orange detecta el tipo de forma automática). Los tipos de datos que maneja el programa internamente son: numérico, categórico, texto y fecha (tiempo). Por último, también permite ignorar columnas o filas que no se quieran propagar al flujo de trabajo, actuando como un primer filtro.

Figura 20

Visualización de widget Import Option de Orange para archivo .csv

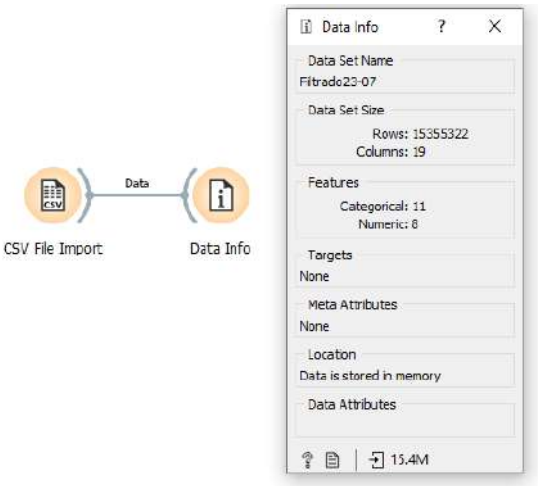


	1	2	3	4	5	6	7	8
1	sexo	edad	edad_años_mes...	residencia_pais...	residencia_prov...	carga_provincia...	fecha_inicio_sin...	fecha_ap
2	M	54	Años	Argentina	CABA	Buenos Aires		2020-06-01
3	F	52	Años	Argentina	CABA	CABA		2021-04-08
4	F	22	Años	Argentina	Buenos Aires	Buenos Aires	2021-04-05	2021-04-08
5	F	44	Años	Argentina	Buenos Aires	CABA		2021-04-08
6	F	34	Años	Argentina	CABA	CABA		2021-04-08
7	F	32	Años	Argentina	CABA	CABA		2021-04-08
8	F	38	Años	Argentina	CABA	CABA		2021-04-08
9	M	57	Años	Argentina	Buenos Aires	Buenos Aires	2021-04-05	2021-04-08
10	M	26	Años	Argentina	Buenos Aires	CABA		2021-04-08
11	M	24	Años	Argentina	Buenos Aires	CABA		2021-04-08
12	F	31	Años	Argentina	Buenos Aires	CABA		2021-04-08
13	F	59	Años	Argentina	CABA	CABA		2021-04-08
14	F	48	Años	Argentina	Buenos Aires	CABA		2021-04-08
15	F	31	Años	Argentina	Buenos Aires	CABA		2021-04-08
16	M	61	Años	Argentina	Santa Fe	Santa Fe		2021-04-08
17	M	41	Años	Argentina	CABA	CABA		2021-04-08
18	F	48	Años	Argentina	Entre Ríos	Entre Ríos	2021-04-03	2021-04-08

Una vez importados los datos, puede utilizarse el componente "Data Info" para obtener información en general sobre el conjunto de datos (ver **Figura 21**). Por ejemplo, puede verse la cantidad de filas y atributos, la cantidad de tipos de datos presentes, la presencia de meta atributos si los hubiere o la ubicación física de los datos (normalmente es en memoria).

Figura 21

Componente Data Info de Orange



Para poder comenzar con el proceso de limpieza, era necesario contar con información específica acerca de cada atributo presente en el *dataset*. El *widget* "Feature Statistics", ilustrado a continuación en las **Figuras 22** y **23**, permite visualizar datos estadísticos básicos de cada uno de ellos.

Figura 22

Workflow con el componente Feature Statistics de Orange

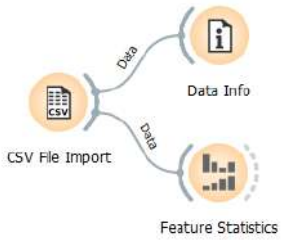
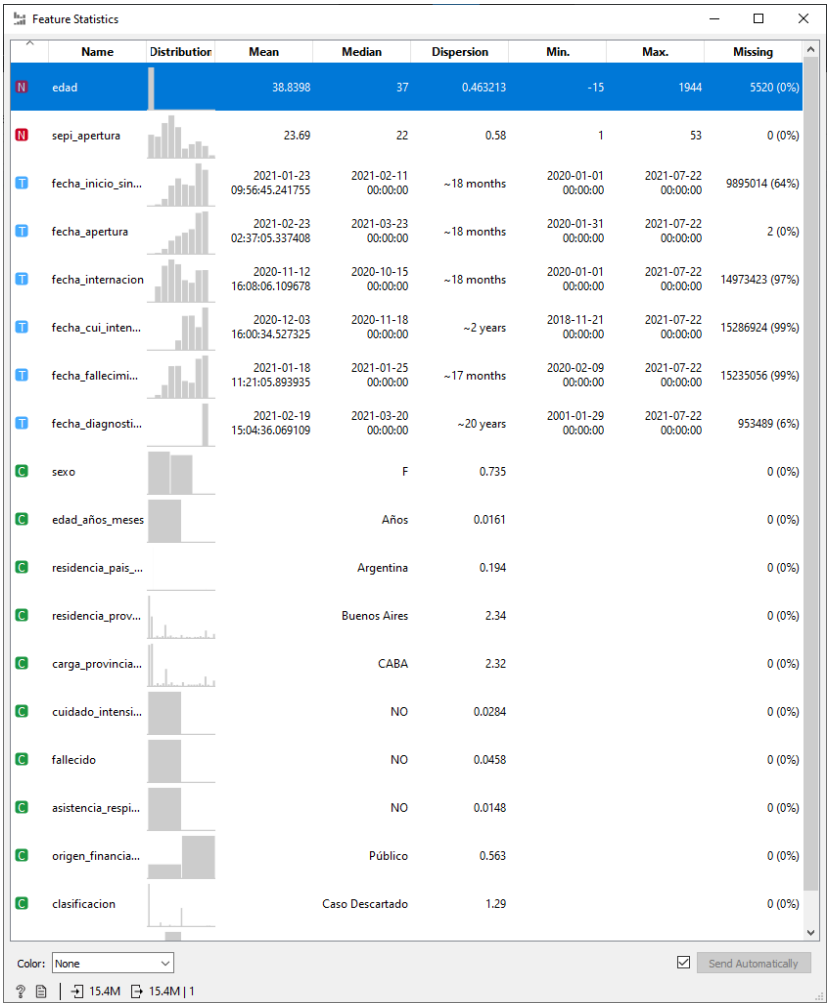


Figura 23

Vista del componente Feature Statistics de Orange



Por cada atributo, el componente brinda información acerca de:

- El tipo de dato.
- El nombre.
- Su distribución, representada por un histograma. Para atributos numéricos, el histograma realiza una agrupación por intervalos.
- El promedio, si se trata de datos numéricos. Nótese que el software transforma los datos de tipo fecha a numérico en este caso (más precisamente, cuenta la cantidad de segundos transcurridos desde el primero de enero de 1970).
- La mediana para atributos numéricos, y la moda para categóricos.
- La dispersión, tratándose del coeficiente de variación para atributos numéricos y de la entropía para atributos categóricos.
- Los valores mínimos y máximos para atributos numéricos.
- La cantidad de campos nulos o faltantes que lo componen.

En base a los datos estadísticos proporcionados por este componente, se pudo extraer la siguiente información útil para realizar la limpieza y selección de datos:

1) El atributo edad presenta campos con valores claramente erróneos (una edad no puede ser negativa ni tomar un valor tan alto como 1944) y probablemente esa sea la causa de la distribución inusual de su histograma. Además, su proporción de valores nulos es muy baja.
2) El atributo sexo posee tres valores distintos (F, M y N.R., este último teniendo una proporción muy baja).

3) Casi todas las edades registradas están expresadas en años.

4) El atributo residencia_pais_nombre posee una dispersión muy baja, siendo casi la totalidad de los casos personas con residencia en Argentina.

5) Más de la mitad de los casos no tienen registrada una fecha de inicio de síntomas, y casi la totalidad de los mismos carecen de fecha de internación, cuidado intensivo o fallecimiento.

6) Existen fechas erróneas de diagnóstico (la más antigua data del año 2001) y de cuidado intensivo (la más antigua data del año 2018). La proporción de valores faltantes para las fechas de diagnóstico y apertura es baja.

7) La fecha de actualización no varía.

Teniendo en cuenta esta información, se procedió con la limpieza y selección de los datos.

4.2.2. Fase de limpieza y selección (parte 1)

Se eliminaron aquellos atributos que a priori se sabían que no serían de utilidad mediante un filtro. Para esto se utilizó el *widget* "Select Columns" que permite ignorar columnas al seleccionarlas (ver **Figura 24**). Los atributos eliminados fueron:

1. **clasificacion**: proporciona información con un nivel demasiado detallado acerca de la clasificación del caso de la persona. Esto puede generar confusión por lo que se opta utilizar el atributo "clasificacion_resumen" en su lugar.

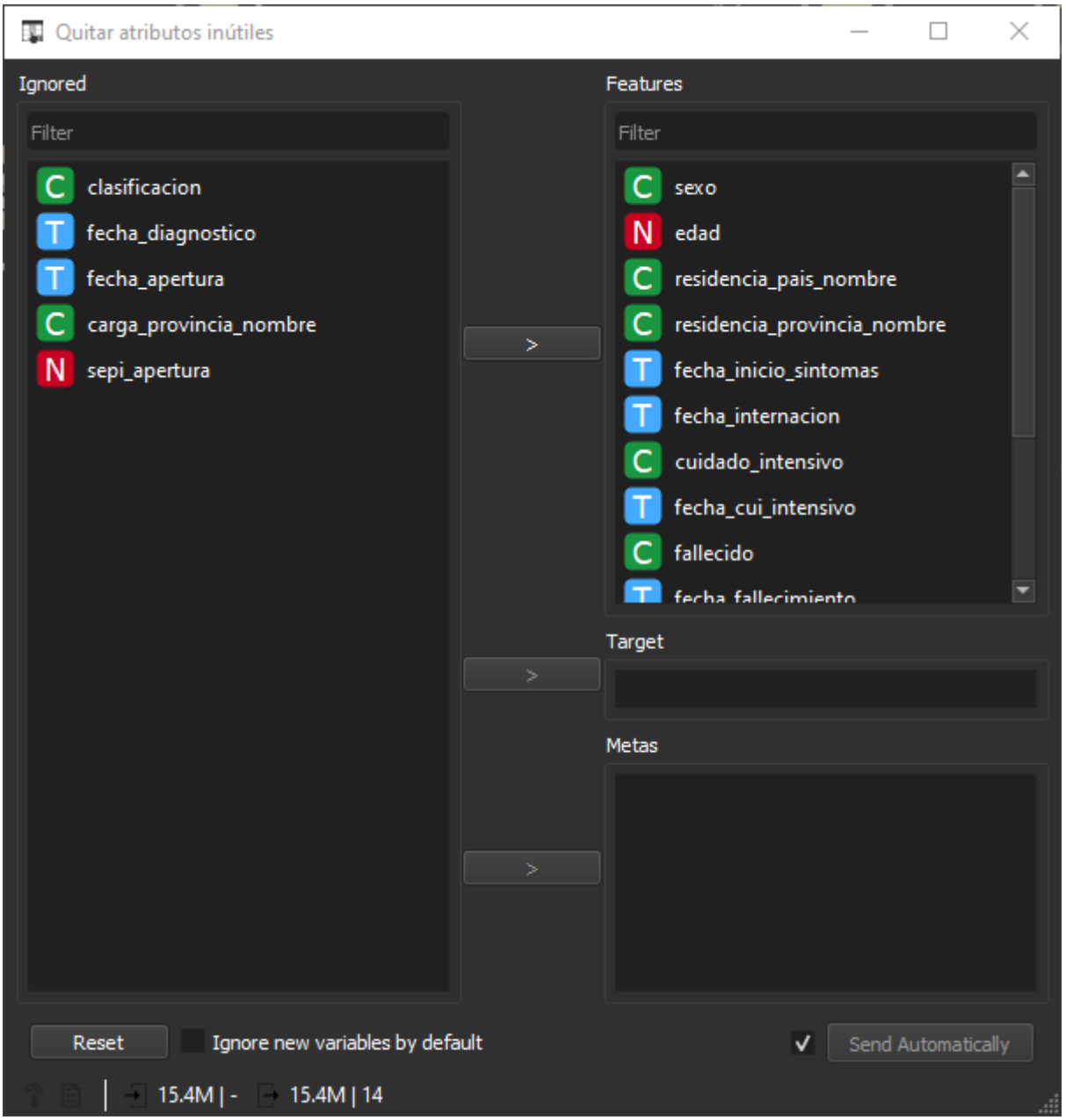
2. **fecha_diagnostico**: no es de interés contar con esta información ya que se utilizará el atributo "fecha_inicio_sintomas" como referencia temporal. La

fecha en que se realiza el diagnóstico está ligada a temas administrativos y no a cuestiones propias de la enfermedad o de los pacientes.

3. **fecha_apertura**: ídem caso anterior, fines administrativos.
4. **sepi_apertura**: fines administrativos.
5. **carga_provincia_nombre**: no resulta de interés conocer la provincia en donde los datos fueron cargados (no confundirse con "residencia_provincia_nombre", la cual sí es de interés).
6. **residencia_departamento_nombre**: debido a que existen muchísimos departamentos en todo el país se consideró que esta información era demasiado detallada. Por eso se prefirió trabajar con las provincias, las cuales tienen un nivel más general.
7. **id_evento_caso**: los identificadores no proporcionan ninguna información útil, por lo que se eliminaron todos los atributos de este tipo.
8. **carga_provincia_id**
9. **residencia_provincia_id**
10. **residencia_departamento_id**
11. **ultima_actualizacion**: al igual que "fecha_diagnostico", este atributo es únicamente de características administrativas.

Figura 24

Componente Select Columns de Orange



Posterior a esto, se hizo un filtro para eliminar aquellos atributos que sean nulos o posean valores erróneos. Para esto se utilizó el *widget* "Select Rows" que permite aplicar condiciones sobre los atributos y seleccionar únicamente las filas que los cumplan (ver **Figura 25**). Las condiciones impuestas fueron:

1. Los valores del atributo “sexo” no deben ser N.R. (se desea conocer el sexo de las personas a analizar).

2. El atributo “edad” no debe ser nulo y sus valores deben comprenderse entre 1 y 103 años (rango razonable de valores).

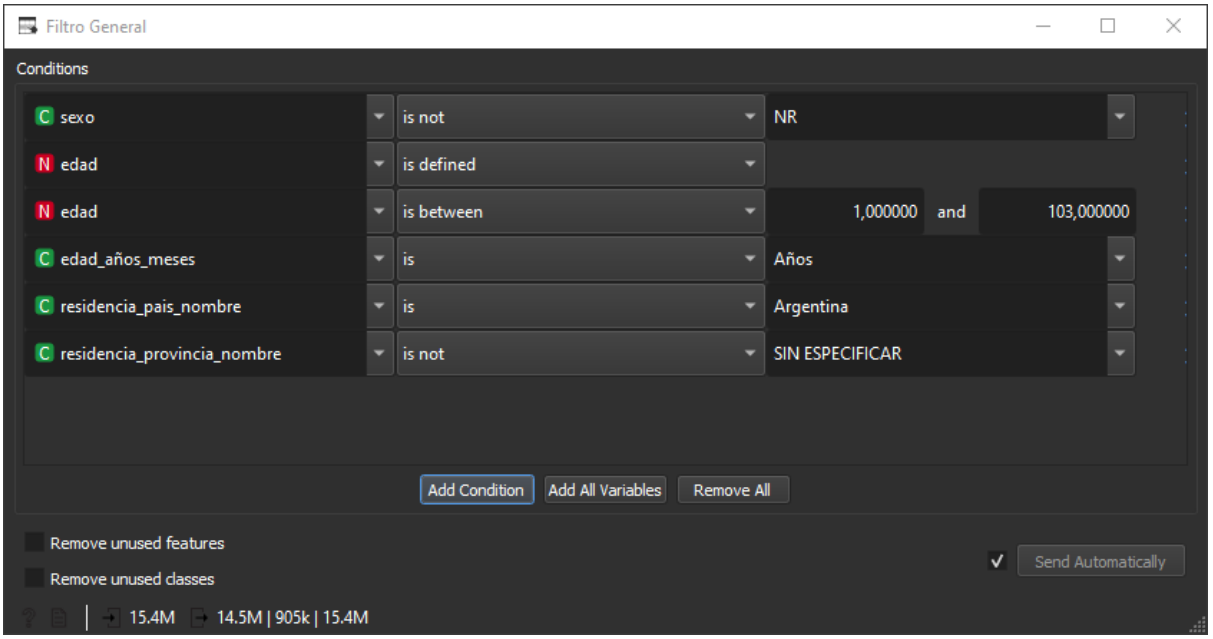
3. El atributo “edad_años_meses” debe contener únicamente el valor “Años” (en concordancia con el punto anterior).

4. El atributo “residencia_pais_nombre” debe contener solamente el valor “Argentina” (corresponde a casi la totalidad de los casos)

5. El atributo “residencia_provincia_nombre” no debe ser “SIN ESPECIFICAR” (se desea conocer la provincia de residencia).

Figura 25

Componente Select Rows de Orange



Luego, se eliminaron los atributos “edad_años_meses” y “residencia_país_nombre”, por resultar redundantes.

A continuación se realizó una segmentación de los datos de tal manera que solo se mantengan aquellos cuyos atributos “fecha_inicio_sintomas”, “fecha_internacion”, “fecha_cui_intensivo” y “fecha_fallecimiento” fueran no nulos. Además, por razones de coherencia, el valor de los atributos “fallecido” y “cuidado_intensivo” debía ser igual a “SI”. Esta segmentación se hizo así principalmente por estas razones:

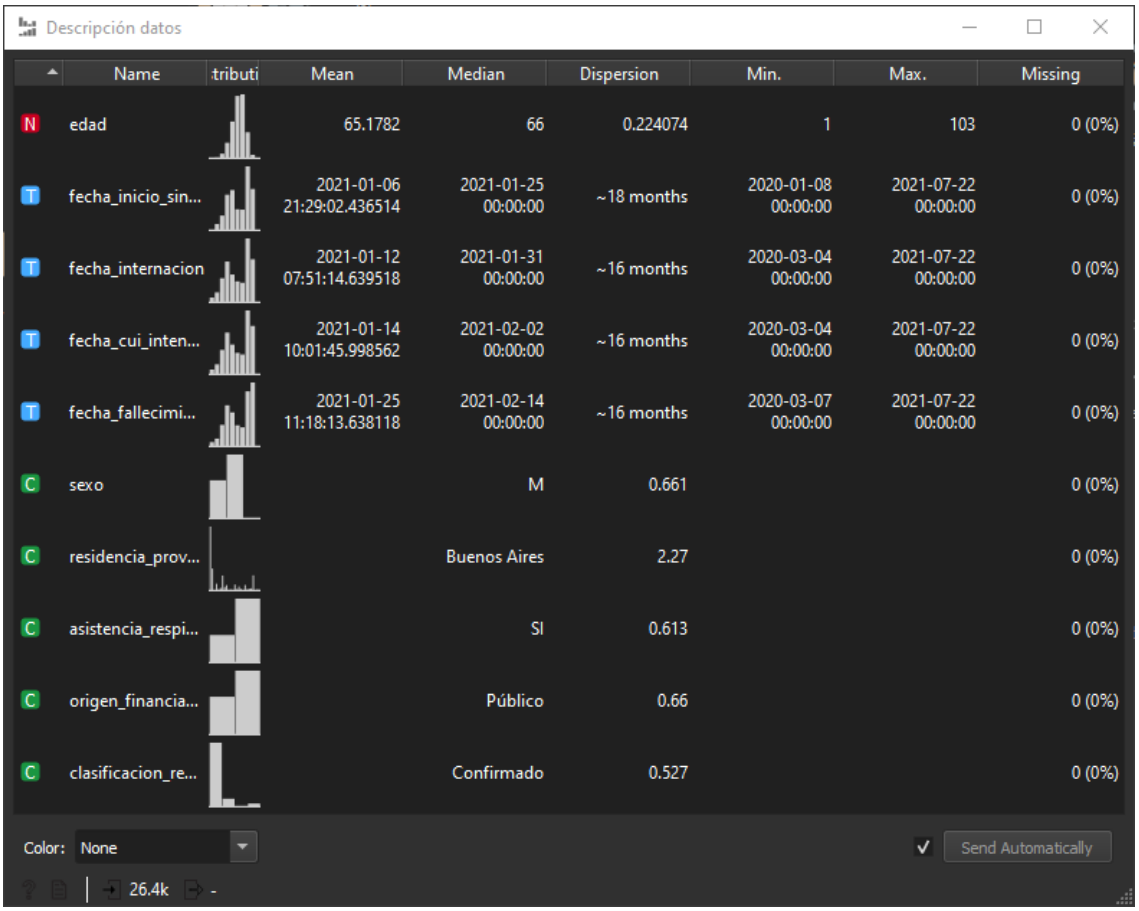
- Al tener un valor definido de las fechas es posible agregar nuevos atributos interesantes a partir de estos datos
- La cantidad obtenida de casos mediante esta segmentación es muchísimo menor que el tamaño total del *dataset* y permite realizar un análisis completo de los mismos sin tener que recurrir a una muestra. Esto es a su vez importante teniendo en cuenta los recursos hardware disponibles (ver **Sección 5.**).

Se utilizaron nuevamente los componentes “Select Rows” para hacer el filtrado y segmentación y “Select Columns” para eliminar los atributos que resultaron redundantes.

Debajo se muestra una descripción de los datos luego de haberse realizado la segmentación, utilizando el componente “Feature Statistics” (**Figura 26**):

Figura 26

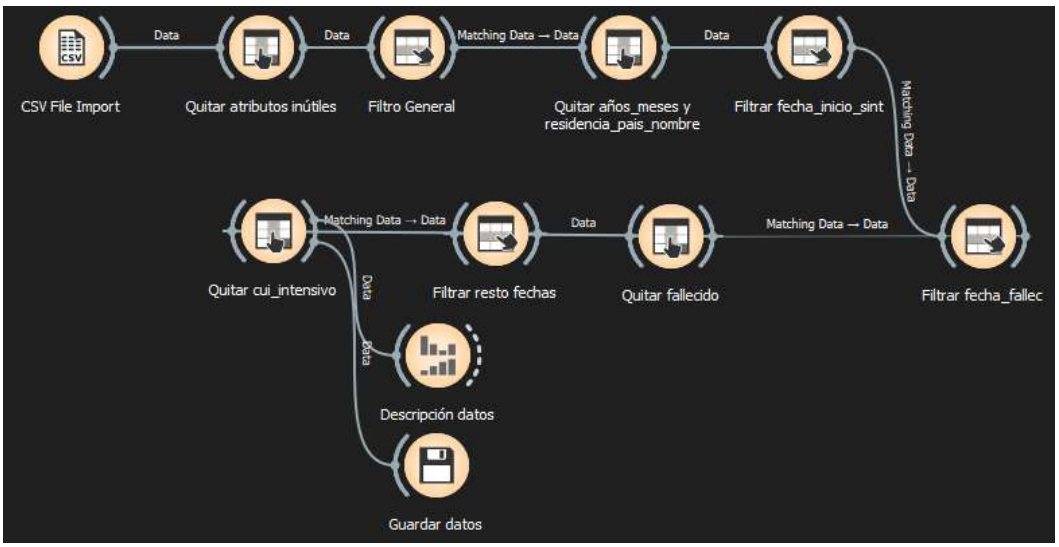
Componente Feature Statistics de Orange al final de la fase de limpieza y selección



También se muestra el *workflow* resultante en Orange (ver **Figura 27**) luego de realizar todos los pasos descritos anteriormente.

Figura 27

Workflow de Orange resultante de la fase de limpieza y selección



4.2.3. Fase de transformación (parte 1)

La siguiente tarea consistió en agregar nuevos atributos a partir de los originales. Aprovechando que todos los casos poseen atributos de fecha no nulos, se agregaron los siguientes utilizando el *widget* "Feature Constructor" (**Figura 28**):

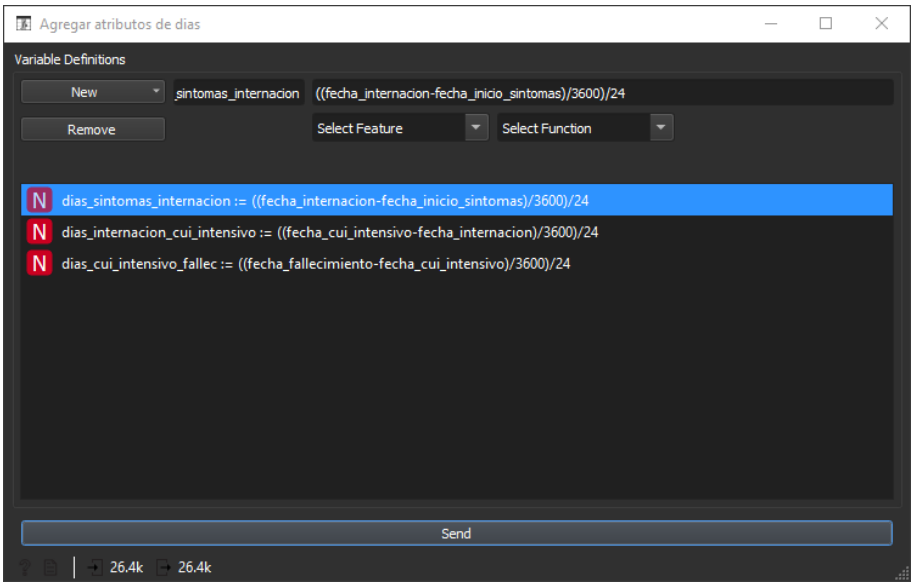
Tabla 2

Descripción de atributos agregados en la fase de transformación (1)

Atributo	Tipo de dato	Descripción
dias_sintomas_internacion	Número entero (*integer*)	Cantidad de días transcurridos entre fecha_inicio_sintomas y fecha_internacion
dias_internacion_cui_intensivo	Número entero (*integer*)	Cantidad de días transcurridos entre fecha_internacion y fecha_cui_intensivo
dias_cui_intensivo_fallec	Número entero (*integer*)	Cantidad de días transcurridos entre fecha_cui_intensivo y fecha_fallecimiento

Figura 28

Componente Feature Constructor de Orange



4.2.4. Fase de limpieza y selección (parte 2)

Teniendo los nuevos atributos agregados (**Tabla 2**), aquellos correspondientes a las fechas ("fecha_inicio_sintomas", "fecha_internacion", "fecha_cui_intensivo" y "fecha_fallecimiento") fueron eliminados por no aportar ninguna información.

Utilizando el componente "Select Rows" se impuso como condición que el valor de estos atributos nuevos sea mayor o igual que cero por razones de coherencia, por ejemplo, una persona no puede haber fallecido antes de haber ingresado a cuidados intensivos.

Haciendo uso del *widget* "Distributions", puede verse qué distribuciones poseen los atributos agregados (dias_sintomas_internacion, dias_internacion_cui_intensivo, dias_cui_intensivo_fallec), visualizándolas en forma de histogramas (ver **Figuras 29**, **30**, **31**).

Figura 29

Componente Distributions de Orange para el atributo dias_sintomas_internacion

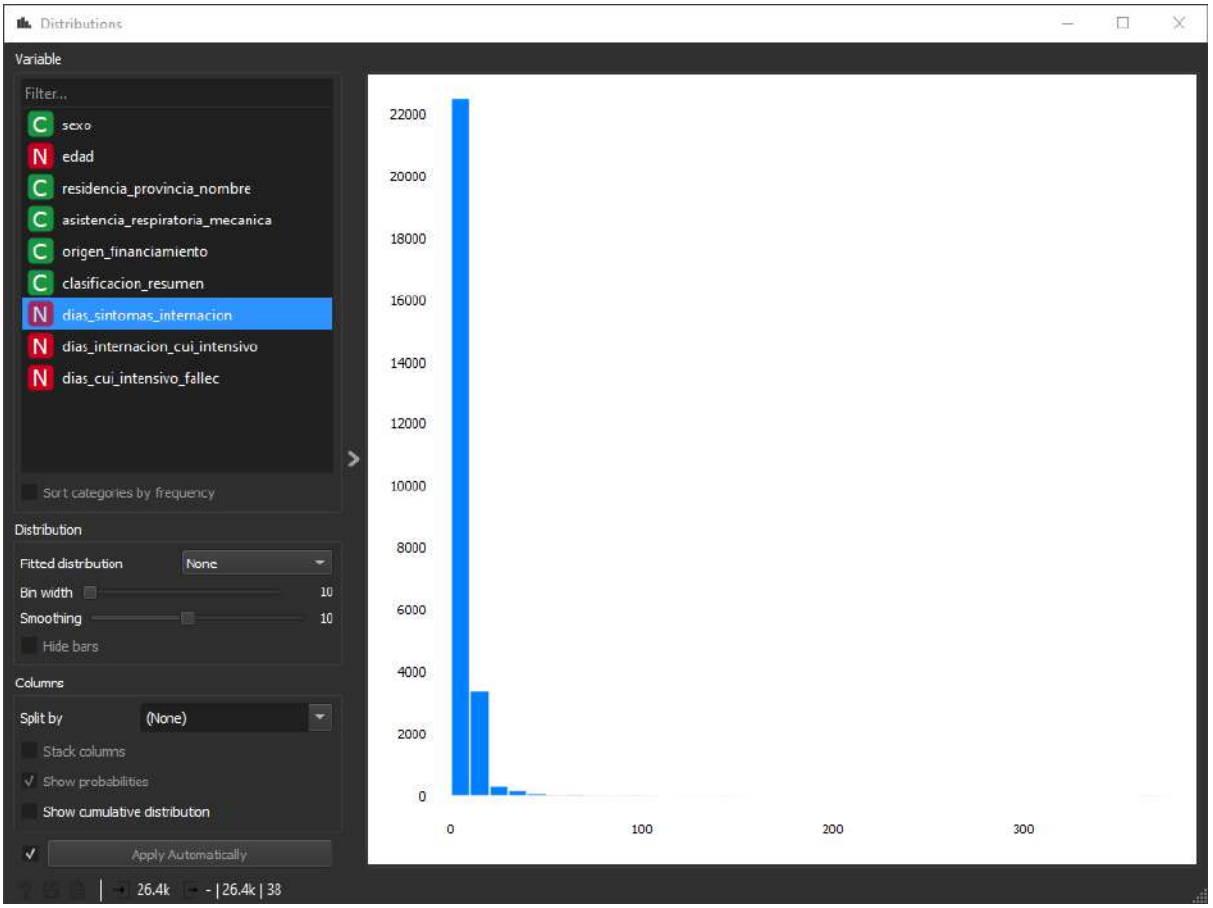


Figura 30

Componente Distributions de Orange para el atributo dias_internacion_cui_intensivo

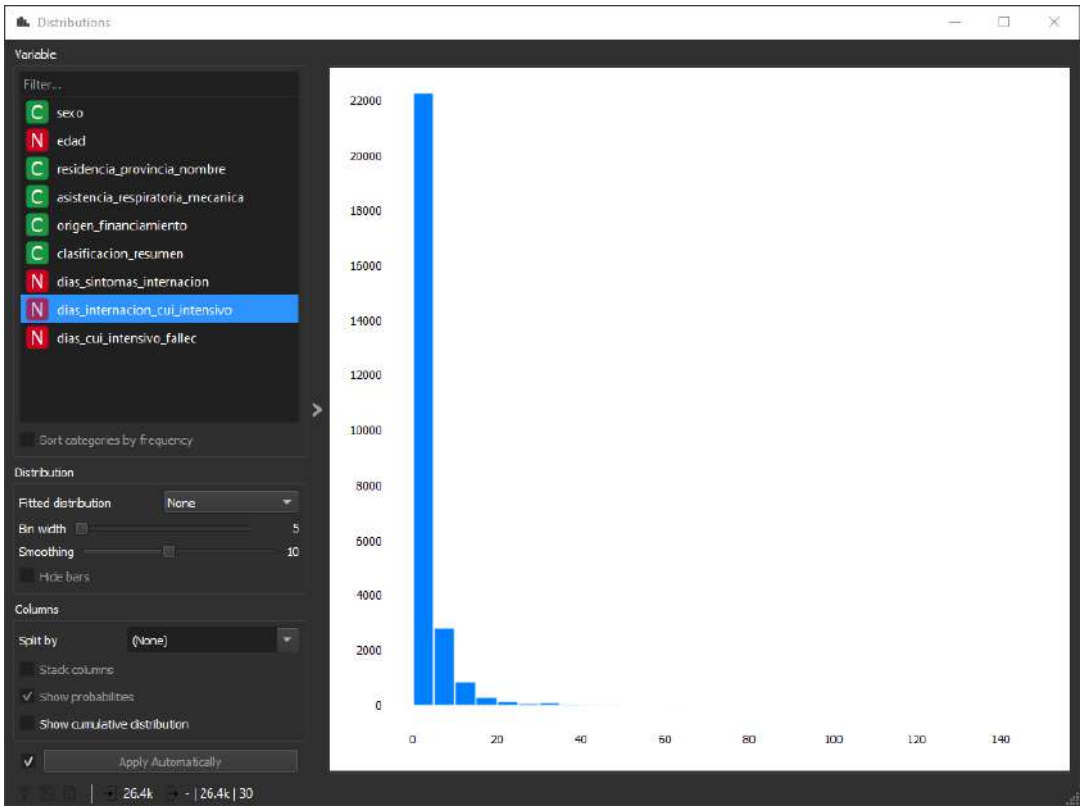
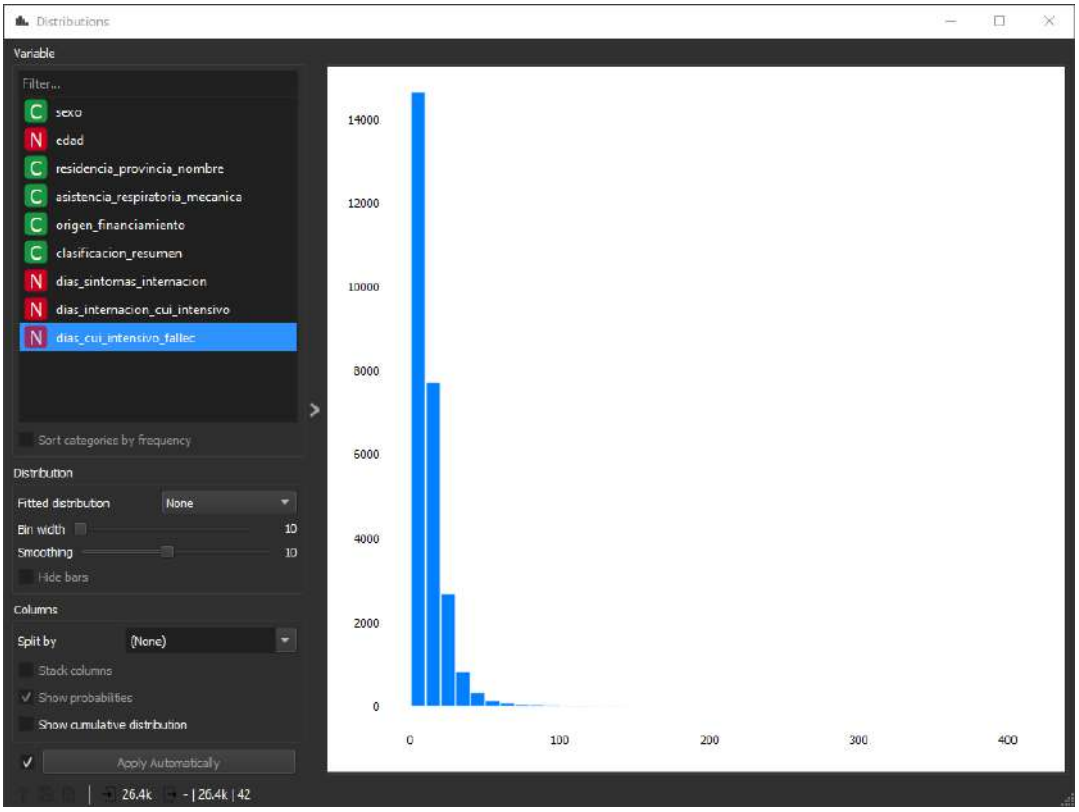


Figura 31

Componente Distributions de Orange para el atributo dias_cui_intensivo_fallec

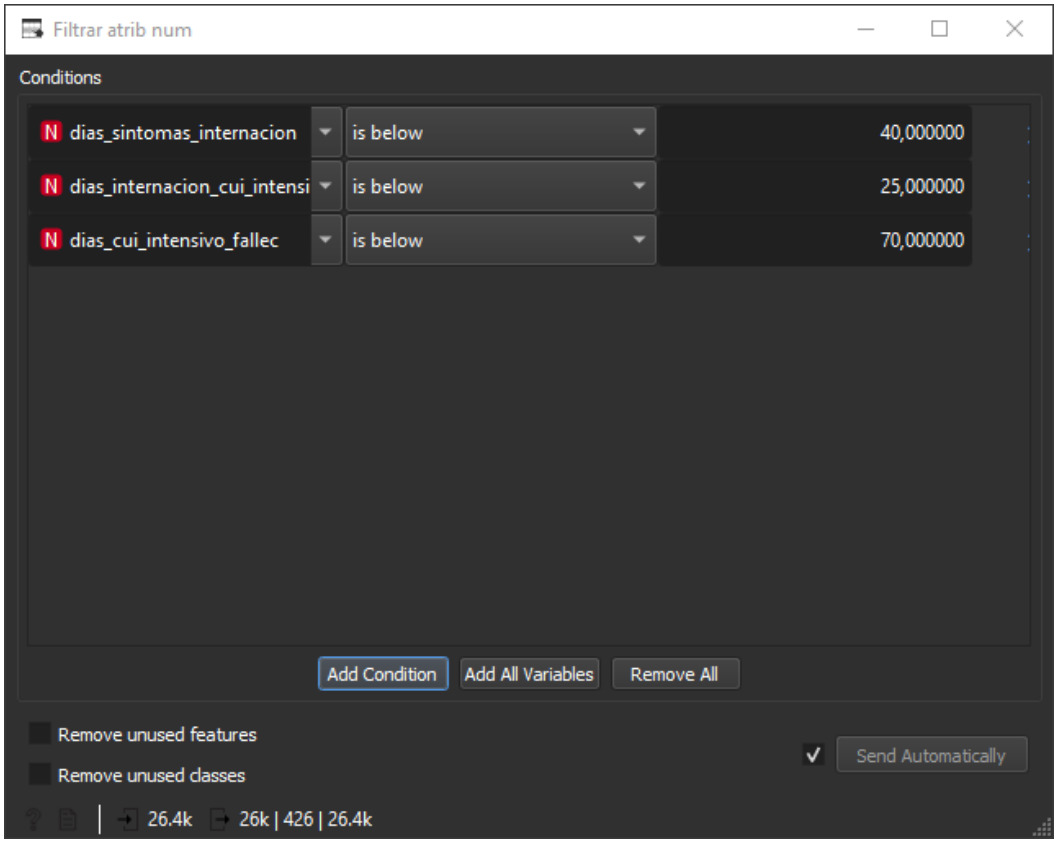


Puede observarse que las tres distribuciones siguen el mismo comportamiento, con la mayoría de los datos concentrados en los valores más bajos (asimétricas y con sesgo a la derecha). También se descubre que para los tres atributos existen, aunque pocos, valores grandes aislados. Esto supuso un problema, puesto que estos valores *outliers* se traducen en ruido y afectan la performance de las técnicas de *data mining*. Se decidió entonces limpiar estos datos imponiendo la condición de que estos atributos no superen un valor límite (ver **Figura 32**). Este valor límite se fijó tal que más del 99% de los datos se encuentren comprendidos bajo este valor. Los valores límites para cada atributo son, respectivamente:

1. dias_sintomas_internacion: 40 días
2. dias_internacion_cui_intensivo: 25 días
3. dias_cui_intensivo_fallec: 70 días

Figura 32

Filtros aplicados sobre los nuevos atributos



El último paso de la fase de pre-tratamiento de los datos fue la normalización de los atributos de tipo numérico. Esta tarea no se realizó en Orange y se incluyó en el mismo *script* que se utilizaría posteriormente en la fase de Data Mining por razones de flexibilidad.

El tipo de normalización elegida para utilizarse sobre los datos fue la normalización lineal uniforme por ser ampliamente aplicada en técnicas basadas en distancia y de simple implementación [2]. Esto significa que los datos numéricos serían mapeados linealmente al intervalo [0,1].

En esta instancia, las características del *dataset* resultante se muestran en la **Tabla 3**:

Tabla 3

Características resultantes del dataset luego de la aplicación de las fases de limpieza, selección y transformación

Atributo	Tipo	Descripción
sexo	Categórico	Sexo registrado del paciente
edad	Numérico	Edad del paciente
dias_sintomas_internacion	Numérico	Días transcurridos desde que la persona presenta síntomas hasta que fue internada
dias_sintomas_cui_intensivos	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos
dias_cui_intensivo_fallecimiento	Numérico	Días desde que el paciente entró a cuidados intensivos hasta que falleció
residencia_provincia_nombre	Categórico	Provincia de residencia
asistencia_respiratoria_mecanica	Categórico	Toma el valor SI/NO dependiendo si el paciente utilizó o no respirador mecánico
origen_financiamiento	Categórico	Toma el valor PÚBLICO/PRIVADO dependiendo el tipo de institución donde fue atendido el paciente
clasificacion_resumen	Categórico	Toma los valores CONFIRMADO/DESCARTADO/ SOSPECHOSO
Otras características: todas las personas que componen el *dataset* corresponden a pacientes fallecidos y que residían en Argentina. **Cantidad total de datos:** 25996.		

4.2.5. Fase de data mining: k-prototypes (parte 1)

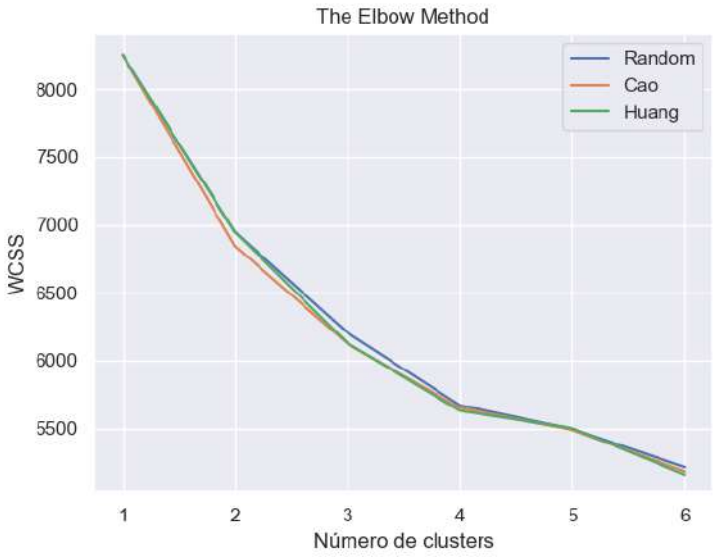
Se comenzó por esta técnica ya que resulta mucho más simple de comprender y ejecutar que el SOM. Además, al ser capaz de procesar datos mixtos, no se necesita realizar una binarización de los atributos categóricos.

Se hicieron varias pruebas del algoritmo variando γ y *k* (ver **Cap. 3.3.2.**) y graficando el elbow method (**Cap. 3.4.1.**) en cada caso para poder obtener su valor óptimo. Además, se probaron distintos métodos de inicialización del algoritmo.

Se probaron los métodos de inicialización que ofrece la librería (ver **Figura 33**). Estos consisten en la inicialización de centroides al azar (*random*) y los métodos con enfoques probabilísticos (Huang y Cao) [12, 41]. Si bien existe una preferencia por los últimos dos tanto en la bibliografía consultada como en la documentación de la librería, en la práctica no se encontraron diferencias notables entre las tres al ser aplicarlas al *dataset*.

Figura 33

Elbow plot de los distintos tipos de inicialización de k-prototypes



Como puede observarse, el comportamiento de la curva en el elbow plot es muy parecido en todos los casos. En esta ejecución se utilizó un γ estimado de forma automática por la librería, pero esto mismo ocurrió para otras pruebas en donde γ se fijó manualmente

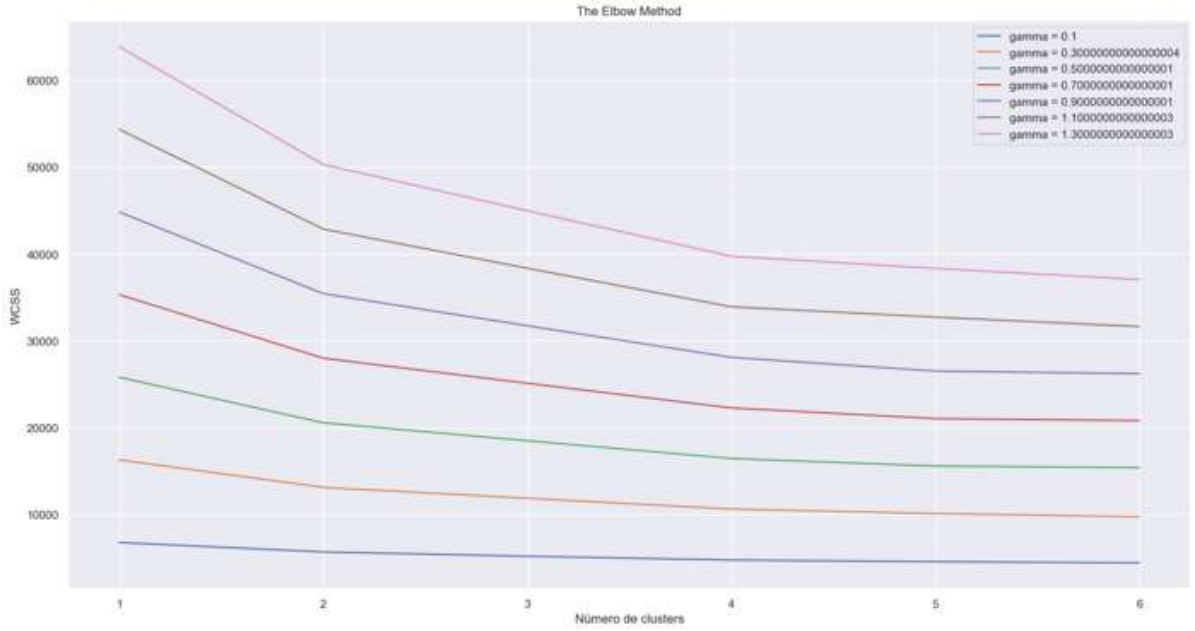
con otros valores. Este comportamiento prácticamente invariante también se vio reflejado en los puntajes obtenidos al aplicar silhouette score sobre los resultados.

Para todas las ejecuciones posteriores, se usó el método “Cao” por ser uno de los más usados en la práctica.

Para establecer un valor adecuado de γ, se realizaron varias ejecuciones incrementando su valor en saltos constantes (**Figura 34**). Por ejemplo, en el siguiente gráfico se muestra el elbow plot correspondientes a valores de γ entre 0,1 y 1,3 en saltos de 0,2.

Figura 34

Elbow plot para valores de γ entre 0,1 y 1,3

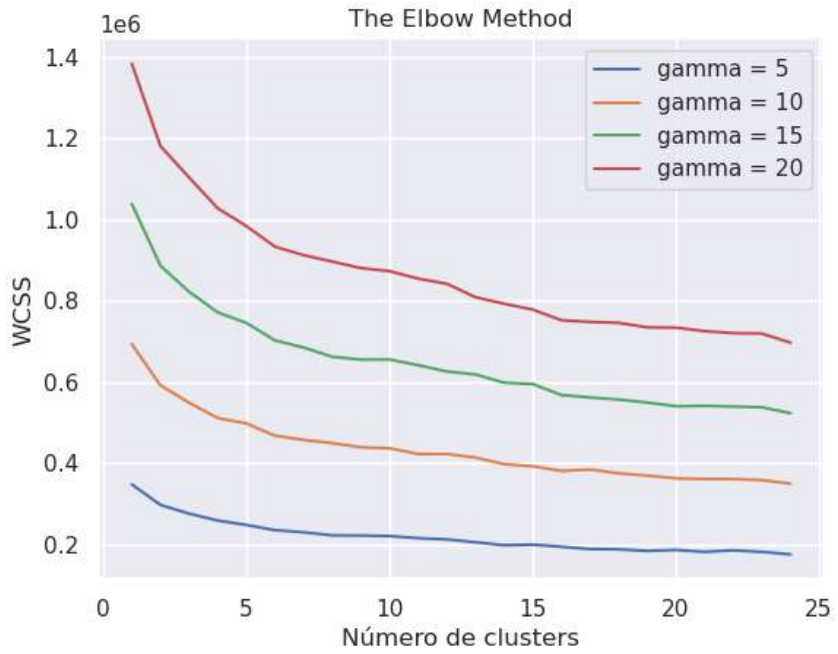


Si bien el valor de γ cambia, la curva se mantiene similar para todos los valores mostrados (resulta más difícil de visualizar en los valores más pequeños debido a la escala).

La misma prueba se realizó para valores de γ superiores a uno. En el siguiente gráfico (**Figura 35**) se muestra el elbow plot correspondientes a valores de γ entre 5 y 20 en saltos de 5. También se muestran los valores de WCSS (suma de los cuadrados intra-cluster) correspondientes a valores mayores de k (nótese cómo las variaciones se vuelven cada vez menores).

Figura 35

Elbow plot para valores de γ entre 5 y 20



Si bien en este gráfico la forma general de las curvas también es bastante parecida para los distintos valores de γ, sí puede apreciarse la presencia de algunas variaciones en ciertos puntos. Se comprobó que para mayores valores de γ estas variaciones se transformaban en oscilaciones con picos agudos haciendo que las curvas se comportaran de forma inestable.

Luego de realizar un análisis y viendo que para este conjunto de datos el valor de γ tenía poco impacto para la formación de clusters en k-prototypes, se decidió que su valor sea igual a uno. Se eligió este valor debido a la "estabilidad" de los valores bajos de este parámetro y para asignarle el mismo peso tanto a la componente numérica como a la categórica.

Definidos el método de inicialización de centroides y el valor de γ, se realizaron varias ejecuciones de k-prototypes haciendo variar k. Se analizaron los resultados de cada corrida mediante los puntajes obtenidos de silhouette score. El cálculo de la matriz de

distancias utilizada en dicha métrica tardó aproximadamente 5 horas utilizando el **Equipo n°1** (ver **Sección 5.**). Este cálculo sólo fue necesario para la ejecución correspondiente al primer valor de k. Una vez calculada, la matriz fue guardada y reutilizada para obtener el puntaje de las ejecuciones para el resto de valores de k. En consecuencia, el tiempo de ejecución se redujo a pocos minutos.

Los puntajes obtenidos de silhouette score en esta primera fase de *data mining* no fueron buenos. Los más altos se registraron para los valores de $k = 2$ y $k = 4$, en correspondencia con los puntos sugeridos del elbow plot (**Figuras 33** y **34**). En la **Tabla 4** se muestran en detalle los valores:

Tabla 4

Puntajes de silhouette score de la primera fase de data mining

Valor de k utilizado	2	4
Silhouette score *cluster* 1	0,3092	0,3042
Silhouette score *cluster* 2	0,2392	0,1012
Silhouette score *cluster* 3	-	0,1946
Silhouette score *cluster* 4	-	0,3453
Silhouette score promedio	0,2893	0,2526

Los valores obtenidos en general (ya sea tanto para los *clusters* como para el promedio total) son cercanos al cero. Esto significa que existe un alto grado de solapamiento entre los grupos y que sus límites no están bien definidos. Esto puede deberse a varias razones:

1. No pueden establecerse grupos claros con los atributos actuales.
2. Los parámetros de entrada no se encuentran bien ajustados.

3. Los grupos se forman, pero de manera no lineal. Esto da la pauta de la necesidad de utilizar una técnica más sofisticada para encontrarlos (por ejemplo, mapas auto-organizados).

A pesar de seguir haciendo pruebas, cambiando los parámetros de entrada, no se consiguió una mejora por lo que se abordó el problema haciendo foco en la razón número 1.

4.2.6. Fase de transformación (parte 2)

Se decidió incorporar un atributo que refleje alguna característica temporal de los casos con el objetivo de poder lograr una mejor separación de los *clusters* a través del aporte de nueva información. Se agregó un atributo llamado “mes/año”, de carácter categórico, que indica el mes y año del inicio de síntomas de la persona. Este atributo fue construído a través de un script en Python e incorporado de forma automática al *dataset*. A partir de ahora y para el resto de la sección de k-prototypes, cuando se haga referencia al *dataset*, será este conjunto de datos que queda luego de esta fase de transformación y cuyos atributos se describen en la **Tabla 5**.

Tabla 5

Características resultantes del dataset luego de la aplicación de las fase de transformación (parte 2).

Atributo	Tipo	Descripción
sexo	Categórico	Sexo registrado del paciente
edad	Numérico	Edad del paciente
dias_sintomas_internacion	Numérico	Días transcurridos desde que la persona presenta síntomas hasta que fue internada
dias_sintomas_cui_intensivos	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos.
dias_cui_intensivo_fallecimiento	Numérico	Días desde que el paciente entró a cuidados intensivos hasta que falleció
residencia_provincia_nombre	Categórico	Provincia de residencia

asistencia_respiratoria_mecanica	Categórico	Toma el valor SI/NO dependiendo si el paciente utilizó o no respirador mecánico
origen_financiamiento	Categórico	Toma el valor PÚBLICO/PRIVADO dependiendo el tipo de institución donde fue atendido el paciente
clasificacion_resumen	Categórico	Toma los valores CONFIRMADO/DESCARTADO/ SOSPECHOSO
mes/año	Categórico	Mes y año en que se produjo la fecha de inicio de síntomas
Otras características: todas las personas que componen el *dataset* corresponden a pacientes fallecidos y que residían en Argentina. **Cantidad total de datos:** 25996.		

4.2.7. Fase de data mining: k-prototypes (parte 2)

Se realizaron nuevas ejecuciones automatizadas de k-prototypes para analizar los resultados de los elbow plot y silhouette score pero desafortunadamente no hubo mejoras con respecto a los resultados obtenidos anteriormente. Los elbow plots no exhibían puntos claros que sugieran un número potencial de *clusters* y los puntajes obtenidos en silhouette score seguían reflejando un evidente solapamiento de grupos. Estos resultados persistían aún variando el valor de γ y probando con otros métodos de inicialización. Pese a contar con un atributo que aporte información sobre la ubicación temporal, el mismo no logró aportar claridad en la formación de grupos.

Se replicó todo lo anterior con una muestra del *dataset* tomando solo casos pertenecientes a la provincia de Buenos Aires (aproximadamente diez mil casos), pero la calidad de los resultados siguió siendo la misma.

Siendo que la misma naturaleza de los datos impedía obtener buenos resultados con este algoritmo de *data mining*, se continuó con SOM, el cual es capaz de detectar agrupaciones de manera no lineal [42]. Antes, sin embargo, se detalla el mejor resultado

obtenido en esta última etapa a través de las **Tablas 6** y **7**. Este resultado se corresponde con valores de $\gamma = 1$ y $k = 6$ y el silhouette score promedio fue de 0,1504. A pesar de todo, estos resultados fueron presentados a un profesional del área de la Salud quien expresó que los mismos se encontraban bien ubicados con respecto a la realidad.

Tabla 6

Composición de los clusters formados por k-prototypes

Cluster	Total	Porcentaje Casos	Var. intra-cluster	sexo	edad	residencia_provincia_nombre
0	7820	30.07	3.38	M (87.15%)	57.48 (9.00)	Buenos Aires (35.20%)
1	3138	12.07	5.43	F (68.20%)	77.99 (7.00)	Buenos Aires (54.94%)
2	4150	15.96	4.22	F (83.69%)	63.58 (9.00)	CABA (25.13%)
3	4424	17.01	3.58	M (96.23%)	67.59 (8.00)	Buenos Aires (47.29%)
4	2590	9.99	3.57	F (91.30%)	70.20 (7.00)	Buenos Aires (56.25%)
5	3874	14.90	4.49	M (84.74%)	66.08 (8.00)	Buenos Aires (37.69%)

Cluster	asistencia_respiratoria_mecanica	origen_financiamiento	clasificacion_resumen
0	SI (94.60%)	Público (95.58%)	Confirmado (89.00%)
1	NO (97.29%)	Privado (79.80%)	Confirmado (73.39%)
2	SI (89.54%)	Público (95.25%)	Confirmado (90.34%)
3	SI (91.52%)	Privado (89.60%)	Confirmado (82.55%)
4	SI (98.11%)	Privado (80.49%)	Confirmado (78.30%)
5	NO (90.91%)	Público (85.60%)	Confirmado (84.56%)

Cluster	dias_sintomas_internacion	dias_internacion_cui_intensivo	dias_cui_intensivo_fallec
0	5.99 (3.00)	2.03 (0.00)	11.42 (6.00)
1	2.92 (2.00)	0.68 (0.00)	7.65 (4.00)
2	4.31 (3.00)	2.65 (1.00)	10.07 (5.00)
3	4.73 (3.00)	2.79 (1.00)	13.27 (7.00)
4	4.39 (3.00)	1.40 (0.00)	9.39 (5.00)
5	5.00 (3.50)	1.09 (0.00)	9.78 (5.00)

Cluster	mes/año
0	05/2021 (35.93%); 10/2020 (8.02%); 04/2021 (7.88%); 08/2020 (6.32%)
1	04/2021 (14.79%); 05/2021 (10.45%); 08/2020 (10.26%); 10/2020 (9.66%)
2	06/2021 (27.13%); 04/2021 (11.42%); 05/2021 (9.93%); 10/2020 (8.67%)
3	04/2021 (29.41%); 10/2020 (8.77%); 08/2020 (7.91%); 05/2021 (7.50%)
4	09/2020 (26.24%); 05/2021 (11.70%); 08/2020 (8.43%); 10/2020 (8.00%)
5	09/2020 (24.88%); 10/2020 (8.93%); 04/2021 (7.80%); 06/2021 (7.64%)

Tabla 7

Referencias sobre los resultados de k-prototypes

Cluster	Nro. Identificador del *cluster* (Total: 6 *clusters*)
Total	Cantidad de casos que componen el *cluster* (Total: 25.996 casos)
Porcentaje Casos	Porcentaje con respecto al número total de casos del *dataset*
Var. intra-cluster	Varianza intra *cluster*
sexo	Sexo de la persona (Masculino | Femenino)
edad	Edad de la persona ([1-103] años)
residencia_provincia_nombre	Provincia de residencia de la persona
asistencia_respiratoria_mecanica	Indicador si la persona requirió asistencia respiratoria mecánica (SI | NO)
origen_financiamiento	Origen de financiamiento (Público | Privado)
clasificacion_resumen	Clasificación del caso (Confirmado | Sospechoso | Descartado)
dias_sintomas_internacion	Cantidad de días transcurridos entre la fecha de comienzo de síntomas y la fecha de internación
dias_internacion_cui_intensivo	Cantidad de días transcurridos entre la fecha de internación y la fecha de entrada a cuidado intensivo
dias_cui_intensivo_fallec	Cantidad de días transcurridos entre la fecha de entrada a cuidado intensivo y la fecha de fallecimiento
mes/año	Mes y año en que se produjo la fecha de inicio de síntomas

Algunas aclaraciones:

En todos los casos, las personas cuentan con las siguientes características:

- Residían en Argentina
- Han tenido síntomas de Covid-19
- Han sido internadas
- Han entrado en cuidado intensivo
- Han fallecido

Para los atributos de tipo númerico (edad, dias_sintomas_internacion, dias_internacion_cui_intensivo y dias_cui_intensivo_fallec) los valores que se indican en la **Tabla 6** corresponden al promedio de los valores del *cluster* correspondiente y los que están entre paréntesis indican la desviación mediana absoluta como medida de dispersión. Esta medida de dispersión estadística fue elegida porque resulta más robusta frente a valores atípicos en contraparte a la desviación estándar [43].

Para los atributos de tipo categórico, los valores que se indican en la tabla corresponden a la moda de los valores del *cluster* correspondiente y los valores entre paréntesis indican el porcentaje de casos cuyo valor es igual al de la moda. Para el caso del atributo "mes/año" se muestran los cuatro valores más frecuentes.

Como conclusión, luego del análisis de las tablas de resultados correspondientes a k-prototypes se puede destacar lo siguiente, pudiéndose hacer a futuro un análisis más profundo y completo de los datos:

1. La mayor incidencia de casos para un periodo dentro de un *cluster* se dio en mayo de 2021 (35,93%) para el cluster 0, compuesto por mayoría masculina (87,15%) con una edad promedio de 57 años (+/- 9). La mayoría (94,60%), necesitó asistencia respiratoria mecánica, 6 días promedio hasta que la persona fue internada, 2 días promedio hasta que pasó a cuidados intensivos, y 11 días promedio (+/- 6) hasta el fallecimiento. Este *cluster* representa el 30,07% de los casos del dataset.

2. El *cluster* 3 refleja también una mayoría masculina (96,23%) con edad promedio 67 años (+/- 8), donde el mayor porcentaje de incidencia en el *cluster* fue para el periodo abril 2021 (29,41%). En este *cluster*, más del 91% usó asistencia respiratoria mecánica, con casi 5 días promedio hasta la internación, casi 3 días promedio hasta pasar a cuidados intensivos y 13 días promedio (+/- 7) hasta el fallecimiento. Este cluster representa el 17% de los casos totales.

3. Con respecto a las mujeres, el *cluster* 4 está compuesto por un 91,30% de personas de sexo femenino, con un promedio de edad de 70 años (+/-7). Este *cluster* representa un 10% de los casos totales. Más del 98% utilizó asistencia respiratoria mecánica, con 4 días promedio hasta la internación, 1 día y medio promedio hasta pasar a cuidados intensivos y 9 días (+/-5) hasta el fallecimiento. El 24,88% de estos casos ocurrieron durante septiembre de 2020.

4.2.8. Fase de transformación (parte 3)

Debido a la elección de la versión clásica del SOM se eliminó el atributo "mes/año" porque la binarización de este atributo implica la creación de uno por cada combinación de mes y año existente, lo que aumenta la dimensionalidad de forma considerable. Por la misma razón también se eliminó el atributo "residencia_provincia_nombre" y se lo reemplazó por otro que indique la región correspondiente del país. Para la asignación de regiones se utilizó como criterio la división estadística que realiza el INDEC [44].

La versión resultante del *dataset* luego de estas modificaciones se detalla en la **Tabla 8**:

Tabla 8

Características resultantes del dataset luego de la aplicación de las fase de transformación

Atributo	Tipo	Descripción
sexo	Categórico	Sexo registrado del paciente
edad	Numérico	Edad del paciente
dias_sintomas_internacion	Numérico	Días transcurridos desde que la persona presenta síntomas hasta que fue internada
dias_sintomas_cui_intensivos	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos
dias_cui_intensivo_fallecimiento	Numérico	Días desde que el paciente entró a cuidados intensivos hasta que

		falleció
region	Categórico	Región de residencia según clasificación de INDEC (BUENOS AIRES/PAMPEANA/NEA/NOA/ CUYO/PATAGONIA)
asistencia_respiratoria_mecanica	Categórico	Toma el valor SI/NO dependiendo si el paciente utilizó o no respirador mecánico
origen_financiamiento	Categórico	Toma el valor PÚBLICO/PRIVADO dependiendo el tipo de institución donde fue atendido el paciente
clasificacion_resumen	Categórico	Toma los valores CONFIRMADO/DESCARTADO/ SOSPECHOSO
Otras características: todas las personas que componen el *dataset* corresponden a pacientes fallecidos y que residían en Argentina. **Cantidad total de datos:** 25996.		

Este conjunto de datos fue utilizado con el objetivo de un primer acercamiento con los mapas auto-organizados a través del uso de Java SOMToolbox (ver **Anexo 2**).

Para las pruebas con Python (ver **Anexo 2**) y MATLAB©, se incorporaron otras características temporales al *dataset* para analizar el impacto en los resultados. Para evitar que se generen demasiados atributos a causa de la binarización, se agregó la estación en la que se produjo el inicio de síntomas en lugar del mes. También se muestra el año en un atributo aparte. La nueva versión del *dataset* se describe en la **Tabla 9**:

Tabla 9

Características resultantes del dataset luego de la aplicación de las fase de transformación

Atributo	Tipo	Descripción
sexo	Categórico	Sexo registrado del paciente
edad	Numérico	Edad del paciente
dias_sintomas_internacion	Numérico	Días transcurridos desde que la persona presenta síntomas hasta que fue internada
dias_sintomas_cui_intensivos	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos

dias_cui_intensivo_fallecimiento	Numérico	Días desde que el paciente entró a cuidados intensivos hasta que falleció
region	Categórico	Región de residencia según clasificación de INDEC (BUENOS AIRES/PAMPEANA/NEA/NOA/ CUYO/PATAGONIA)
asistencia_respiratoria_mecanica	Categórico	Toma el valor SI/NO dependiendo si el paciente utilizó o no respirador mecánico
origen_financiamiento	Categórico	Toma el valor PÚBLICO/PRIVADO dependiendo el tipo de institución donde fue atendido el paciente
clasificacion_resumen	Categórico	Toma los valores CONFIRMADO/DESCARTADO/ SOSPECHOSO
estacion	Categórico	Estación del año al momento de inicio de síntomas. Toma los valores OTOÑO/INVIERNO/PRIMAVERA/VERANO
año	Categórico	Año al momento de inicio de síntomas. Toma los valores 2020/2021
Otras características: todas las personas que componen el *dataset* corresponden a pacientes fallecidos y que residían en Argentina. **Cantidad total de datos:** 25996.		

4.2.9. Fase de data mining: SOM

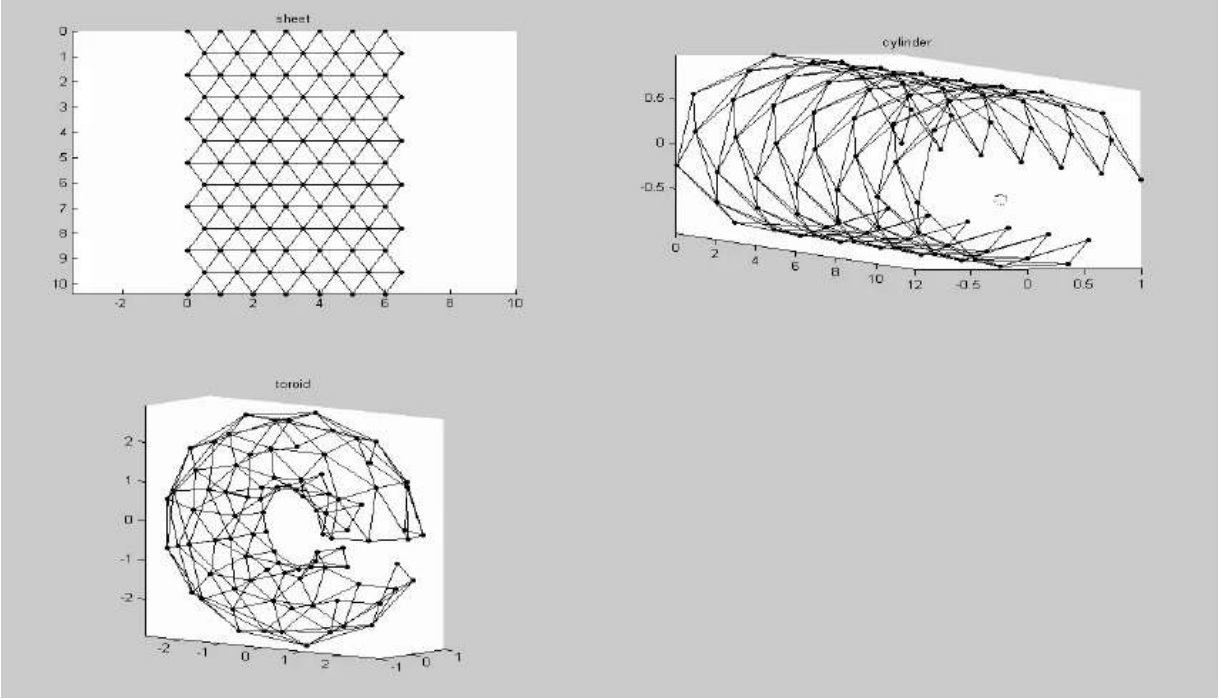
Posterior a la exploración de distintas plataformas como Java SOMToolbox y Minisom Python (ver **Anexo 2**), se seleccionó SOMToolbox de MATLAB©, una herramienta clásica y robusta desarrollada en la Universidad de Helsinki. Esto es porque es el único software que implementa la topología toroidal para el mapa SOM permitiendo solucionar el efecto borde durante el entrenamiento [50].

Las topologías cíclicas (toroidal, esférica o cilíndrica) son formas especiales de representar la matriz en donde no existen los límites y por lo tanto, son una solución a los

problemas de distorsiones y discontinuidades que suelen ocurrir en los bordes de una matriz con topología de hoja (*sheet*). Las diferencias entre las distintas topologías se pueden ver en la **Figura 36**.

Figura 36

Ejemplos topologías sheet, cylinder y toroid para los mapas auto-organizados



Nota. Tomado de "Slideshare a Scribd company", Curso 2006 Sesion 1 Kohonen Presentation, 2006, https://es.slideshare.net/askroll/curso-2006-sesion-1-kohonen-presentation

4.2.9.1. Topología toroidal

La versión del *dataset* utilizado para esta prueba (ver **Tabla 9**) es el mismo que se utilizó en Python (ver **Anexo 2**). El objetivo no era encontrar los mapas finales con los cuales seguir trabajando sino analizar los resultados entre una topología cíclica y la comúnmente usada, *sheet*. Es por esta razón que no se hizo hincapié en la elección de los parámetros y se utilizaron los sugeridos por Haykin S [45]. Los resultados son ilustrados en las **Figuras 37** y **38**.

Figura 37

Matriz-U resultante de utilizar topología sheet

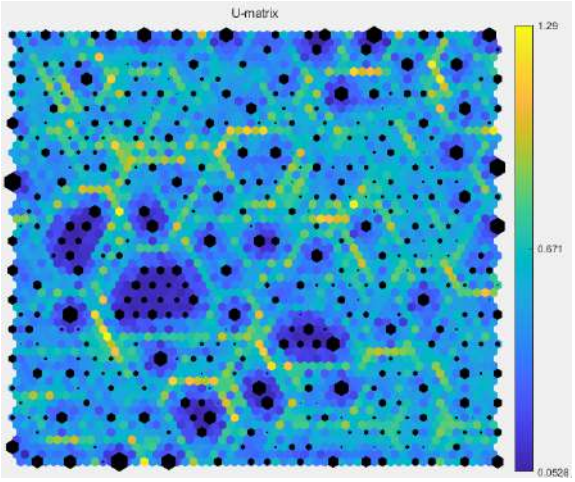
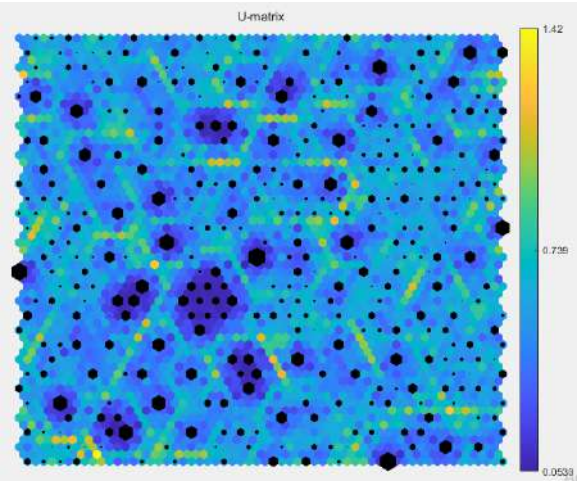


Figura 38

Matriz-U resultante de utilizar topología toroidal



Nota. Los *hits* de cada neurona se representan con un hexágono negro y denotan la cantidad de veces que dicha neurona fue BMU. Cuantas más veces haya sido BMU una neurona, mayor será el tamaño del hexágono negro en su interior.

Resulta evidente la diferencia en los límites de ambas imágenes y cómo una topología toroidal disminuye el denominado *boundary effect*. Si bien con esto se logró un gran avance en el proyecto, en las imágenes se observa que los resultados podrían mejorar aún más puesto que era complicado poder visualizar *clusters* definidos con los que continuar.

4.2.9.2. Agrupación con distintos atributos

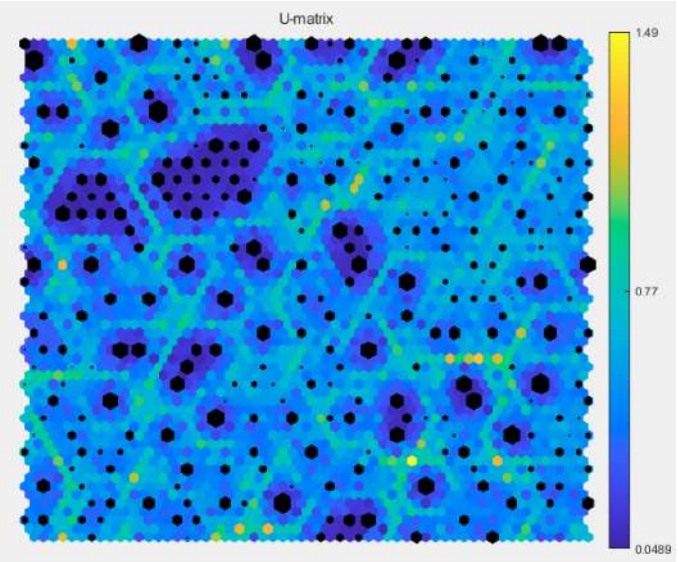
El solo uso de la topología toroidal no era suficiente para obtener resultados de calidad, por lo que se hizo evidente la posibilidad de que la naturaleza de los datos no permitieran que estos pudieran agruparse. Por lo tanto, se realizaron pruebas variando los atributos que eran tenidos en cuenta para la agrupación y ejecución del SOM y así descartar o no esta idea. Para la elección de los parámetros se utilizó los sugeridos por Haykin S [45].

A continuación se enumeran los resultados obtenidos **sin tener en cuenta** los siguientes atributos para el ordenamiento del SOM con sus respectivas medidas de calidad:

1) Atributo: **origen_financiamiento**, QE: 0,4652 y TE: 0,0233 (ver **Figura 39)**.

Figura 39

Matriz-U resultante de procesar el dataset sin tener en cuenta el atributo origen_financiamiento para realizar el agrupamiento

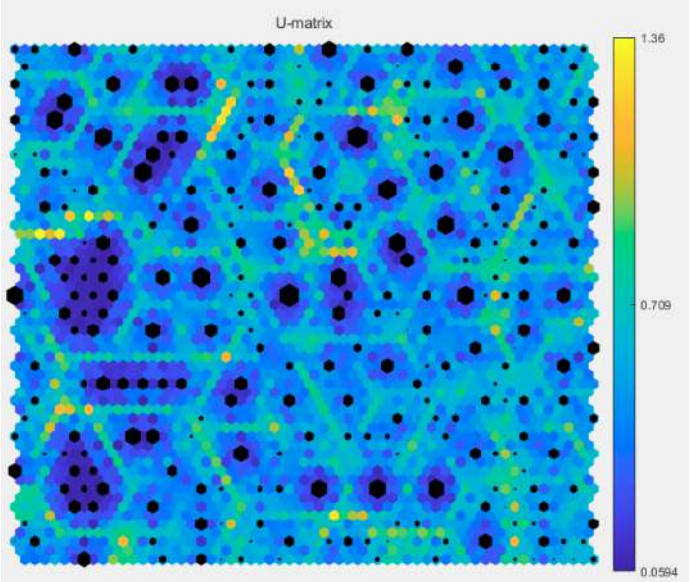


Nota. Los *hits* de cada neurona se representan con un hexágono negro y denotan la cantidad de veces que dicha neurona fue BMU. Cuantas más veces haya sido BMU una neurona, mayor será el tamaño del hexágono negro en su interior.

2) Atributo: **clasificacion_resumen**, QE: 0,4423, TE: 0,0255 (ver **Figura 40**).

Figura 40

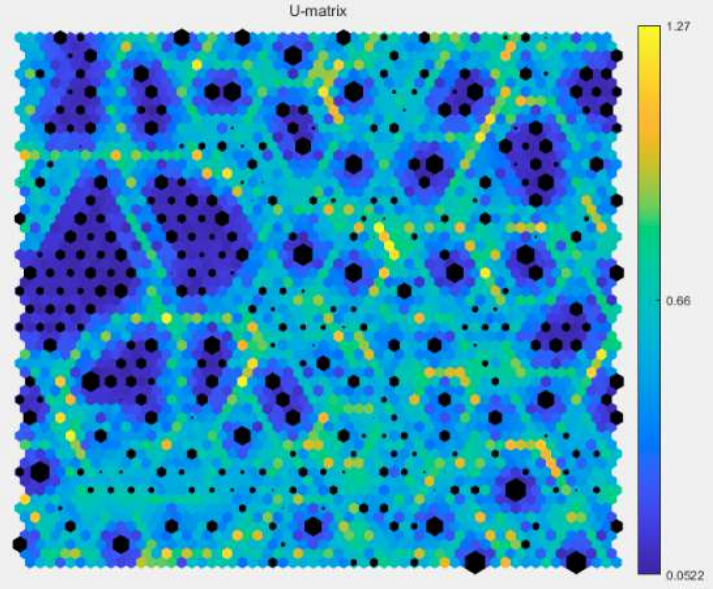
Matriz-U resultante de procesar el dataset sin tener en cuenta el atributo clasificacion_resumen para realizar el agrupamiento



3) Atributo: **regiones**, QE: 0,3361, TE: 0,037 (ver **Figura 41**).

Figura 41

Matriz-U resultante de procesar el dataset sin tener en cuenta el atributo regiones para realizar el agrupamiento

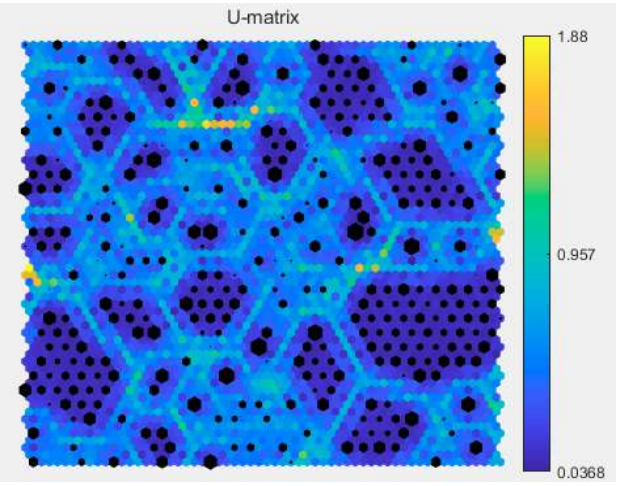


En este punto se observa que al menos quitando uno de estos atributos el agrupamiento mejora no sólo en las medidas de calidad sino que también visualmente. Se continuó con las pruebas pero esta vez quitando más de un atributo por vez.

4) Atributos: **regiones** y **origen_financiamiento**, QE: 0,244, TE: 0,0206 (ver **Figura 42**).

Figura 42

Matriz-U resultante de procesar el dataset sin tener en cuenta los atributos regiones y origen_financiamiento para realizar el agrupamiento



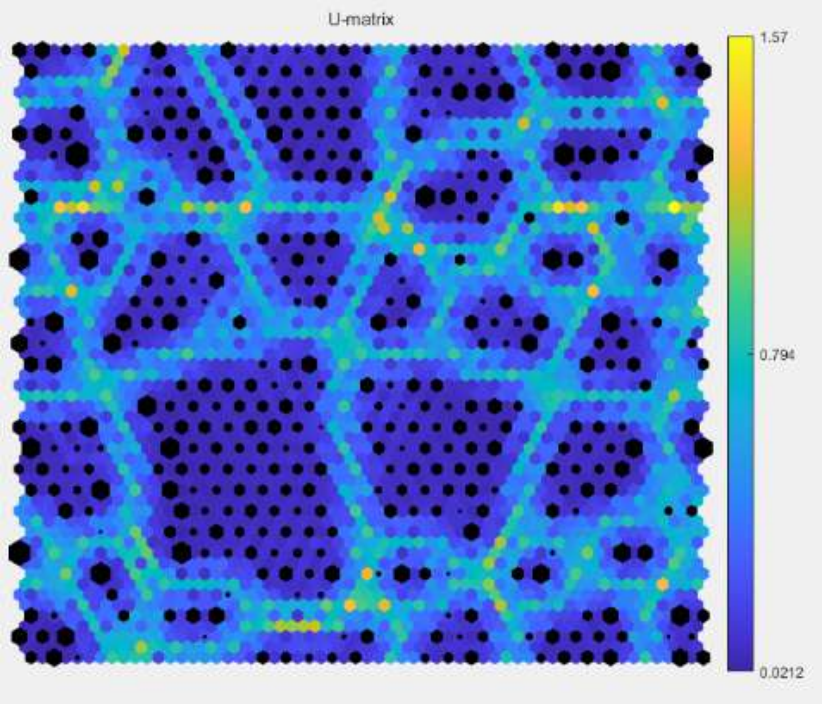
El resultado anterior (**Figura 42**) fue uno de los mejores obtenidos hasta el momento. Por lo tanto, con esto se puede concluir que el problema de no poder alcanzar resultados aceptables estaba muy influenciado por las características de los datos.

4.2.9.3. Base de datos pública y datos del HPC

No fue posible obtener de la base de datos del HPC todos los atributos que se estaban analizando hasta el momento. En consecuencia, se adaptó el dataset de la base de datos pública del Ministerio de Salud para que coincida con el del hospital (ver sección 4.2.10.1) y así se puedan analizar ambos al mismo nivel. Se realizó un entrenamiento de SOM a modo de prueba con esta nueva versión y el resultado es el siguiente:

Figura 43

Matriz-U resultante de la ejecución del SOM con la nueva versión del dataset del Ministerio de Salud que cuenta con los mismos atributos que el dataset del HPC



El error de cuantización de la **Figura 43** es 0,1207 y el error topográfico es de 0,0289. Comparado a los resultados de las figuras anteriores se puede apreciar una mejora: los

clusters son más definidos, más grandes, en menor cantidad y con mejores medidas de calidad.

Finalmente, teniendo en cuenta la mejora lograda con la topología toroidal y el progreso obtenido mediante el cambio de los atributos utilizados para el ordenamiento, se siguió trabajando con MATLAB© para las ejecuciones finales del SOM.

4.2.10. Selección de *datasets* finales

En el transcurso del desarrollo del proyecto se utilizaron distintas versiones de la base de datos de COVID-19 del Ministerio de Salud (detalladas y justificadas en los apartados de selección, limpieza y transformación) conforme se analizaban los resultados de las distintas pruebas. Para el entrenamiento de las versiones finales de SOM y evaluación e interpretación de resultados se seleccionaron 4 versiones potenciales a ser útiles en el descubrimiento de conocimiento (además del *dataset* obtenido del HPC).

4.2.10.1 *Dataset* 1

Este *dataset* corresponde a personas **fallecidas** que residían en la provincia de **Buenos Aires** y fueron atendidas en instituciones de origen de financiamiento **privado**. Las personas fueron internadas y tratadas en una Unidad de Cuidados Intensivos. Estos datos se extrajeron del *dataset* público proporcionado por el Ministerio de Salud. Todos sus atributos se tuvieron en cuenta para el ordenamiento del SOM (ver **Tabla 10**). En total se encontraron 4150 casos registrados desde el 31 de enero de 2020 hasta el 01 de mayo de 2022. Se buscó que los atributos coincidan con los que se pudieron obtener del HPC (*dataset* 2). El *dataset* se compone por los siguientes atributos:

Tabla 10

Descripción atributos dataset 1

Atributo	Tipo	Descripción
edad	Numérico	Edad de la persona al inicio de síntomas
dias_internacion_cui_intensivo	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos
dias_cui_intensivo_fallec	Numérico	Días transcurridos desde que la persona entró a cuidados intensivos hasta que falleció
sexo	Categórico	Femenino/Masculino
asistencia_respiratoria_mecanica	Categórico	Toma dos posibles valores (SI/NO) dependiendo si la persona utilizó o no asistencia respiratoria mecánica
estacion	Categórico	Estación del año al momento de inicio de síntomas
año	Categórico	Año al momento de inicio de síntomas

4.2.10.2 *Dataset* 2

Este *dataset* corresponde a personas **fallecidas** que residían en la ciudad de **Mar del Plata** y fueron atendidas en el **HPC** (institución de origen de financiamiento **privado**). Las personas fueron internadas y tratadas en una Unidad de Cuidados Intensivos. Estos datos fueron proporcionados por el propio hospital. Todos sus atributos se tuvieron en cuenta para el ordenamiento del SOM. En total se encontraron 631 casos registrados desde marzo de 2020 hasta marzo de 2022. El *dataset* se compone por los mismos atributos que el *dataset* 1 (ver **Tabla 11**):

Tabla 11

Descripción atributos dataset 2

Atributo	Tipo	Descripción
edad	Numérico	Edad de la persona al inicio de síntomas
dias_internacion_cui_intensivo	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos
dias_cui_intensivo_fallec	Numérico	Días transcurridos desde que la persona entró a cuidados intensivos hasta que falleció
sexo	Categórico	Femenino/Masculino
asistencia_respiratoria_mecanica	Categórico	Toma dos posibles valores (SI/NO) dependiendo si la persona utilizó o no asistencia respiratoria mecánica
estacion	Categórico	Estación del año al momento de inicio de síntomas
año	Categórico	Año al momento de inicio de síntomas

4.2.10.3 *Dataset* 3

Este *dataset* es idéntico al 1. La única diferencia es que no se tuvo en cuenta el atributo **año** para el entrenamiento del SOM. Se observó que la eliminación de este atributo producía un mapa con una de las mejores medidas de calidad y una Matriz-U que reflejaba *clusters* bien delimitados. Más adelante se verá en mayor detalle.

4.2.10.4 *Dataset* 4

Este *dataset* corresponde a personas **fallecidas** que residían en la provincia de **Buenos Aires** y fueron atendidas en instituciones de origen de financiamiento **público** (a diferencia del *dataset* 1, donde solo se tienen en cuenta datos vinculados a financiamiento privado). Las personas fueron internadas y tratadas en una Unidad de Cuidados Intensivos. Estos datos

fueron extraídos del *dataset* público proporcionado por el Ministerio de Salud. Todos sus atributos se tuvieron en cuenta para el ordenamiento del SOM. En total se encontraron 4767 casos registrados desde el 31 de enero de 2020 hasta el 01 de mayo de 2022. El *dataset* se compone por los siguientes atributos listados en la **Tabla 12** (mismos atributos que el *dataset* 1):

Tabla 12

Descripción atributos dataset 4

Atributo	Tipo	Descripción
edad	Numérico	Edad de la persona al inicio de síntomas
dias_internacion_cui_intensivo	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos
dias_cui_intensivo_fallec	Numérico	Días transcurridos desde que la persona entró a cuidados intensivos hasta que falleció
sexo	Categórico	Femenino/Masculino
asistencia_respiratoria_mecanica	Categórico	Toma dos posibles valores (SI/NO) dependiendo si la persona utilizó o no asistencia respiratoria mecánica
estacion	Categórico	Estación del año al momento de inicio de síntomas
año	Categórico	Año al momento de inicio de síntomas

4.2.10.5 *Dataset* 5

Este *dataset* corresponde a personas **fallecidas** que residían en **Argentina** y fueron atendidas en instituciones de origen de financiamiento tanto **público** como **privado**. Las personas fueron internadas y tratadas en una Unidad de Cuidados Intensivos. Estos datos fueron extraídos del *dataset* público proporcionado por el Ministerio de Salud. No se tuvieron en cuenta para el ordenamiento del SOM los atributos **año** y **provincia**. En total se encontraron 27787 casos registrados desde el 31 de enero de 2020 hasta el 01 de mayo de 2022. El *dataset* está compuesto por los siguientes atributos (ver **Tabla 13**):

Tabla 13

Descripción atributos dataset 5

Atributo	Tipo	Descripción
edad	Numérico	Edad de la persona al inicio de síntomas
dias_sintomas_internacion	Numérico	Días transcurridos desde que la persona presentó síntomas hasta que fue internada
dias_internacion_cui_intensivo	Numérico	Días transcurridos desde que la persona fue internada hasta que pasó a cuidados intensivos
dias_cui_intensivo_fallec	Numérico	Días transcurridos desde que la persona entró a cuidados intensivos hasta que falleció
sexo	Categórico	Femenino/Masculino
residencia_provincia_nombre	Categórico	Provincia de residencia de la persona
asistencia_respiratoria_mecanica	Categórico	Toma dos posibles valores (SI/NO) dependiendo si la persona utilizó o no asistencia respiratoria mecánica
origen_financiamiento	Categórico	Toma dos posibles valores (PUBLICO/PRIVADO) dependiendo si la institución donde fue atendido el paciente en cuestión es privada o pública
estacion	Categórico	Estación del año al momento de inicio de síntomas

4.2.10.6. Tabla resumen

A continuación en la **Tabla 14** se muestra a modo de resumen los 5 *datasets* descritos anteriormente:

Tabla 14

Resumen de las características de los 5 datasets seleccionados

	Dataset 1	*Dataset* 2	*Dataset* 3	*Dataset* 4	*Dataset* 5
Cobertura	Prov. Buenos Aires	Mar del Plata (HPC)	Prov. Buenos Aires	Prov. Buenos Aires	Argentina
Origen Financiamiento	Privado	Privado	Privado	Público	Ambos
Cantidad de casos	4150	631	4150	4767	27787
Comentarios	-	-	No tiene en cuenta año para el ordenamiento	-	No tiene en cuenta provincia y año para el ordenamiento

4.2.11. Automatización para elección de parámetros

Se desarrolló un proceso automático en MATLAB que varía los parámetros de acuerdo a un rango específico. El objetivo de este procedimiento es analizar las medidas de calidad (QE, TE) de los resultados y, teniendo en cuenta la matriz-U, elegir el más conveniente. Los parámetros involucrados en el proceso son:

- **Tamaño del mapa**: teniendo en cuenta la cantidad de casos en los *datasets* definidos el tamaño mínimo utilizado fue de 10x10. El tamaño máximo fue de 100x100 aunque no en todos los casos se llegó a este límite ya que dependía del tamaño del *dataset*. El rango se hizo variar de 5 en 5 en cada eje hasta llegar al máximo especificado, como por ejemplo: 10x10, 10x15, 15x15, 15x20, 20x20, …, 95x95, 100x100.

- **Radio de vecindad inicial en la etapa de ordenamiento**: se probaron 3 tipos de funciones para el cálculo del radio en la etapa de entrenamiento según distintos enfoques [45, 50] (ver **Ecuaciones 37**, **38** y **39**):

  - $\sqrt{X^{2} + Y^{2}}$ **(Ec. 37)**
  - $\frac{máx\,(X, Y)}{2}$ **(Ec. 38)**
  - $máx(X, Y) * 0,2$ **(Ec. 39)**

  Siendo X e Y el tamaño de la grilla en el eje “x” e “y” respectivamente y $máx(X, Y)$ el valor máximo entre ambos valores.

- **Iteraciones en la etapa de ordenamiento**: se probó con 3 valores diferentes: 1000, 5000 y 10000.

- **Tasa de aprendizaje inicial en etapa de ordenamiento**: 0,1; 0,3; 0,5; 0,7 y 1.

- **Tasa de aprendizaje inicial en etapa de convergencia**: 0,001; 0,01; 0,05 y 0,1.

Los restantes parámetros permanecían con un valor fijo o se calculaban en base a otros:

- **Radio final en etapa de ordenamiento**: 1.
- **Radio inicial en etapa de convergencia**: 1.
- **Radio final en etapa de convergencia**: 1.
- **Iteraciones en la etapa de convergencia**: $X * Y * 500$.
- **Topología**: toroidal y hexagonal.
- **Función de aprendizaje**: de tipo exponencial utilizada en la librería.

El pseudocódigo del proceso es el siguiente:

```
Inicio
        datos = carga_de_datos()
        tamaño_mapa = [10; 10; 10; 15; … ; N; N; N; M; M; M]
        iteraciones_iniciales = [1000; 5000; 10000]
        tasa_aprend_i_ord = [0,1; 0,3; 0,5; 0,7; 1]
        tasa_aprend_i_conv = [0,001; 0,01; 0,05; 0,1]
        radio_f_ord = 1
        radio_i_conv = 1
        radio_f_conv = 1
        topología = "toroidal - hexagonal"

        Desde i=1 de a pasos de a 2 hasta longitud(tamaño_mapa)

                radio_i_ord = [sqrt( tamaño_mapa[i] ^ 2, tamaño_mapa[i+1] ^ 2); máx(tamaño_mapa[i], tamaño_mapa[i+1])/2; máx(tamaño_mapa[i], tamaño_mapa[i+1]) * 0,2]

                Desde j=1 hasta longitud(radio_i_ord)

                        Desde k=1 hasta longitud(iteraciones_iniciales)

                                Desde p=1 hasta longitud(tasa_aprend_i_ord)

                                        Desde q=1 hasta longitud(tasa_aprend_i_conv)

                                                sm = SOM_inicialización(datos, tamaño_mapa[i], tamaño_mapa[i+1], topología)
                                                sm = SOM_ordenamiento(sm, datos, radio_i_ord[j], radio_f_ord, iteraciones_iniciales[k], tasa_aprend_i_ord[p])
                                                iteraciones_finales = tamaño_mapa[i], tamaño_mapa[i+1] * 500
                                                sm = SOM_convergencia(sm, datos, radio_i_conv, radio_f_conv, iteraciones_finales, tasa_aprend_i_conv[q])
                                                qe = calculo_qe(sm)
                                                te = calculo_te(sm)
                                                guardar_resultados()

                                        Siguiente

                                Siguiente

                        Siguiente

                Siguiente

        Siguiente
Fin
```

4.2.12. Fase de data mining: combinación de SOM con k-means

4.2.12.1. *Dataset* 1

Luego de las pruebas realizadas para la obtención de un mapa y una agrupación de calidad para el *dataset* 1 se optó por un mapa con las siguientes características y parámetros (**Tabla 15**):

Tabla 15

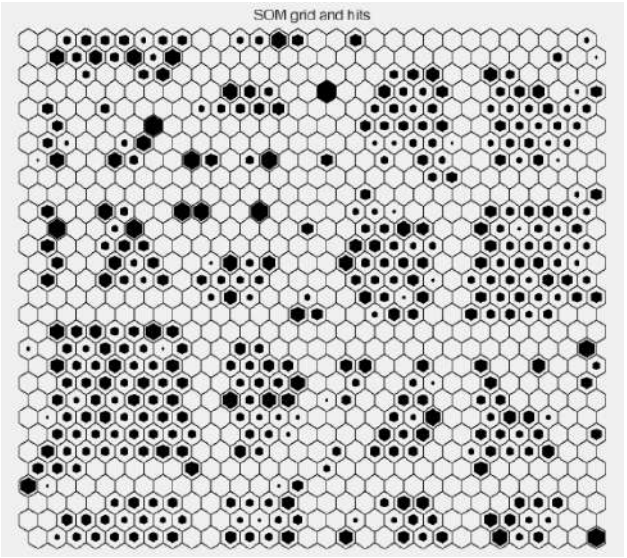
Parámetros para el entrenamiento del SOM del dataset 1 junto a las medidas de calidad resultantes (QE y TE)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	30	30
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,1	0,05
Cantidad de iteraciones (T)	5000	450000
Radio de vecindad inicial (σ_0)	6	1
Radio de vecindad final(σ_f)	1	1
Topología	Hexagonal, Toroide	Hexagonal, Toroide
Función de aprendizaje	Tipo exponencial	Tipo exponencial
QE final	-	0,1194
TE final	-	0,0106

A continuación se muestra la grilla correspondiente al SOM con los *hits* resultantes de presentar los puntos de datos del *dataset* 1 a la red (ver **Figura 44**):

Figura 44

Grilla SOM con hits

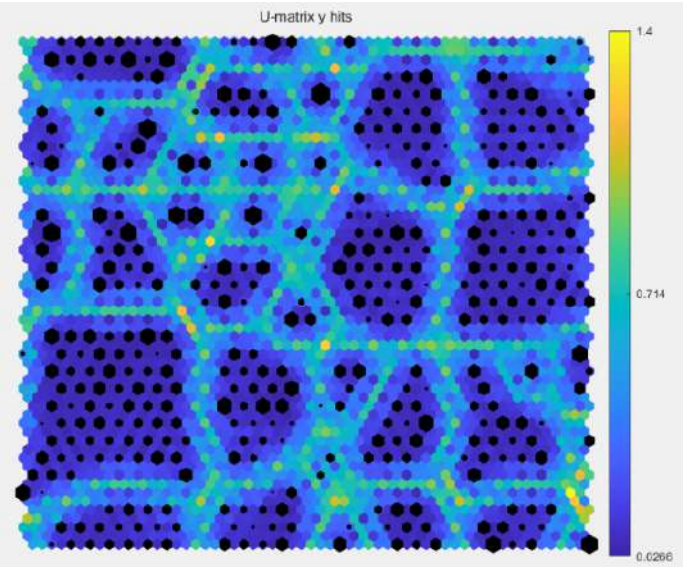


Nota. Los *hits* de cada neurona se representan con un hexágono negro y denotan la cantidad de veces que dicha neurona fue BMU. Cuantas más veces haya sido BMU una neurona, mayor será el tamaño del hexágono negro en su interior.

Observando la imágen anterior, puede notarse posibles agrupaciones. Para poder ver mejor los resultados se utilizó la matriz-U junto con los *hits* correspondientes (**Figura 45**):

Figura 45

Matriz-U con hits resultante del SOM con dataset 1

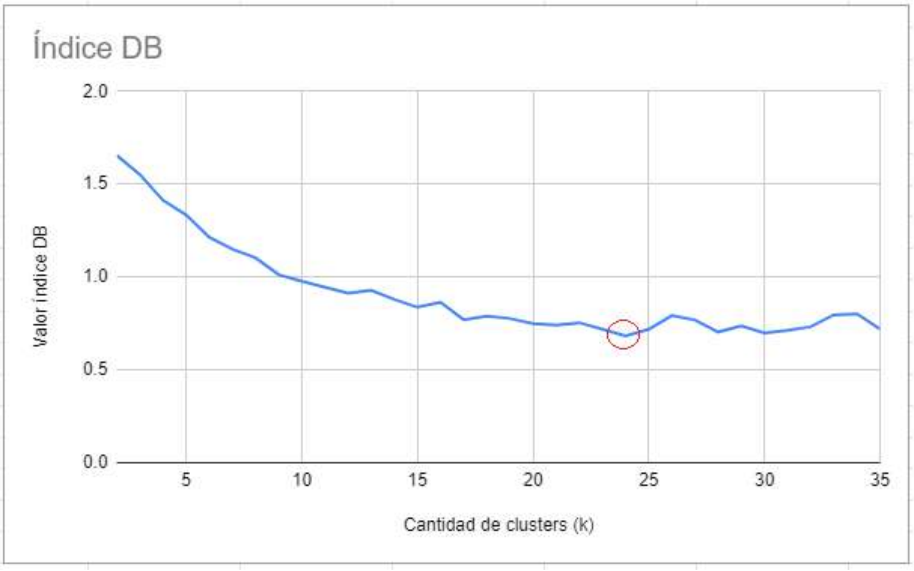


Con esta imagen, gracias a la visualización de las distancias más grandes en amarillo, se pueden notar fronteras que permiten delimitar las diferentes agrupaciones que resultaron. Sin embargo, se pueden notar algunas muy pequeñas, es decir, que están compuestas por pocas neuronas y también en algunos casos que tienen pocos *hits*. Por lo tanto, se decidió realizar un k-means sobre las neuronas del mapa para poder hacer un refinamiento y ver la posibilidad de juntar uno o más conglomerados pequeños con otros [11].

K-means ayuda a tomar la decisión de en dónde es más adecuado unir un *cluster* con otro. Se ejecutó dicho algoritmo variando *k* en un rango entre 2 y un número lo suficientemente grande (mayor a la cantidad que se podrían observar en la matriz-U) para encontrar el *k* que tenga el menor índice de Davies-Bouldin [20] (ver **Figura 46**).

Figura 46

Gráfico que muestra el valor que toma el índice DB en cada número de cluster (k)



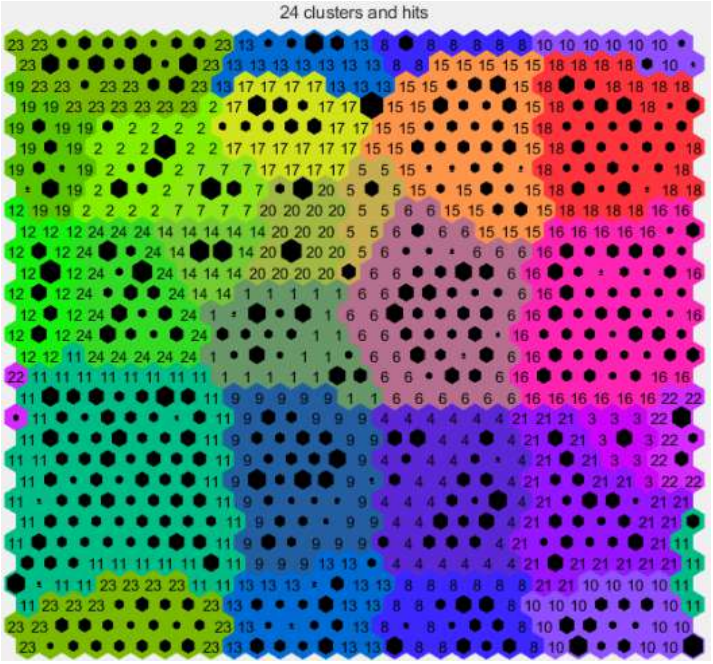
Como se ve en el gráfico, el *k* con menor índice de DB fue 24 por lo que este valor fue el elegido para procesar las neuronas del SOM con k-means y poder obtener los *clusters* finales.

También, con esta técnica se pudo realizar un etiquetado de cada neurona para tener información de a qué *cluster* pertenece cada una y así, luego poder etiquetar los puntos de datos al *cluster* que pertenecen dependiendo de cuál es la BMU de cada uno.

Por último, el resultado final está presentado en la siguiente figura (**Figura 47**):

Figura 47

Clusters finales (24) encontrados para el dataset 1

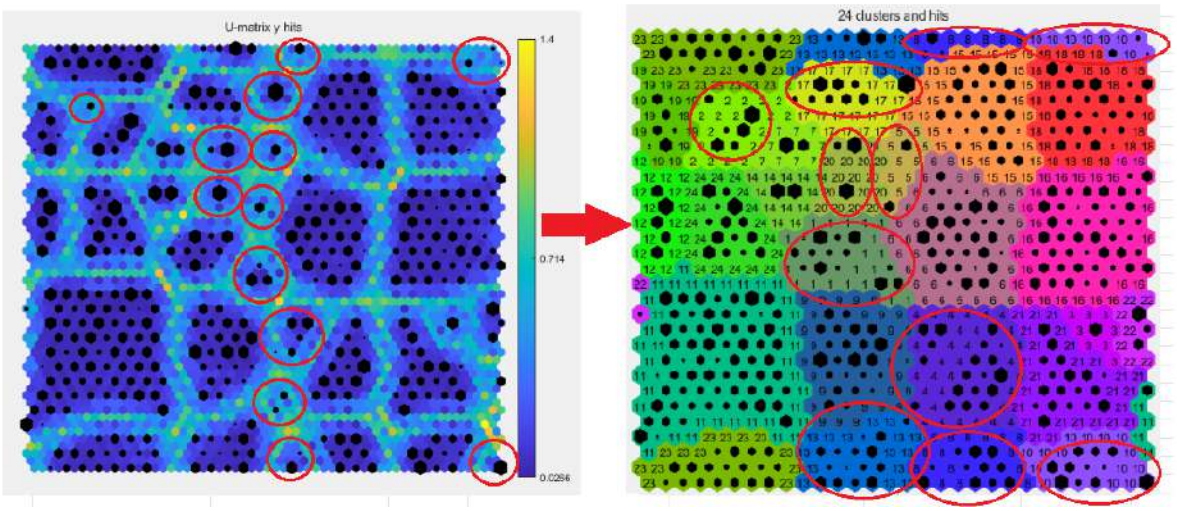


Nota. Los *hits* de cada neurona se representan con un hexágono negro y denotan la cantidad de veces que dicha neurona fue BMU. El número (y color) de cada una indica la pertenencia a un determinado *cluster*.

Se puede notar como algunos grupos, que se podían ver como únicos en la matriz-U, luego de aplicar k-means se unieron a otros. Esto está plasmado en la imagen siguiente (ver **Figura 48**), donde a modo de ejemplo se marcaron pequeños grupos que fueron fusionados con otros:

Figura 48

Proceso de unión de clusters en otros más grandes luego de aplicar k-means para dataset 1



Si bien k-means tiene la desventaja de generar agrupaciones de tamaño similar, en este caso resultó ventajosa la unión de ciertos grupos con otros. Esto es porque no se podría considerar como un *cluster* único a aquel compuesto por muy pocos datos. Aunque se tiene que tener en cuenta que esto podría ser discutible, porque si en un futuro se presentara a la red este mismo *dataset* pero con más tuplas (actualizado en el tiempo) quizás no quedarían *clusters* con pocos datos y no sería necesario realizar fusiones de unos con otros.

4.2.12.2. *Dataset* 2

Los parámetros y las características del mapa elegido para procesar el *dataset* 2 son los siguientes (ver **Tabla 16**):

Tabla 16

Parámetros para el entrenamiento del SOM del dataset 2 junto a las medidas de calidad resultantes (QE y TE)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	25	25
Tamaño *y*	25	25

Tasa de aprendizaje inicial (α_0)	0,3	0,1
Cantidad de iteraciones (T)	10000	312500
Radio de vecindad inicial (σ_0)	12,5	1
Radio de vecindad final(σ_f)	1	1
Topología	Hexagonal, Toroide	Hexagonal, Toroide
Función de aprendizaje	Tipo exponencial	Tipo exponencial
QE final	-	0,0747
TE final	-	0,0015

La grilla del SOM resultante y la matriz-U con los *hits* se encuentra a continuación (**Figuras 49 y 50**):

Figura 49

Grilla SOM con hits resultante dataset 2

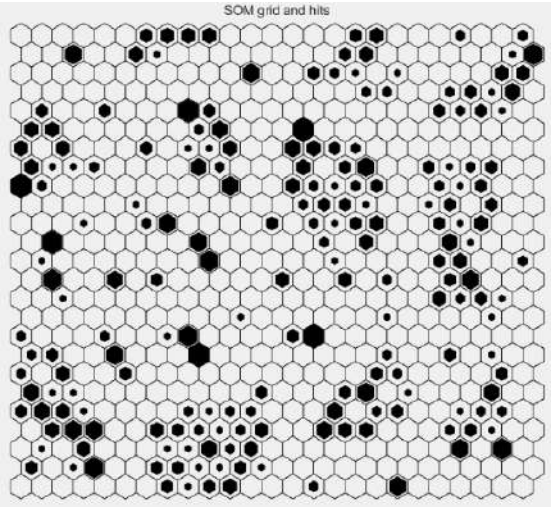
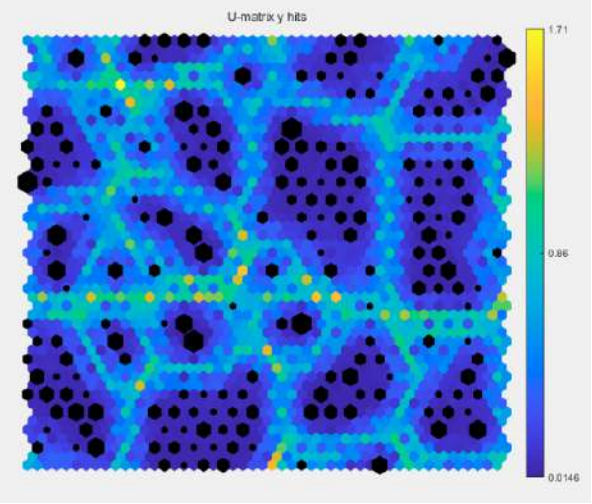


Figura 50

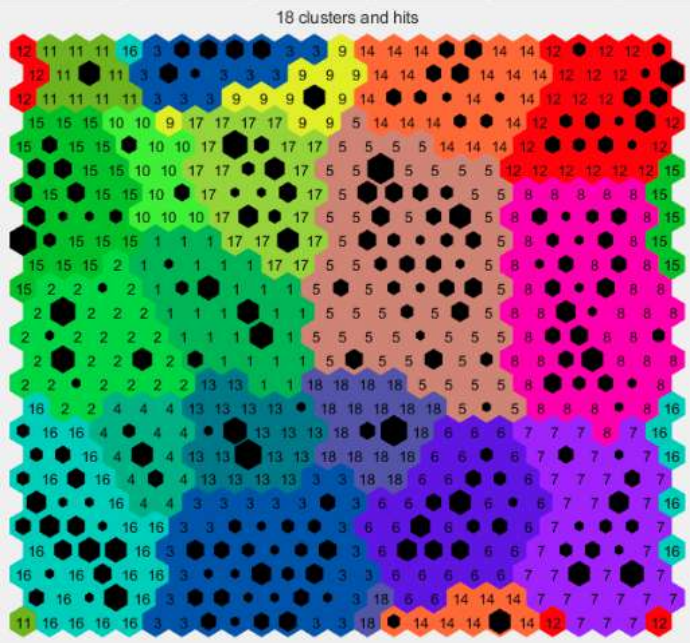
Matriz-U con hits resultante dataset 2



Al igual que en el *dataset* anterior hay presencia de grupos pequeños con pocos *hits*. Por lo tanto, se realizó un refinamiento aplicando k-means sobre las neuronas del SOM variando los *k* hasta encontrar el que tenga el menor índice de DB. Como resultado se obtuvo un $k = 18$ con un índice de 0.666. También, k-means cumplió la función de etiquetado (ver **Figura 51**).

Figura 51

Clusters finales (18) encontrados para el dataset 2



Tanto en el *dataset* anterior como en este y en los siguientes que se presentarán se puede ver que al momento de realizar las agrupaciones, los atributos categóricos son los más influyentes.

4.2.12.3. *Dataset* 3

Los parámetros y las características del mapa elegido para procesar el *dataset* 3 son los siguientes (ver **Tabla 17**):

Tabla 17

Parámetros para el entrenamiento del SOM del dataset 3 junto a las medidas de calidad resultantes (QE y TE)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	30	30
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,7	0,05
Cantidad de iteraciones (*T*)	10000	450000

Radio de vecindad inicial (σ_0)	6	1
Radio de vecindad final(σ_f)	1	1
Topología	Hexagonal, Toroide	Hexagonal, Toroide
Función de aprendizaje	Tipo exponencial	Tipo exponencial
QE final	-	0,0877
TE final	-	0,0149

La grilla del SOM resultante y la matriz-u con los *hits* se encuentra a continuación (**Figura 52 y 53**) :

Figura 52

Grilla SOM con hits resultante dataset 3

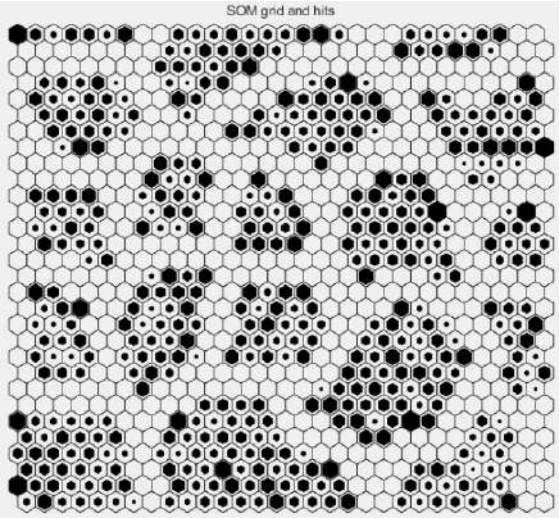
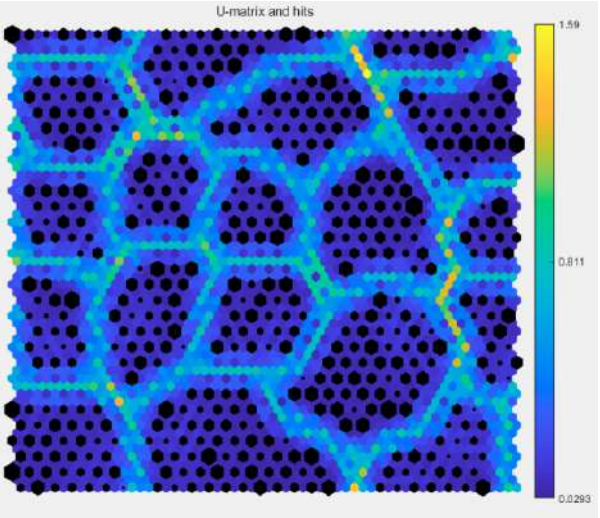


Figura 53

Matriz-U con hits resultante dataset 3

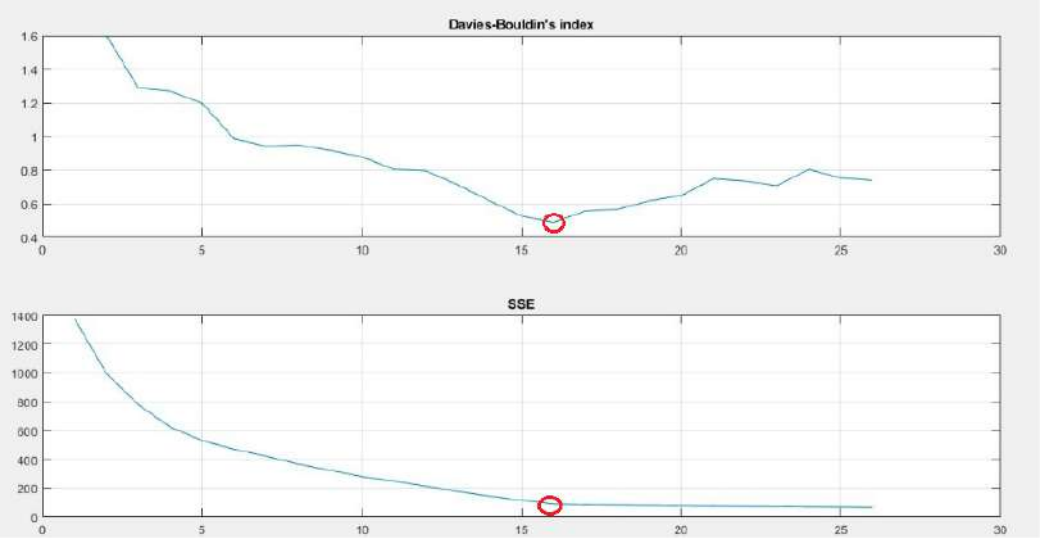


En este caso en particular, se puede notar que el hecho de no tener en cuenta el atributo año para realizar el agrupamiento resultó un mapa de mejor calidad. No hay presencia de grupos pequeños y los *clusters* están bien delimitados.

Al aplicar k-means no se produjo una mejora en la agrupación por lo que sólo cumplió función de etiquetado para las neuronas y los puntos de datos. Se utilizó un $k = 16$ por los resultados arrojados en los gráficos a continuación (**Figura 54**):

Figura 54

Gráfico que muestra el valor que toma el índice DB (arriba) y WCSS (abajo) para cada cantidad de clusters (k)



Si se analiza el resultado de aplicar k-means sobre las neuronas se puede notar que el algoritmo formó los *clusters* de la misma manera que uno intuitivamente podría hacerlo sólo observando la matriz-U (ver **Figuras 55 y 56**).

Figura 55

Matriz-U con hits resultante dataset 3

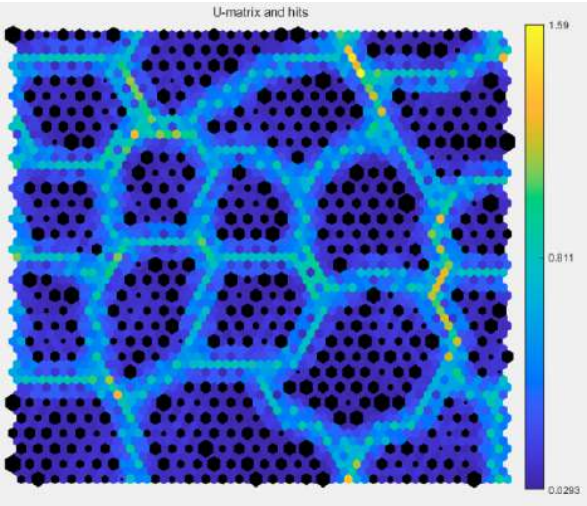
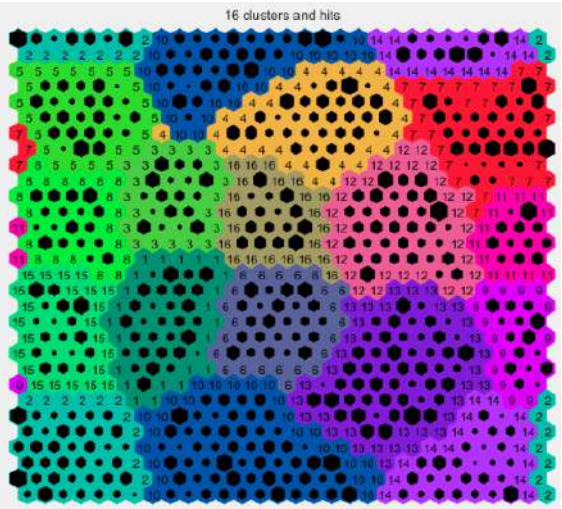


Figura 56

Clusters finales (16) encontrados para el dataset 3



4.2.12.4. *Dataset* 4

Los parámetros y las características del mapa elegido para procesar el *dataset* 4 son los siguientes (**Tabla 18**):

Tabla 18

Parámetros para el entrenamiento del SOM del dataset 4 junto a las medidas de calidad resultantes (QE y TE)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	30	30
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,5	0,1
Cantidad de iteraciones (T)	1000	450000
Radio de vecindad inicial (σ_0)	6	1
Radio de vecindad final(σ_f)	1	1
Topología	Hexagonal, Toroide	Hexagonal, Toroide
Función de aprendizaje	Tipo exponencial	Tipo exponencial
QE final	-	0,1222
TE final	-	0,0140

La grilla del SOM resultante y la matriz-u con los *hits* se encuentra a continuación (**Figuras 57** y **58**):

Figura 57

Grilla SOM con hits resultante dataset 4

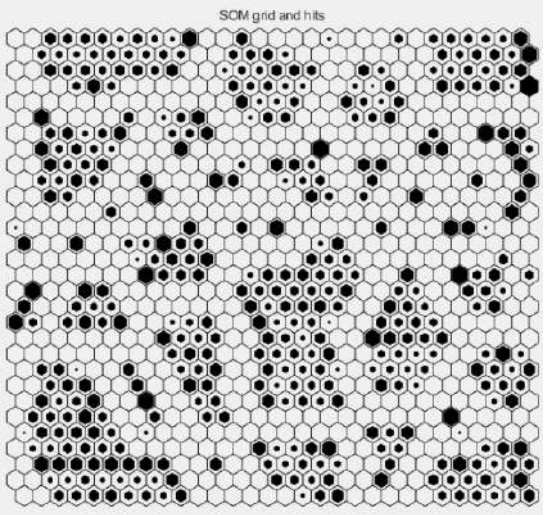
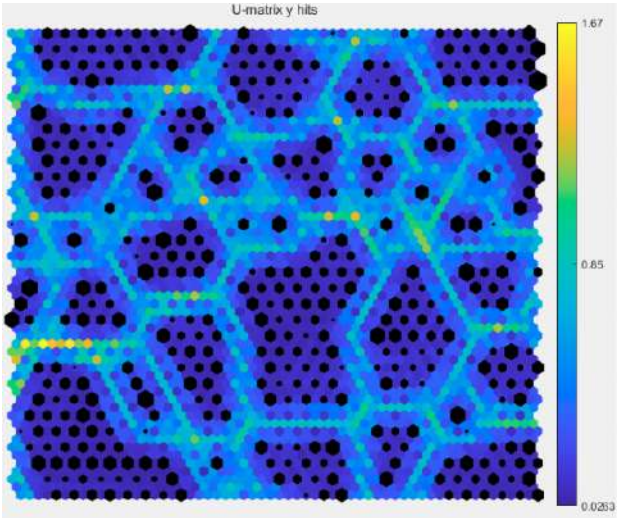


Figura 58

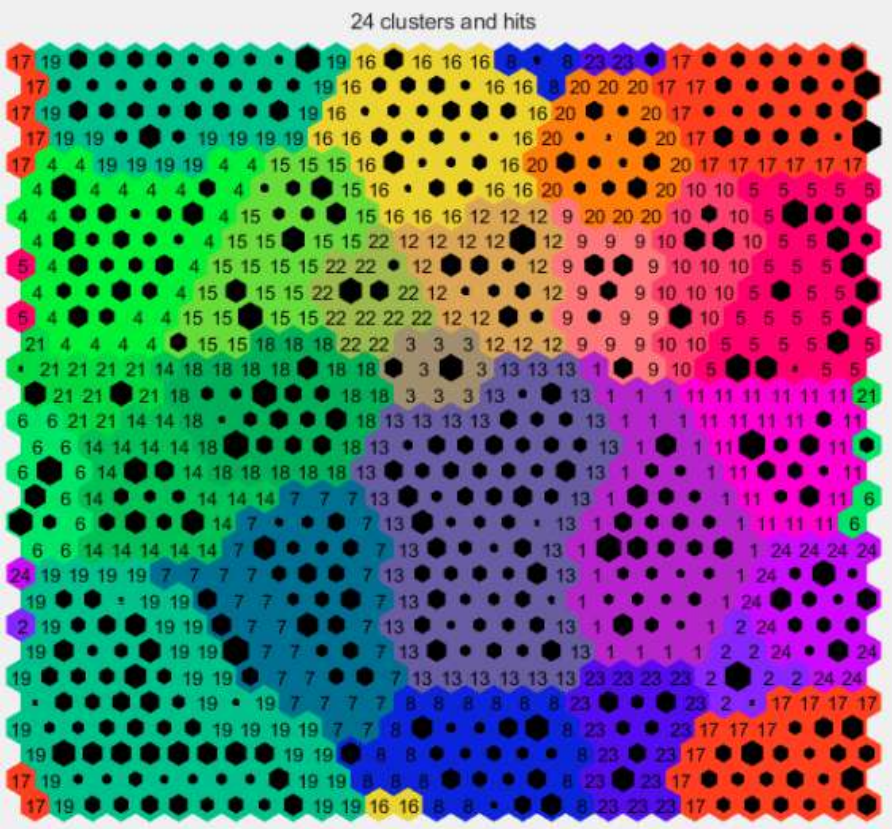
Matriz-U con hits resultante dataset 4



Luego de aplicar k-means para el refinamiento y etiquetado de los *clusters,* el resultado es (**Figura 59**):

Figura 59

Clusters finales (24) encontrados para el dataset 4



4.2.12.5. *Dataset* 5

Los parámetros y las características del mapa elegido para procesar el *dataset* 4 son los siguientes (**Tabla 19**):

Tabla 19

Parámetros para el entrenamiento del SOM del dataset 5 junto a las medidas de calidad resultantes (QE y TE)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	30	30
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,5	0,1
Cantidad de iteraciones (T)	1000	450000
Radio de vecindad inicial (σ_0)	6	1
Radio de vecindad final(σ_f)	1	1
Topología	Hexagonal, Toroide	Hexagonal, Toroide
Función de aprendizaje	Tipo exponencial	Tipo exponencial
QE final	-	0,1642
TE final	-	0,0215

La grilla del SOM resultante y la matriz-u con los *hits* se encuentra a continuación (**Figuras 60** y **61**):

Figura 60

Grilla SOM con hits resultante dataset 5

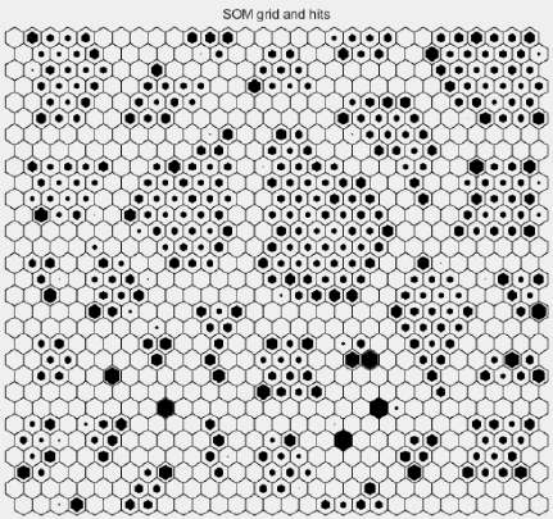
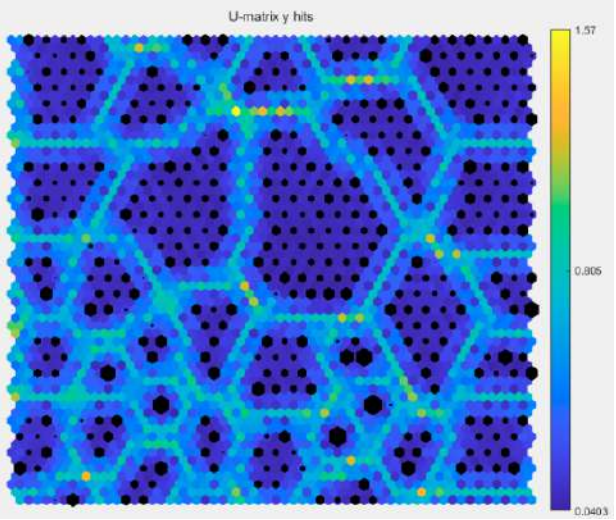


Figura 61

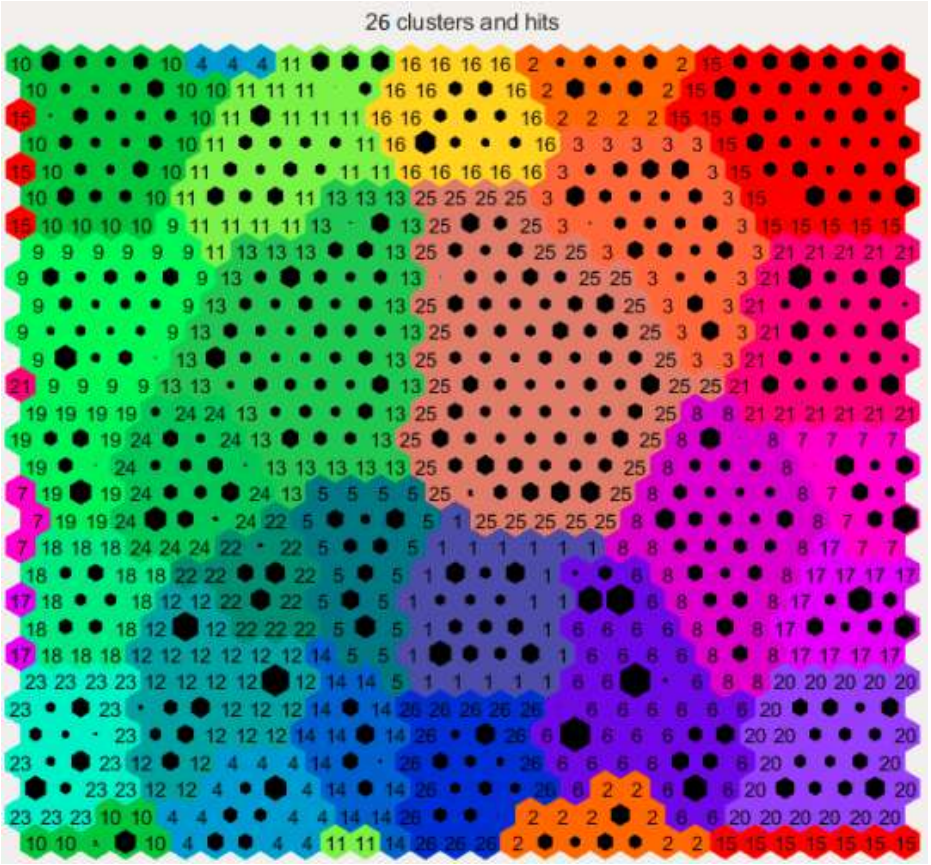
Matriz-u con hits resultante dataset 5



Luego de aplicar k-means para el refinamiento y etiquetado de los *clusters,* el resultado es (**Figura 62**):

Figura 62

Clusters finales (26) encontrados para el dataset 5



4.2.13. Evaluación e interpretación de resultados

4.2.13.1. Métodos de interpretación de clusters

Una de las dificultades encontradas luego de la formación de los agrupamientos en los mapas SOM fue la caracterización de cada *cluster* mediante la definición de una etiqueta o descripción. Si bien esta última está dada por el vector prototipo de la neurona más cercana al centroide, la interpretación resulta compleja y confusa en problemas con muchas variables. Se buscó entonces una representación gráfica de las características de cada *cluster* para facilitar la comprensión de cada uno. Luego de explorar distintos métodos, como puede observarse en el **Anexo 2**, se representó el contenido de cada uno con gráficos estadísticos como el diagrama de caja y bigote para atributos numéricos y gráficos de torta para categóricos (todos estos se encuentran disponibles en el **Anexo 1**).

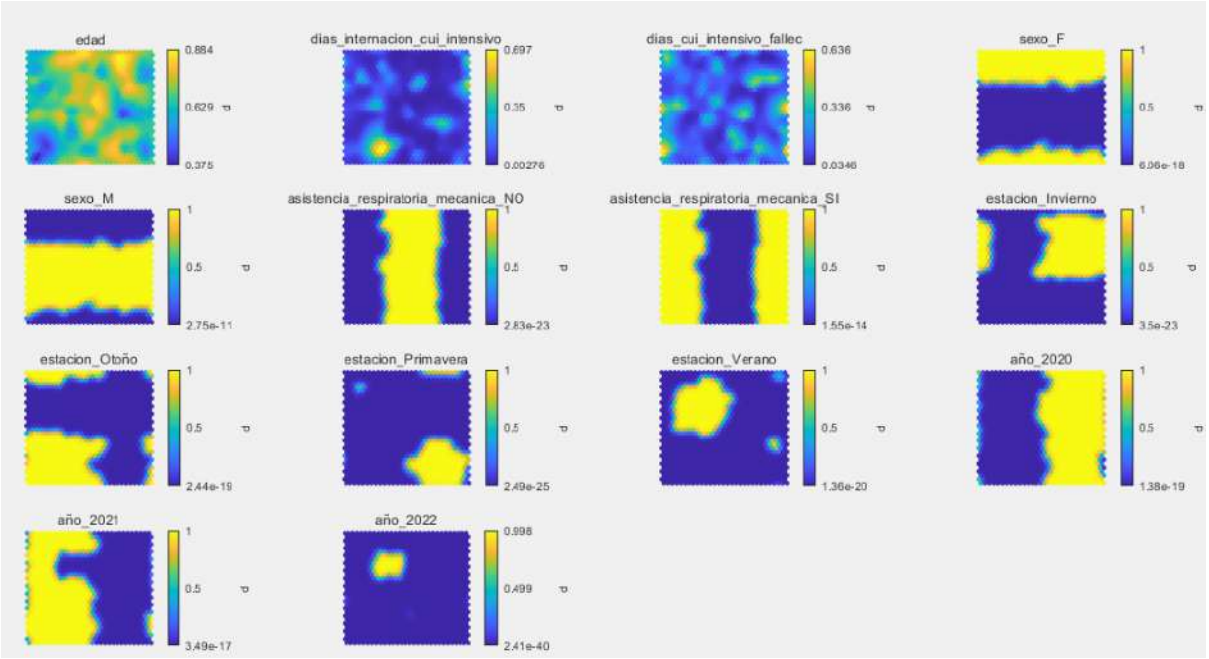
Para poder tener más información de cómo resultó la distribución de los atributos de los datos al ser procesados por el SOM se utilizaron planos de componentes [51]. En dichos planos se muestran los valores que las variables tienen en la estructura del mapa.

4.2.13.2. *Dataset* 1

A continuación se muestra un plano por atributo para el dataset 1 (**Figura 63**):

Figura 63

Planos de componentes de todos los atributos que componen el dataset 1



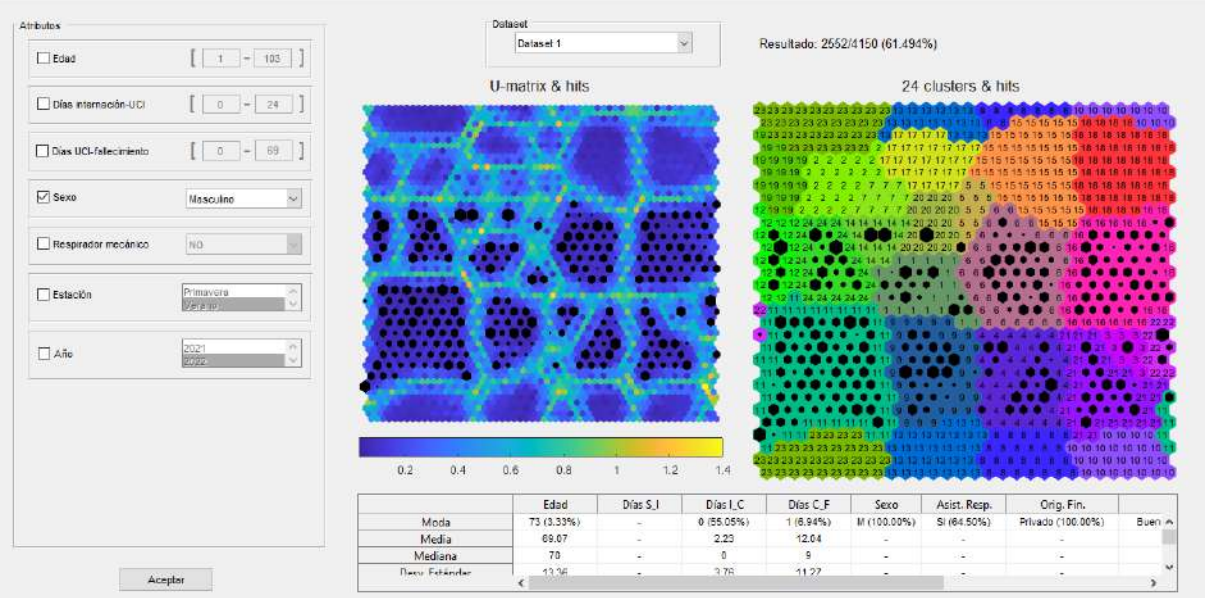
Si bien los atributos están normalizados, las barras laterales muestran valores desde los más bajos (parte inferior azul) hasta los más altos (parte superior amarilla). En el caso de los atributos categóricos, el color amarillo correspondiente al valor 1 significa, por ejemplo observando el atributo **sexo_M**, que esa región del mapa corresponde a pacientes de sexo masculino.

Si se observa detalladamente la distribución de valores de los atributos categóricos, se pueden notar regiones fuertemente delimitadas y casi sin superposición. Esto se puede replicar haciendo uso del software desarrollado durante el trabajo.

Si se hace un filtrado para que sólo se presenten a la red los pacientes del *dataset* 1 de, por ejemplo, sexo masculino se obtiene lo siguiente (**Figura 64**):

Figura 64

Filtro de pacientes masculinos realizado al dataset 1 utilizando el software desarrollado



Esto refleja exactamente lo mismo que en los planos de componentes mostrados anteriormente.

La formación de estas regiones tiene sentido si se recuerda lo que la red conceptualmente representa, ya que esto deja en evidencia una de las características principales de este tipo de red neuronal artificial y es que los grupos que están topográficamente más cerca presentan características similares. Por lo tanto, no sólo los datos que pertenezcan al mismo grupo tienen características similares sino que entre los grupos cercanos ocurre lo mismo.

Como otra conclusión de los *clusters* resultantes se puede mencionar que su formación está fuertemente influenciada por los atributos categóricos más que los numéricos y esto es consecuencia de la binarización. Cuando se realiza la binarización y normalización de las características de los puntos de datos para ser presentados a la red, los atributos numéricos toman valores en el rango entre 0 y 1 mientras que los atributos categóricos toman sólo los valores 0 o 1 (valores extremos) por lo que a la hora de realizar los cálculos

necesarios estos tienen un mayor peso. Como consecuencia de esto, los *clusters* se distinguen entre sí más por atributos categóricos de los puntos de datos que por los numéricos.

Finalmente, analizando los gráficos de cada *cluster* se pueden mencionar algunas conclusiones:

- De los 24 *clusters*, 12 corresponden en su mayoría a pacientes de sexo femenino y los restantes 12 a masculino.
- La mayoría de los *clusters* corresponden a pacientes que hicieron uso de asistencia respiratoria mecánica. Estos *clusters* son: **2**, **3**, **7**, **10**, **11**, **12**, **14**, **16**, **18**, **19**, **21**, **22**, **23**, y **24**.
- De los 6 *clusters* que corresponden a estación invierno, 4 tienen como característica que los pacientes hicieron uso de respirador. Estos *clusters* son: **6**, **12**, **15**, **16**, **18** y **19**.
- El ***cluster* 11** se destaca por ser el que tiene mayor número de casos (693) y además, por ser el que tiene mayor valor de media y mediana en el atributo **dias_internacion_cui_intensivo**. Este *cluster* corresponde a pacientes masculinos que utilizaron respirador mecánico en otoño del año 2021.
- Los *clusters* que tienen al menos el 75% de pacientes con 0 días de internación no hicieron uso de respirador. Todos estos *clusters*, excepto 1 que corresponde al año 2020 (***cluster* 17**), están asociados al año 2021. Estos *clusters* son: **4**, **5**, **6**, **8**, **15** y **17**.
- Se observa que **11 *clusters***, en más del 95.7% de sus casos, son correspondientes al año 2020. 10 grupos corresponden al 2021 (también con un mínimo del 99.6%). Finalmente, los 3 restantes (con un 100%) del 2022.

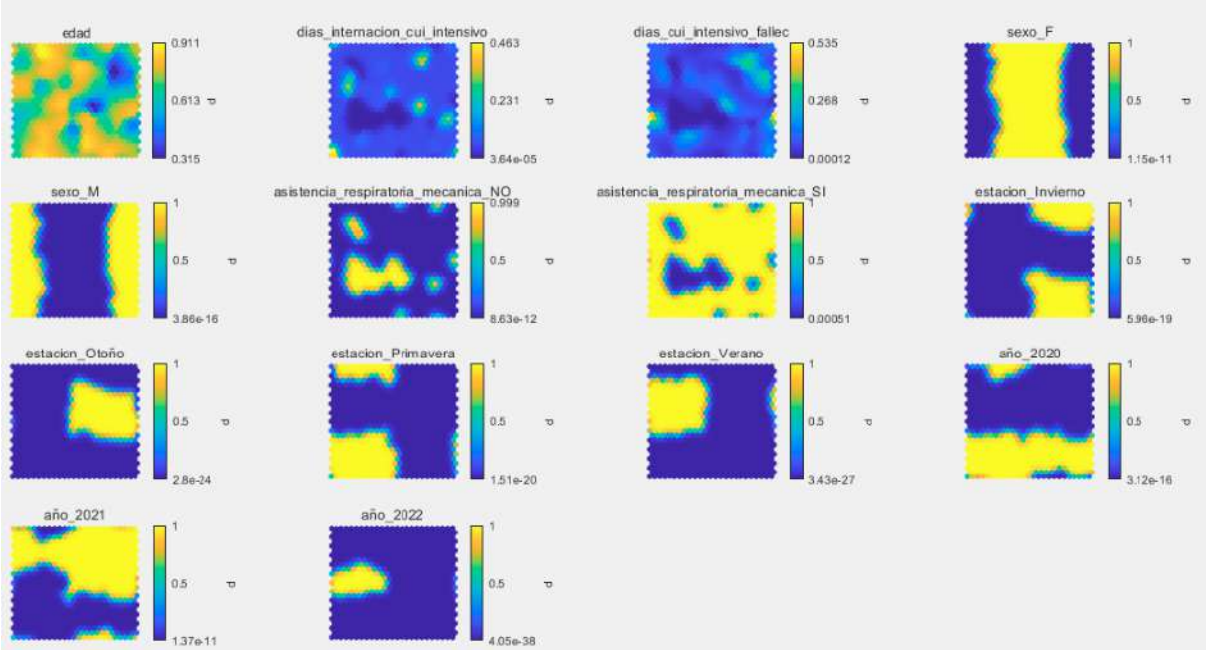
Estas y muchas conclusiones más se pueden ver analizando el **Anexo 1** con los resultados detallados de cada *cluster* y haciendo uso del software desarrollado.

4.2.13.3. *Dataset* 2

A continuación se muestra, en la **Figura 65**, el plano de componentes correspondiente al *dataset* del HPC:

Figura 65

Planos de componentes de todos los atributos que componen el dataset 2



Si se quisieran mencionar algunas conclusiones observando el gráfico anterior y los resultados en este *dataset* se podría decir que:

- Al igual que en el *dataset* anterior, se observan regiones fuertemente delimitadas en los atributos categóricos.
- En los planos de componentes puede observarse cómo los pacientes que utilizaron asistencia respiratoria mecánica predominan en la mayor parte del mapa.

- Mirando los atributos **dias_internacion_cui_intensivo**, **dias_cui_intensivo_fallec** y **asistencia_respiratoria_mecanica_NO** del gráfico anterior, se concluye que en las zonas del mapa donde están los menores valores de **dias_internacion_cui_intensivo** y **dias_cui_intensivo_fallec** corresponden a pacientes que no hicieron uso de asistencia respiratoria mecánica. Esto también se puede notar analizando los gráficos descriptivos de los *clusters* **4**, **10**, **13** y **18**.

- El ***cluster* 5** es el que mayor cantidad de puntos de datos tiene. El 100% son pacientes femeninos, el 100% corresponden a la estación otoño, el 98.9% al año 2021 y finalmente 95.5% utilizaron respirador. El ***cluster* 3** (que tiene aproximadamente la misma cantidad de puntos de datos que el ***cluster* 5**) también corresponde a pacientes femeninos que en su mayoría usaron respirador, pero difiere en la estación (primavera) y en el año (2020). En cuanto a atributos numéricos, son muy similares ambos *clusters* en los atributos **dias_internacion_cui_intensivo** (con al menos 75% de los pacientes con 1 día) y también similares en el atributo **dias_cui_intensivo_fallecimiento** (con una mediana de 5.5 y 5 respectivamente). En cuanto a la edad, difieren ligeramente, ya que en el ***cluster 3*** la mediana corresponde al 88 y el ***cluster 5*** al 84.

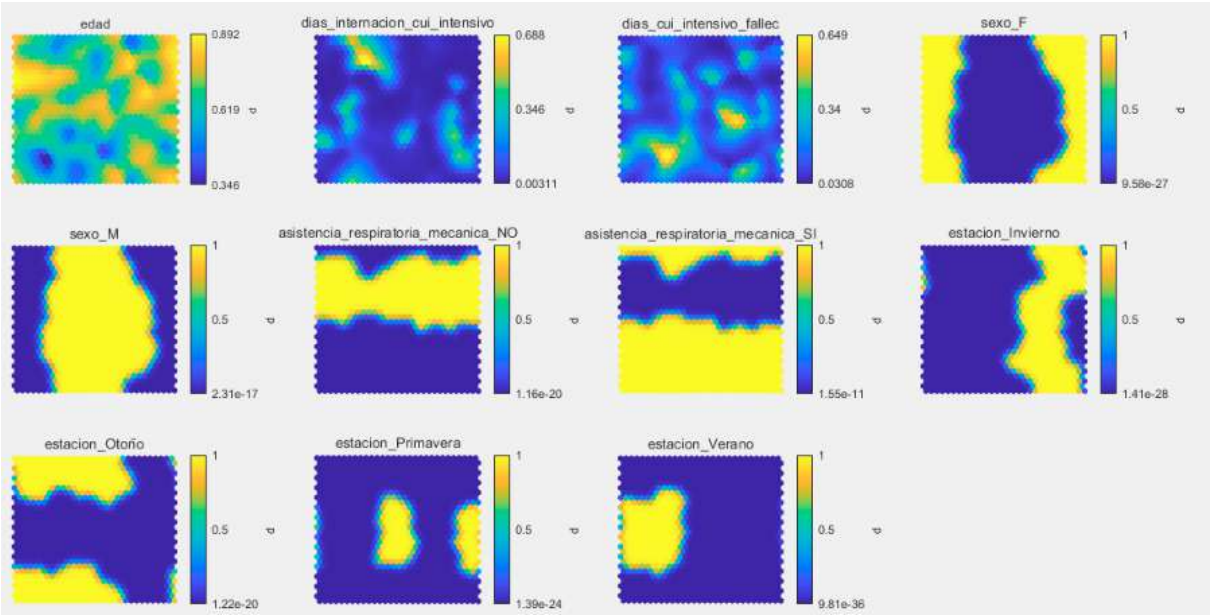
- Se puede observar que en los *clusters* correspondientes al sexo masculino, el uso de asistencia respiratoria es dominante excepto en el ***cluster 4*** (cuya cantidad de casos es de sólo 11) y el ***cluster 10*** (del cual sólo 3 son pacientes masculinos). Los *clusters* **2**, **7**, **8**, **11**, **12**, **15** y **16** tienen entre 78.3% y 100% de uso de respirador.

4.2.13.4. *Dataset* 3

A continuación se muestra, en la **Figura 66**, el plano de componentes correspondiente al *dataset* 3:

Figura 66

Planos de componentes de todos los atributos que componen el dataset 3



Algunas conclusiones:

- Todos los *clusters* que presentan una porción significativa del año 2022, corresponden a la estación de verano. Estos *clusters* son: **1**, **3**, **8** y **15**.
- Todos los *clusters* que presentan como año dominante el 2020 corresponde a estaciones únicamente de primavera e invierno (*clusters* **6**, **7**, **9**, **11**, **12**, **13**, **14** y **16**) y los *clusters* con año dominante 2021 a estaciones otoño y verano (*clusters* **1**, **2**, **3**, **4**, **5**, **8**, **10** y **15**).

- Todos los *clusters* que poseen al menos un 75% de casos con cero días de internación (***clusters* 3**, **5**, **7**, **8**, **11**, **12** y **16**) corresponden a casos que no utilizaron respiración mecánica asistida.

- El *cluster* con mayor número de casos (***cluster* 10**) destaca además por ser el que menor valor tiene en cuanto a media y mediana en el atributo edad y el que mayor valor tiene en el atributo **dias_internacion_cui_intensivo**.

- La menor mediana en cuanto al atributo **dias_cui_intensivo_fallec** se corresponde al ***cluster* 4**.

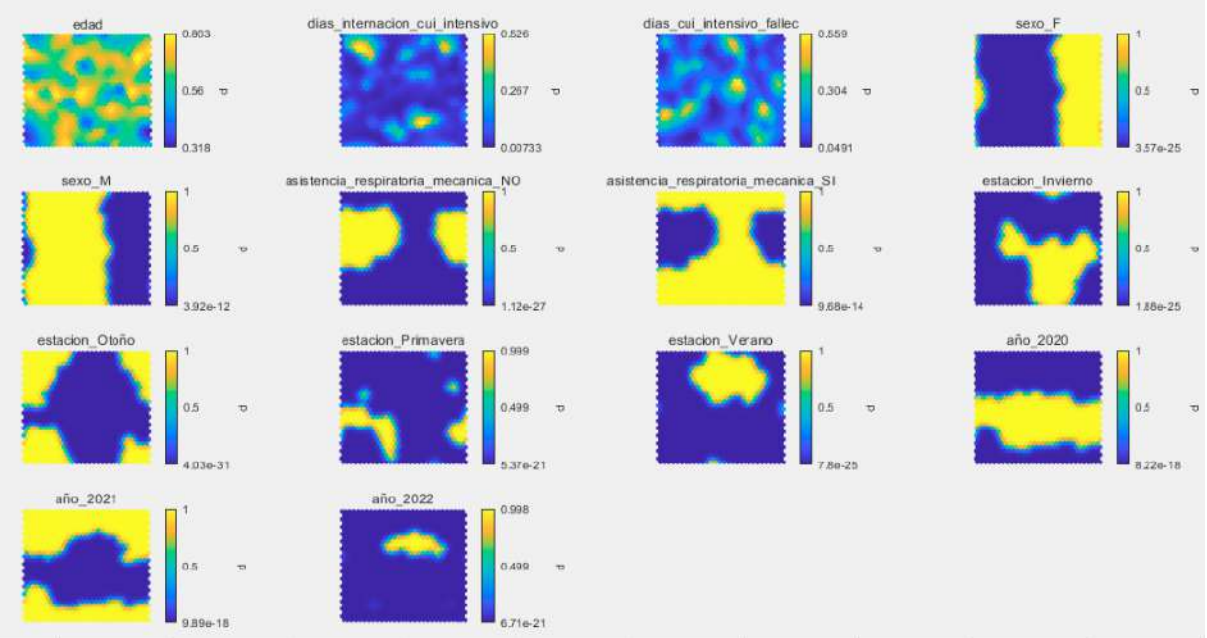
- Los *clusters* se distribuyen uniformemente entre los atributos categóricos, es decir, de los 16 *clusters* 8 corresponden a pacientes únicamente femeninos y los restantes 8 a masculinos. Lo mismo ocurre para el atributo **asistencia_respitaroria_mecanica** y para las estaciones.

4.2.13.5. *Dataset* 4

A continuación se muestra, en la **Figura 67**, el plano de componentes correspondiente al *dataset* 4:

Figura 67

Planos de componentes de todos los atributos que componen el dataset 4



Algunas conclusiones:

- Se observa que la mayoría de los *clusters* (7 correspondientes a pacientes masculinos y 7 a femeninos) corresponden al uso de asistencia respiratoria mecánica. Estos *cluster* son: **1**, **2**, **3**, **7**, **8**, **9**, **12**, **13**, **16**, **17**, **19**, **20**, **23**, **24**.

- El *cluster* con mayor número de casos (***cluster* 19** con 872 casos) destaca además por ser el que menor valor tiene en cuanto a media y mediana en el atributo edad y el de mayor en **dias_internacion_cui_intensivo**. Corresponde a pacientes masculinos que utilizaron respiración asistida en otoño de 2021.

- En cuanto a los *clusters* que poseen al menos el 75% de los pacientes con 0 días de internación, ninguno utilizó asistencia respiratoria mecánica.

    - Uno corresponde a pacientes femeninos en invierno de 2020.

    - Otro corresponde a pacientes masculinos en invierno de 2020.

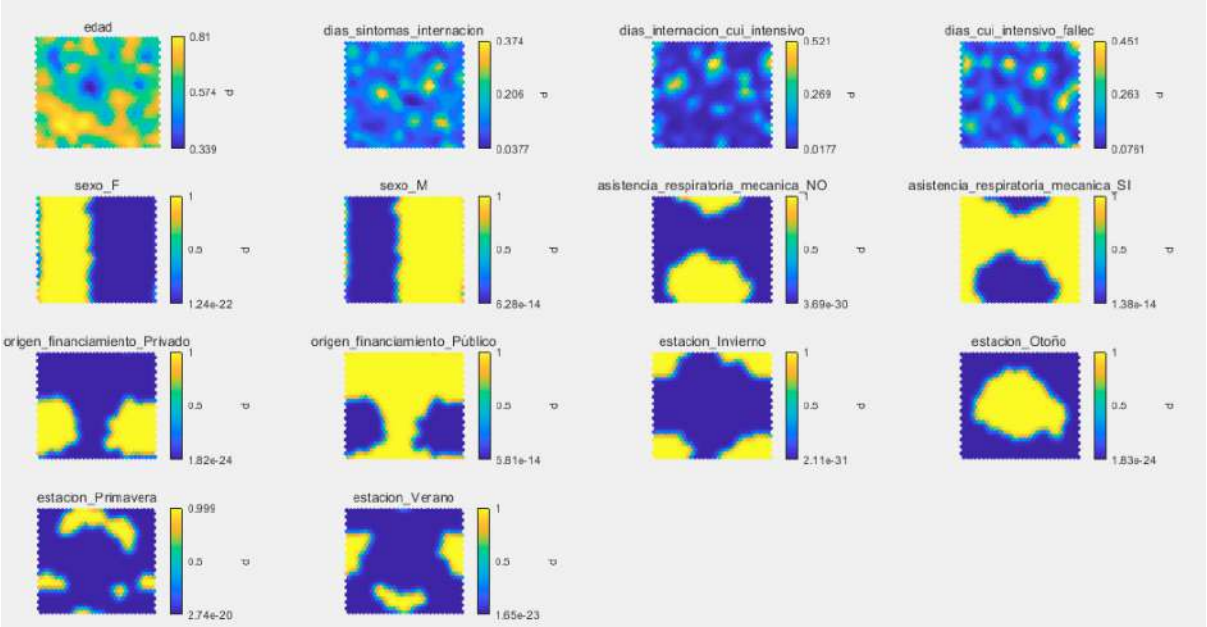
- El último, corresponde a pacientes masculinos mayormente repartidos entre verano e invierno (en su mayoría) de 2021.

4.2.13.6. *Dataset* 5

A continuación se muestra el plano de componentes correspondiente al *dataset* 5 (ver **Figura 68**):

Figura 68

Planos de componentes de todos los atributos que componen el dataset 5



Observando la caracterización de cada *cluster* del *dataset* 5 presentada en el **Anexo 1** y la **Figura 72**, se pueden resaltar algunas conclusiones.:

Se observan dos grandes *clusters* en el centro del mapa: en color tostado el nro. 25 contiguo al nro. 13 en verde y éste contiguo al nro. 24. Además llama la atención el *cluster* 6 que contiene las neuronas con mayor número de activaciones

- El *cluster* 25 corresponde a pacientes de edad promedio 58,91 años, de sexo masculino, el 26% de los cuales residía en provincia de Buenos Aires. Tuvieron 5,81

días promedio antes de la internación, 2,67 días para pasar a cuidados intensivos y 11,48 días promedio hasta el fallecimiento. Todos requirieron de asistencia respiratoria mecánica y todos los casos ocurrieron en otoño.

- El *cluster* contiguo nro. 13 presenta pacientes de edad promedio 59,28 años de sexo femenino, el 25% de los cuales residía en provincia de Buenos Aires. Tuvieron 5,53 días promedio antes de la internación, 2,46 días promedio para pasar a cuidados intensivos y 11,08 días hasta el fallecimiento. Todos requirieron asistencia respiratoria mecánica y todos los casos ocurrieron en otoño. Se puede ver que este *cluster* es vecino al *cluster* 25 y ambos poseen muchas características similares.

- Por otro lado, el *cluster* 24 vecino al 13, presenta casos de pacientes de 66,04 años de edad promedio de sexo femenino, de los cuales un 47,7 % residía en provincia de Buenos Aires. Tuvieron 5,05 días promedio hasta la internación, 2,39 días para pasar a cuidados intensivos y 11,34 días promedio hasta el fallecimiento. Todos requirieron asistencia respiratoria mecánica y los casos se dieron en otoño.

- El *cluster* 6 un poco más alejado en el mapa, muestra pacientes de edad promedio 73,68 años de sexo masculino, el 52,2 % de los cuales residía en provincia de Buenos Aires. Tuvieron 3,92 días promedio antes de la internación, pasando a cuidados intensivos en un lapso de 1,13 días y 10,30 días hasta el fallecimiento. No requirieron asistencia respiratoria mecánica. Estos casos ocurrieron mayormente en invierno (32%) y en otoño (30%).

Se puede añadir que existió un ordenamiento natural con el atributo "**provincia**" en este *dataset*. Por ejemplo, los casos de Chaco, Chubut, Córdoba, Santa Cruz y Tierra del Fuego tendieron a agruparse principalmente en la parte superior del mapa (ver **Figuras 69** a **73**).

Figura 69

SOM del dataset 5 con solo los hits correspondientes la provincia de Chaco

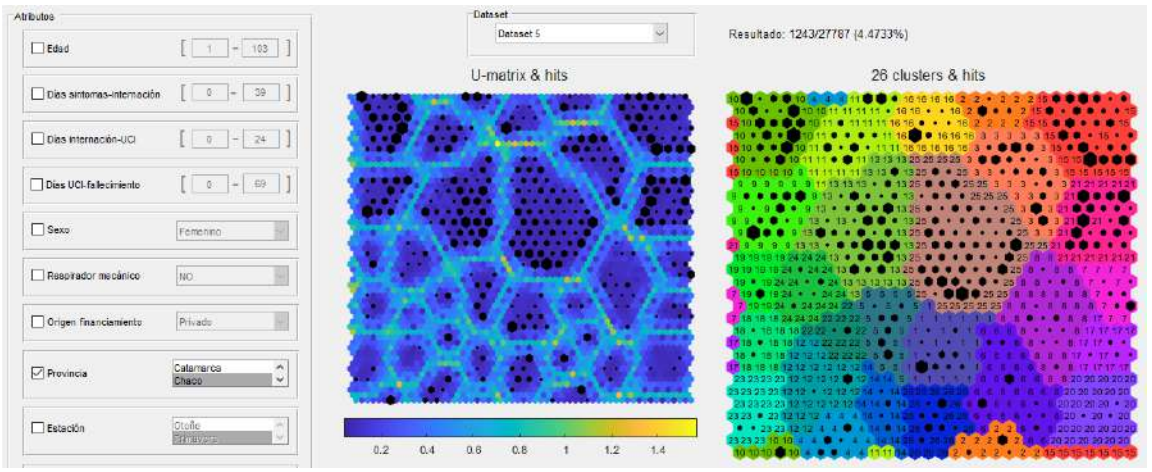


Figura 70

SOM del dataset 5 con solo los hits correspondientes la provincia de Chubut

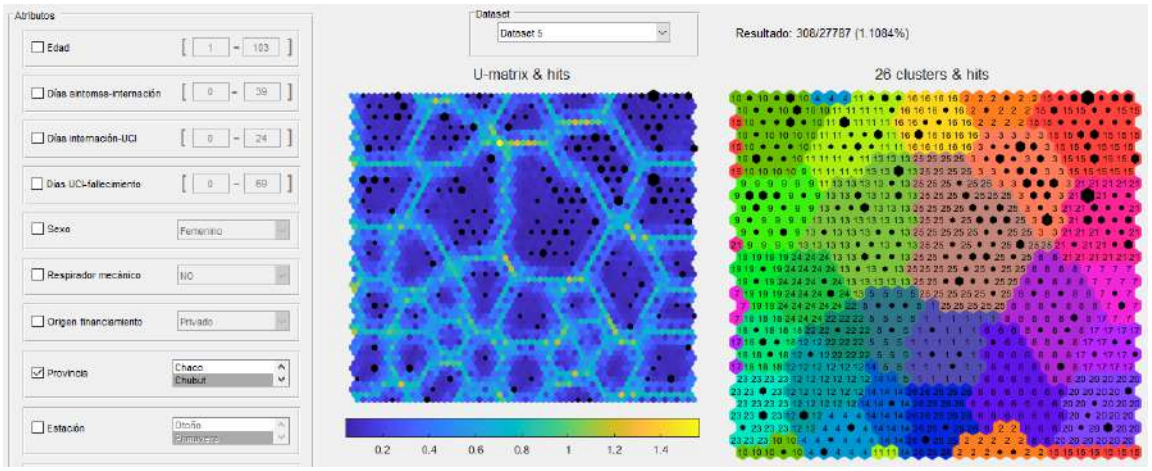


Figura 71

SOM del dataset 5 con solo los hits correspondientes la provincia de Córdoba

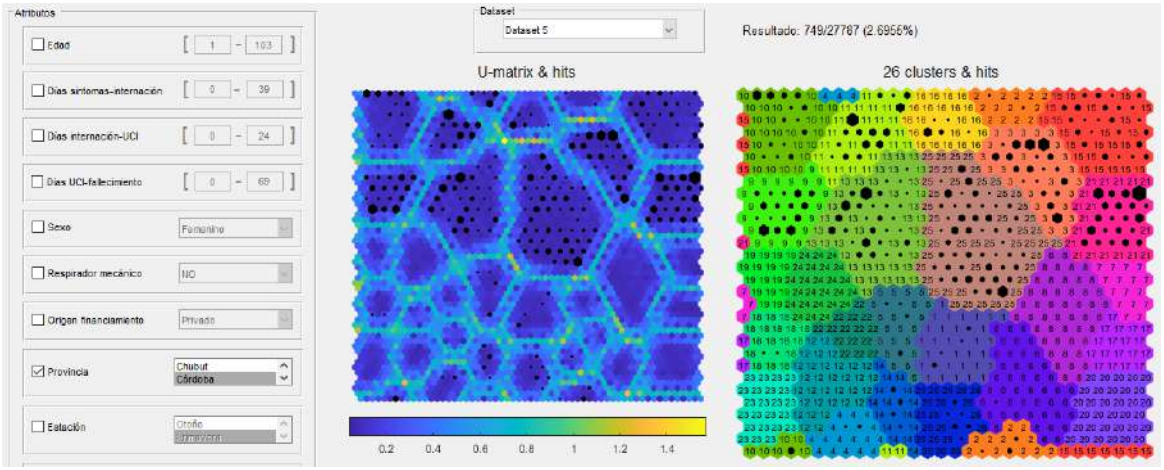


Figura 72

SOM del dataset 5 con solo los hits correspondientes la provincia de Santa Cruz

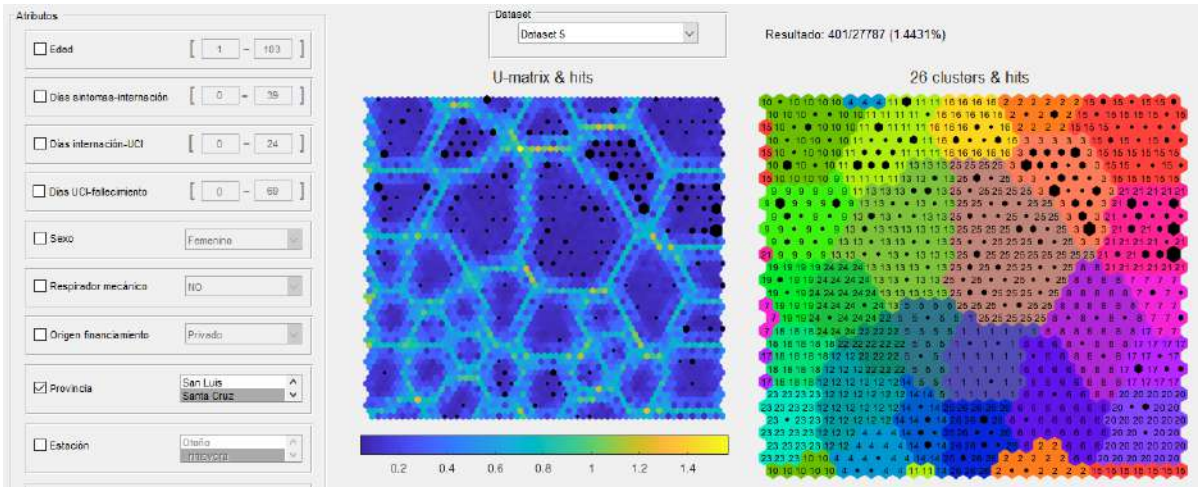
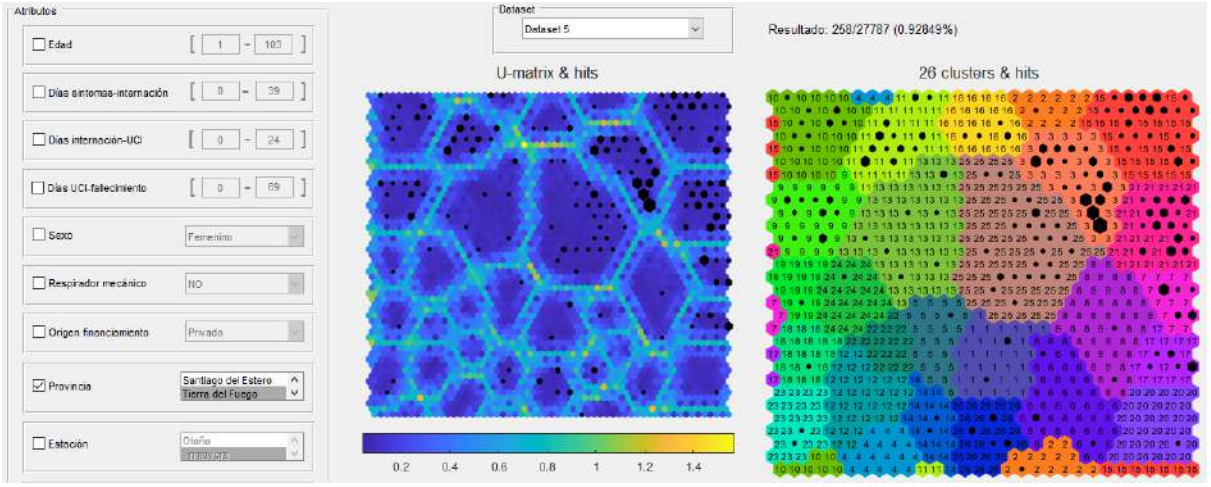


Figura 73

SOM del dataset 5 con solo los hits correspondientes la provincia de Tierra del Fuego



Por otra parte, las provincias Buenos Aires, Corrientes, Mendoza, Salta, Santa Fe y también CABA se distribuyeron a lo largo de todo el mapa por lo que tienen presencia en prácticamente todos los *clusters*.

Este es un ejemplo de conclusiones que podrían establecerse en función de los resultados obtenidos, pudiéndose ampliar el análisis.

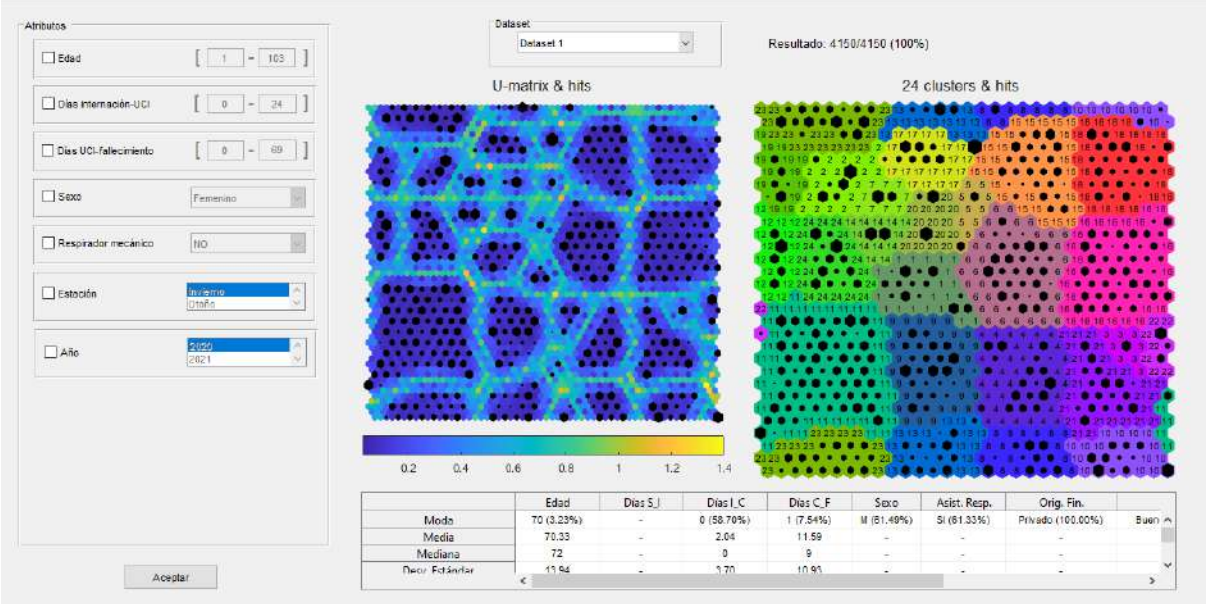
4.3. Software prototipo para análisis de resultados de data mining

Se desarrolló un prototipo en MATLAB© para facilitar el análisis de los resultados obtenidos en la fase de *data mining* de los 5 *datasets* utilizados para el trabajo final.

Presenta una interfaz simple con todos los atributos que componen el *dataset* seleccionado junto con su matriz-U, los *clusters* y un cuadro que indica el valor que toman los parámetros estadísticos del resultado. Todos estos componentes se pueden visualizar en la **Figura 74**.

Figura 74

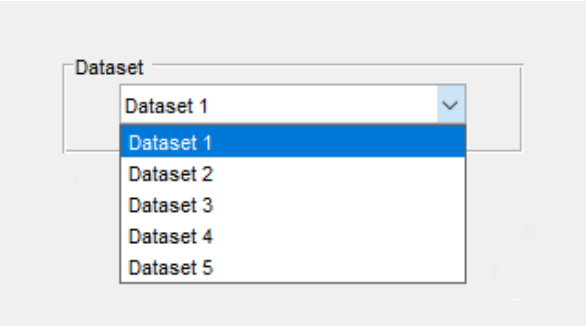
Interfaz del prototipo MATLAB© para análisis de resultados



En programa se ejecuta por defecto con el *dataset* 1 pero es posible elegir cualquiera de los 5 a través de la lista en la parte superior de la ventana (ver **Figura 75**).

Figura 75

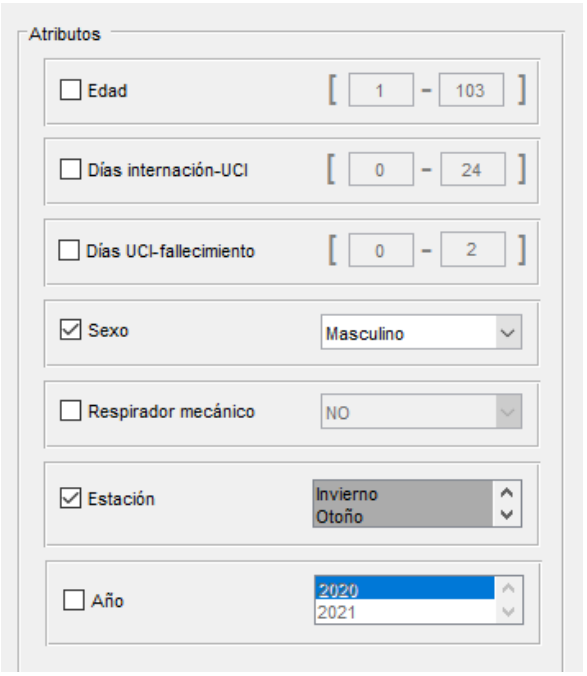
Componente lista del prototipo con las 5 opciones disponibles



A la izquierda de la ventana se encuentran los atributos del conjunto de datos correspondiente. Cada uno presenta un *checkbox* a la izquierda que permite que se seleccione o no el atributo en cuestión y a su derecha un componente para indicar el o los valores o rango que se quiere que tomen (ver **Figura 76**). El propósito de esto es poder elegir por cuáles atributos y con qué valores se desea filtrar el *dataset*.

Figura 76

Lista de atributos para el dataset 1 con los checkboxes para su selección y con los componentes para la carga de valores.

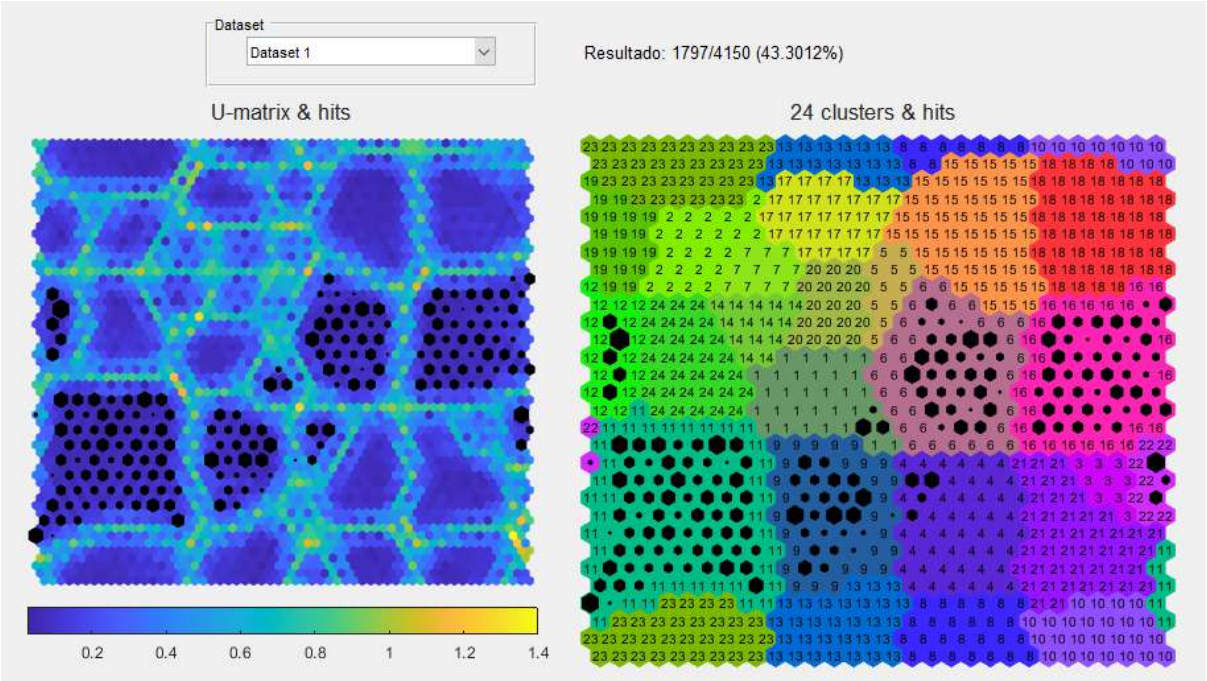


Nota. En este ejemplo se seleccionaron los atributos "sexo" con el valor "masculino" y "estación" con los valores "invierno" y "otoño". Para seleccionar más de una opción en los atributos categóricos se debe mantener presionada la tecla "ctrl" y al mismo tiempo presionar las opciones deseadas.

Al presionar en el botón “aceptar” se filtran los datos y se muestran los *hits* resultantes sobre la matriz-U y sobre el mapa con los *clusters* del *dataset* elegido como se puede ver en la **Figura 77**.

Figura 77

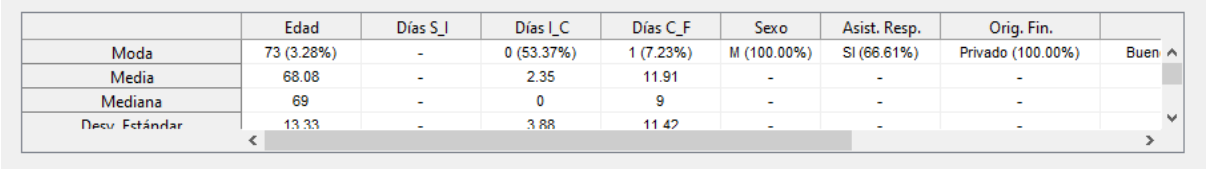
Resultado de haber seleccionado los atributos “sexo” con el valor “masculino” y “estación” con los valores “invierno” y “otoño” del dataset 1.



En la parte inferior del programa se encuentra el cuadro con los parámetros estadísticos. Este indica los valores de moda, media, mediana, desviación estándar, desviación mediana absoluta, mínimo y máximo por atributo del resultado (ver **Figura 78**).

Figura 78

Cuadro estadístico por atributo del resultado de filtrar los datos por sexo masculino y por estaciones otoño e invierno del dataset 1.



	Edad	Días S_I	Días I_C	Días C_F	Sexo	Asist. Resp.	Orig. Fin.	
Moda	73 (3.28%)	-	0 (53.37%)	1 (7.23%)	M (100.00%)	SI (66.61%)	Privado (100.00%)	Buen
Media	68.08	-	2.35	11.91	-	-	-	
Mediana	69	-	0	9	-	-	-	
Desv. Estándar	13.33	-	3.88	11.42	-	-	-	

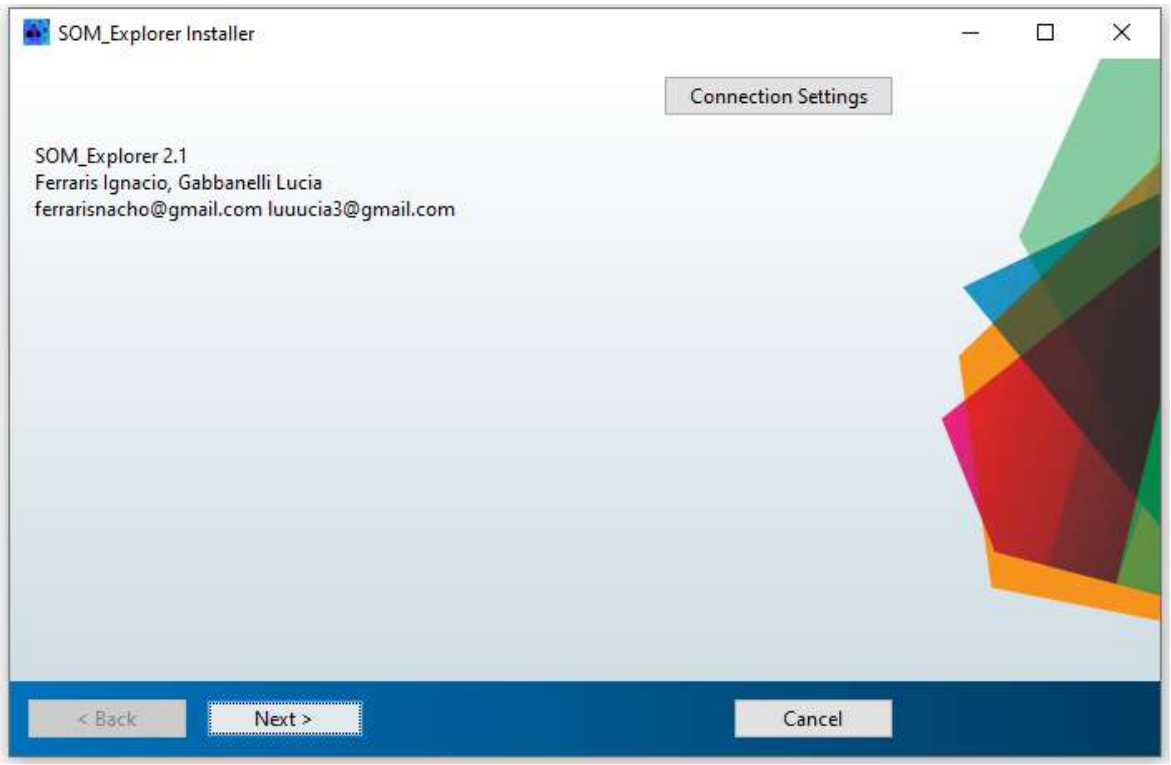
La idea es poder filtrar cualquiera de los 5 *datasets* por conjuntos de atributos con valores de interés y ver en qué parte del mapa se encuentran los puntos de datos con esas características. Estos *hits* se visualizan tanto en la matriz-U como en el mapa de grupos porque en la primera se puede obtener información de las distancias entre las neuronas que rodean los resultados y en el segundo se puede conocer la etiqueta de los *clusters* implicados.

El requerimiento principal para ejecutar este software es tener instalado el entorno de ejecución de MATLAB©. Este se puede descargar de forma gratuita desde la página https://www.mathworks.com/products/compiler/matlab-runtime.html. Se recomienda además contar con por lo menos 700 *megabytes* de memoria disponible para su ejecución.

En el siguiente link se puede acceder al instalador, ilustrado en la **Figura 79**: https://drive.google.com/drive/folders/1zwg6yFd2e0P6WE-3PM677_9piiqzfIzz

Figura 79

Interfaz de instalación del software.



5. Equipos utilizados para el procesamiento

A continuación se presentarán los detalles de los distintos equipos que se utilizaron a lo largo del proyecto:

1. **Equipo N°1**: contaba con un procesador AMD FX-8350 y 12 *Gigabytes* de memoria RAM. Se utilizó principalmente durante las primeras etapas de preprocesamiento y hasta la fase de data mining con k-prototypes inclusive. Fue descartado luego de presentar fallas técnicas.

2. **Equipo N°2**: reemplazó al **Equipo N°1** luego de la fase de k-prototypes y se siguió utilizando hasta la culminación del proyecto. Cuenta con un procesador AMD Ryzen 5 5500 y 16 Gb de memoria RAM.

3. **Equipo N°3**: se utilizó en conjunto con el **Equipo N°2** desde la fase de *data mining* con SOM hasta la finalización del proyecto. Presenta como característica un procesador Intel(R) Core(TM) i7-1165G7 y 16 Gb de memoria RAM.

6. Gestión del proyecto

6.1. Análisis FODA

Previo al inicio formal del proyecto, se realizó un análisis acerca de las fortalezas, oportunidades, debilidades y amenazas que pudieran intervenir directa o indirectamente sobre la viabilidad y los objetivos del mismo. Esto permitió conocer tanto las ventajas a disposición como los puntos críticos que suponían riesgos para el trabajo.

Fortalezas

- Los estudiantes integrantes del proyecto final ya han trabajado juntos anteriormente en otros proyectos tanto dentro como fuera del ámbito universitario.
- Los docentes participantes del proyecto final han dirigido otros de carácter similar en el pasado y cuentan con una amplia experiencia profesional.
- La directora del proyecto cuenta con experiencia en reconocimiento de patrones y redes neuronales.
- Uno de los docentes del proyecto está relacionado con la institución privada involucrada y puede actuar de nexo entre ambas partes para lograr una mejor comunicación.

Oportunidades

- Tanto la institución privada como la sociedad en general puede beneficiarse de la información resultante que será producto de este proyecto.
- El proceso resultante puede ser reutilizado en el futuro con distintos *datasets* de interés para obtener conclusiones relevantes.

- La institución privada involucrada tiene almacenada una gran cantidad de datos relevantes a procesar dado el tiempo que pasó desde el comienzo de la pandemia hasta el día de hoy. Esto permitiría tener una mayor precisión en los resultados obtenidos a la hora de aplicar los algoritmos.
- Las técnicas de *data mining* que se utilizarán ya han sido puestas a prueba en estudios que abarcan diversos ámbitos (incluyendo la gestión de la salud) y se han obtenido resultados satisfactorios, por lo que el proyecto tiene potencial.

Debilidades

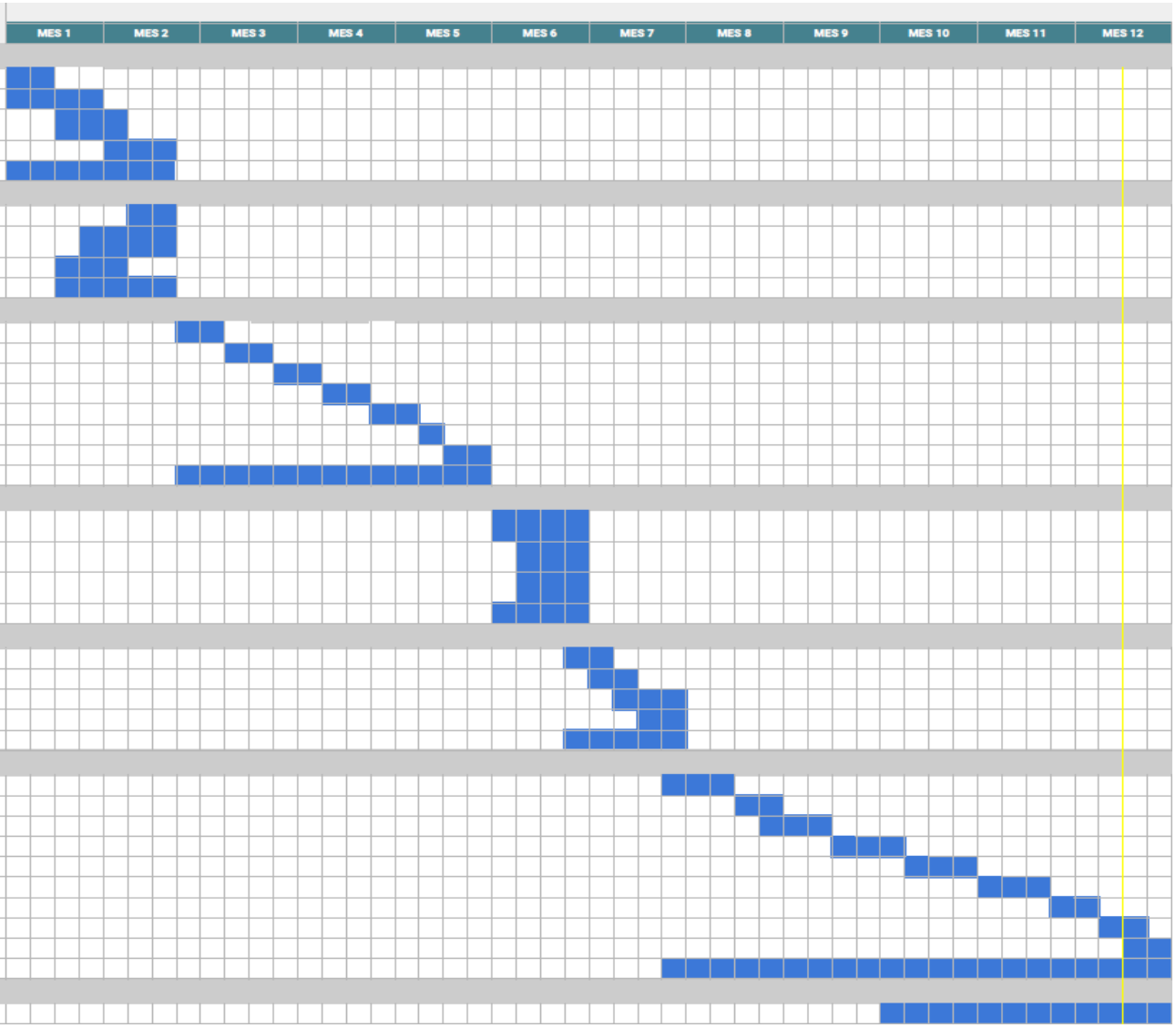
- Los alumnos integrantes del proyecto no han trabajado anteriormente con todas las herramientas y lenguajes de programación a utilizar durante el desarrollo del proyecto.
- Los alumnos integrantes del proyecto no tienen experiencia con las técnicas de *data mining* que se utilizarán, y el contenido relacionado al área de la Inteligencia Artificial que se ha visto en las asignaturas de la carrera es muy poco.
- Puede resultar limitado el hardware que disponen los estudiantes dada la cantidad y la complejidad de datos disponibles a procesar.

Amenazas

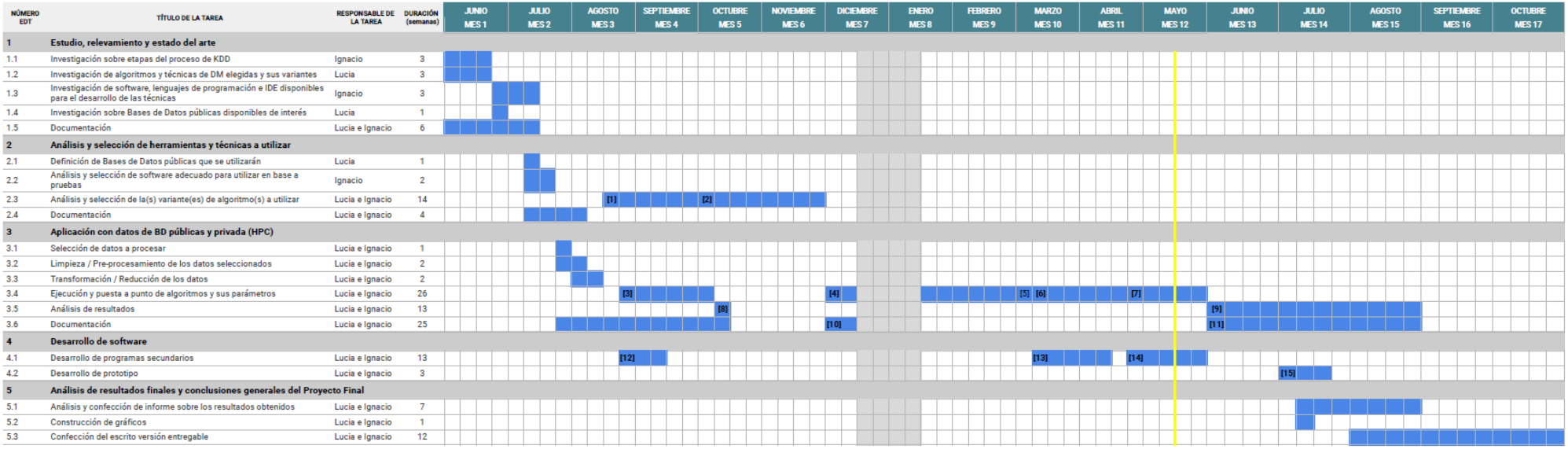
- Posibles conflictos a la hora de tener acceso a los datos debido a que son de procedencia privada (seguridad y confidencialidad).
- Dado el contexto de pandemia actual, siempre pueden surgir inconvenientes de cualquier tipo durante el desarrollo del proyecto, y más aún si se está trabajando con una institución relacionada con la salud. Esto podría afectar en el período de tiempo de finalización del mismo.

6.2. Diagrama de Gantt original

NÚMERO EDT	TÍTULO DE LA TAREA	RESPONSABLE DE LA TAREA	DURACIÓN (semanas)
1	**Estudio, relevamiento y estado del arte**		
1.1	Investigación sobre etapas del proceso de KDD	Ignacio	2
1.2	Investigación de algoritmos y técnicas de DM elegidas y sus variantes	Lucia	4
1.3	Investigación de software, lenguajes de programación e IDE disponibles para el desarrollo de las técnicas	Ignacio	3
1.4	Investigación sobre Bases de Datos públicas disponibles de interés	Lucia	3
1.5	Documentación	Lucia e Ignacio	7
2	**Análisis y selección de herramientas y técnicas a utilizar**		
2.1	Definición de Bases de Datos públicas que se utilizarán	Lucia	2
2.2	Análisis y selección de software adecuado para utilizar en base a pruebas	Ignacio	4
2.3	Análisis y selección de la(s) variante(es) de algoritmo(s) a utilizar	Lucia	3
2.4	Documentación	Lucia e Ignacio	5
3	**Aplicación con datos de BD públicas**		
3.1	Selección de datos a procesar	Lucia e Ignacio	2
3.2	Limpieza / Pre-procesamiento de los datos seleccionados	Lucia e Ignacio	2
3.3	Transformación / Reducción de los datos	Lucia e Ignacio	2
3.4	Puesta a punto de algoritmos y sus parámetros	Lucia e Ignacio	2
3.5	Ejecución de los algoritmos (Data Mining)	Lucia e Ignacio	2
3.6	Construcción de gráficos	Lucia e Ignacio	1
3.7	Análisis de resultados	Lucia e Ignacio	2
3.8	Documentación	Lucia e Ignacio	13
4	**Análisis del sistema del HPC**		
4.1	Análisis de la BD del hospital: su estructura, tipos de datos que maneja, compatibilidad, entre otros.	Lucia e Ignacio	4
4.2	Identificación de posibles necesidades de información a obtener por parte de la institución	Ignacio	3
4.3	Definir requerimientos (funcionales y no funcionales) con respecto al software.	Lucia	3
4.4	Documentación	Lucia e Ignacio	4
5	**Desarrollo de programas secundarios**		
5.1	Análisis de programas secundarios	Lucia	2
5.2	Diseño de programas secundarios	Ignacio	2
5.3	Desarrollo de programas secundarios	Lucia	3
5.4	Testing de programas secundarios	Ignacio	2
5.5	Documentación	Lucia e Ignacio	5
6	**Aplicación con BD del HPC**		
6.1	Selección de datos a procesar	Lucia e Ignacio	3
6.2	Extracción y obtención de los datos	Lucia	2
6.3	Limpieza / Pre-procesamiento de los datos obtenidos	Ignacio	3
6.4	Transformación / Reducción de datos	Lucia	3
6.5	Puesta a punto de algoritmos y sus parámetros	Lucia e Ignacio	3
6.6	Ejecución de los algoritmos (Data Mining)	Lucia e Ignacio	3
6.7	Construcción de gráficos	Ignacio	2
6.8	Análisis de resultados	Lucia e Ignacio	2
6.9	Confección de informe sobre los resultados obtenidos	Lucia e Ignacio	2
6.10	Documentación	Lucia e Ignacio	21
7	**Análisis de resultados finales y conclusiones generales del Proyecto Final**		
7.1	Finalización del escrito	Lucia e Ignacio	12



6.3. Diagrama de Gantt rectificado



NÚMERO EDT	TÍTULO DE LA TAREA	RESPONSABLE DE LA TAREA	DURACIÓN (semanas)
1	**Estudio, relevamiento y estado del arte**		
1.1	Investigación sobre etapas del proceso de KDD	Ignacio	3
1.2	Investigación de algoritmos y técnicas de DM elegidas y sus variantes	Lucia	3
1.3	Investigación de software, lenguajes de programación e IDE disponibles para el desarrollo de las técnicas	Ignacio	3
1.4	Investigación sobre Bases de Datos públicas disponibles de interés	Lucia	1
1.5	Documentación	Lucia e Ignacio	6
2	**Análisis y selección de herramientas y técnicas a utilizar**		
2.1	Definición de Bases de Datos públicas que se utilizarán	Lucia	1
2.2	Análisis y selección de software adecuado para utilizar en base a pruebas	Ignacio	2
2.3	Análisis y selección de la(s) variante(es) de algoritmo(s) a utilizar	Lucia e Ignacio	14
2.4	Documentación	Lucia e Ignacio	4
3	**Aplicación con datos de BD públicas y privada (HPC)**		
3.1	Selección de datos a procesar	Lucia e Ignacio	1
3.2	Limpieza / Pre-procesamiento de los datos seleccionados	Lucia e Ignacio	2
3.3	Transformación / Reducción de los datos	Lucia e Ignacio	2
3.4	Ejecución y puesta a punto de algoritmos y sus parámetros	Lucia e Ignacio	26
3.5	Análisis de resultados	Lucia e Ignacio	13
3.6	Documentación	Lucia e Ignacio	25
4	**Desarrollo de software**		
4.1	Desarrollo de programas secundarios	Lucia e Ignacio	13
4.2	Desarrollo de prototipo	Lucia e Ignacio	3
5	**Análisis de resultados finales y conclusiones generales del Proyecto Final**		
5.1	Análisis y confección de informe sobre los resultados obtenidos	Lucia e Ignacio	7
5.2	Construcción de gráficos	Lucia e Ignacio	1
5.3	Confección del escrito versión entregable	Lucia e Ignacio	12

Nota: las celdas en gris significan semanas de inactividad.

Este nuevo diagrama fue elaborado recién durante las últimas etapas del proyecto (indicado con una línea vertical amarilla en ambos diagramas) a modo de *checkpoint* para tener una mejor organización de las tareas que aún faltaban completarse. Se realizaron cambios en la estructura del mismo por las razones detalladas en la sección de Planificación vs. ejecución.

Notas agregadas en el diagrama de Gantt rectificado:

[1] Análisis variantes k-means

[2] Análisis variantes SOM

[3] K-prototypes

[4] SOM (Java)

[5] MATLAB©

[6] SOM (Python)

[7] SOM (MATLAB©)

[8] Análisis resultados k-prototypes

[9] Análisis ejecuciones en MATLAB©

[10] Documentación Java

[11] Documentación ejecuciones en MATLAB©

[12] Programas k-prototypes Python

[13] Programas SOM Python

[14] Programas MATLAB©

[15] Prototipo MATLAB©

6.4. Análisis de tiempos

Para la estimación de tiempos se tuvo en cuenta lo siguiente:

- Se consideraron 2 horas promedio por día en cada tarea.
- Las tareas de documentación se realizan en simultáneo al desarrollo de las demás, por lo que sólo se sumó tiempo extra en el caso de que la actividad de documentar estuviera sin estar superpuesta por otra fase en el Gantt.
- Se consideraron 5 días de actividad por semana.

Las fases implicadas en el proyecto junto a su descripción y el tiempo dedicado se detalla a continuación:

- Estudio, relevamiento y estado del arte: esta etapa incluye la lectura e investigación de las etapas del proceso de KDD y de las técnicas elegidas de *data mining* con sus variantes. También, incluye la búsqueda e investigación de software, lenguaje de programación e IDEs disponibles para el uso de los algoritmos. Por último, la búsqueda de bases de datos públicas para pruebas y desarrollo del proyecto. **Tiempo total**: 120 horas.
- Análisis y selección de herramientas y técnicas a utilizar: se incluye las pruebas y análisis de los distintos software disponibles para ejecutar las distintas etapas del KDD, el completo entendimiento de las variantes (tanto de k-means como de SOM) para analizar posible implementación (para el caso de variantes que no se encuentren en librerías) y para realizar la selección. Por último, la selección de la base de datos a utilizar en el proyecto. **Tiempo total**: 190 horas.
- Aplicación a bases de datos públicas y privada (HPC): esta etapa incluye selección de

datos a procesar, limpieza/ pre-procesamiento de los datos seleccionados, transformación/ reducción de los datos, ejecución y puesta a punto de algoritmos y análisis de los resultados obtenidos. **Tiempo total**: 430 horas.

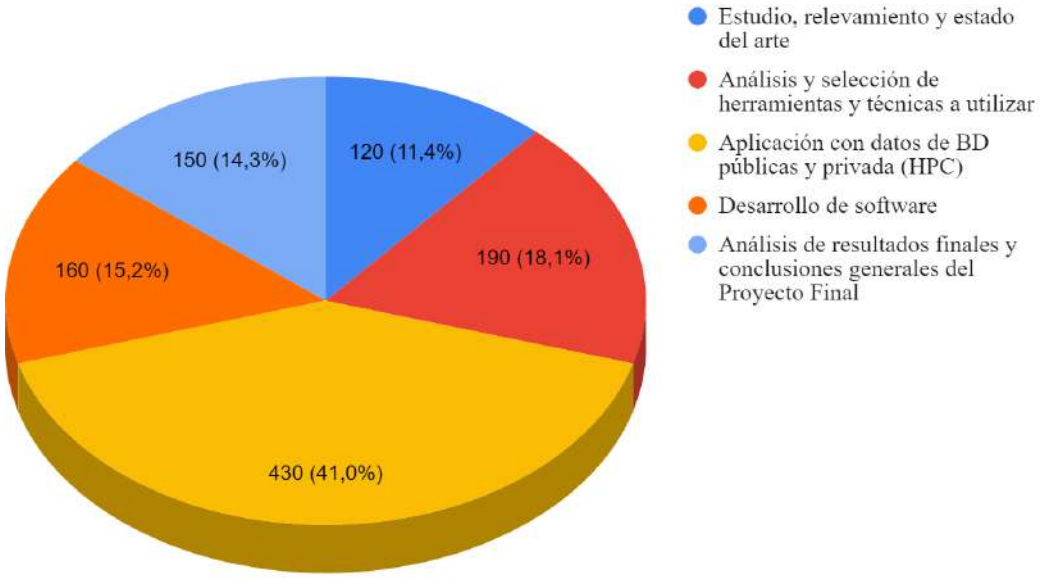
- Desarrollo de software: se incluye la codificación de programas necesarios para llevar a cabo distintas etapas del proyecto y para el desarrollo en MATLAB© de un software prototipo para el análisis de resultados. **Tiempo total**: 160 horas.

- Análisis de resultados finales y conclusiones generales del proyecto: esta etapa incluye el análisis y confección de informe sobre los resultados obtenidos, construcción de gráficos para visualización de resultados y, por último, la confección del escrito versión entregable. **Tiempo total**: 150 horas.

Tiempo total empleado para el proyecto: 1050 horas

Gráfico 1

Gráfico de torta sobre la distribución de los tiempos de cada etapa del proyecto.



Nota: los valores en el gráfico están expresados en horas.

6.5. División de tareas

Inicialmente la asignación de cada tarea se planteó de manera que el tiempo requerido total del proyecto se distribuya lo más equitativamente posible entre los integrantes. La idea era paralelizar tareas, en la medida de lo posible, y optimizar tiempos. La distribución de actividades fue según interés, ya que no se tenía un conocimiento previo sobre los temas que se abordarían. En caso contrario, se habría realizado una planificación teniendo en cuenta el manejo de dominio de cada integrante sobre los temas, y así lograr minimizar tanto la curva de aprendizaje como los tiempos de desarrollo.

Comparando la planificación esperada con la realidad se observaron varias diferencias. Pese a que se intentó sostener la idea de que cada integrante sólo realice las tareas asignadas, no fue posible en todos los casos. Esto sucedió principalmente en las primeras fases del proyecto (investigación y análisis), en donde muchas de las temáticas tratadas eran nuevas y complejas y requerían de la participación de ambos integrantes para el desarrollo y/o toma de decisiones.

Fue recién en la última parte del proyecto donde se pudo respetar la división de las actividades a realizar. Esto fue porque ya se contaba con todo el conocimiento necesario para llevar a cabo los trabajos de manera individual. Esto ocurrió, por ejemplo, en las etapas de ejecución de las técnicas de *data mining*, la confección de gráficos para la visualización de resultados y la documentación de conclusiones.

6.6. Planificación vs. ejecución

En la propuesta del trabajo final se confeccionó un Gantt estableciendo las etapas que se creían que iban a estar involucradas con su esperada duración en semanas. Si bien se tenía en cuenta que al no tener dominio sobre el tema a trabajar este Gantt podría sufrir cambios, la

estimación inicial fue optimista ya que surgieron muchos factores que afectaron en gran medida la planificación inicial.

En primer lugar, y como ya se mencionó en detalle anteriormente, la división de tareas fue lo primero que empezó a alterarse. Se vio necesaria la participación, en mayor o menor medida, de los dos integrantes en conjunto en buena parte de las etapas.

Por otra parte, al ser un proyecto final que no está orientado a la elaboración de un software o sistema como tal sino a un proceso de aprendizaje y a su aplicación en la realidad, las etapas del Gantt no resultaron secuenciales. Ninguno de los métodos o técnicas iniciales que fueron probados respondieron frente al juego de datos que había para trabajar, lo que requirió volver a etapas anteriores para poder solventar los obstáculos que se iban presentando durante el desarrollo. Hicieron falta varias iteraciones entre las fases de minería de datos para lograr resultados satisfactorios.

Si se analiza el **Gráfico 4** se puede observar que no hay grandes cambios en las primeras tareas, sin embargo, en la tarea 2.3, correspondiente al análisis y selección de la(s) variante(es) de algoritmo(s) a utilizar, las semanas de tiempo ejecutado superan ampliamente a las del planificado. Esto es porque en un principio se tenía la idea de poder aplicar alguna variante de SOM que sea para datos mixtos, pero en la etapa de investigación se encontró con que no había librerías que implementaran lo que se estaba buscando. Por lo tanto, se intentó evaluar la posibilidad de realizar una implementación propia para llevar a cabo el objetivo, y es por eso qué se dedicó mucho tiempo en analizar y seleccionar papers referidos al tema. Si bien se terminó utilizando la versión clásica de SOM, el tiempo utilizado para la tarea 2.3 fue mucho más de lo esperado.

Algo sorpresivo a destacar fue la cantidad de semanas extra, comparada a la planificación inicial, que se necesitó para la ejecución de la tarea de puesta a punto de los algoritmos (tarea 3.4). Esta actividad requirió volver a la fase de investigación, ya que poder

tener una agrupación que tuviera sentido y que sea de calidad fue una tarea complicada de solucionar. También, fueron necesarias reuniones extra para poder enfrentar en conjunto esta problemática.

Otro cambio en la planificación surgió en la etapa 4 referida al proceso de obtención de datos del Hospital Privado de Comunidad. En un principio, se esperaba tener que aplicar todos los pasos necesarios para extraer de la base de datos del hospital la información requerida para la evolución del trabajo. Esto no fue así, ya que el codirector del proyecto, con su amplia trayectoria en el establecimiento privado y su dominio sobre el tema, proporcionó un archivo .csv con los datos listos para ser procesados por las técnicas de *data mining* estudiadas. Los atributos de los puntos de datos obtenidos no fueron al azar, sino que se tuvo la intención de que estos coincidieran con los atributos de los datos que ya se habían trabajado y así comparar resultados. Por lo tanto, tanto la extracción de datos como la elección de los mismos resultó una tarea sencilla y no fue necesario, entonces, contemplar esta etapa en el Gantt modificado.

Pasada la mitad del proyecto, durante la etapa de parametrización para la obtención de resultados aceptables, se hizo un análisis utilizando el diagrama de Gantt original para ver cómo afectaba en la estimación global que dicha etapa requiriera más tiempo del esperado. Se decidió no hacer en ese momento una reestructuración de este tiempo debido a que la obtención de datos del HPC requirió mucho menos esfuerzo del estimado, haciendo que exista una compensación entre las etapas. Recién cuando se estaba finalizando con la puesta a punto final de los parámetros se procedió a elaborar una versión modificada del diagrama, con el objetivo de organizar mejor las últimas tareas (análisis de resultados, confección de gráficos, desarrollo del prototipo, redacción del informe).

En la etapa 5 (Desarrollo de Programas Secundarios) los programas que se realizaron, ya sea los necesarios para el desarrollo del proyecto como el realizado para el análisis de

resultados, no eran lo suficientemente extensos como para considerar las tareas de diseño y testing. Eran códigos acotados a una necesidad específica que tan solo con hacer unas pequeñas pruebas era suficiente para su correcta ejecución. Es por eso que no se justificó contemplar las tareas 5.1, 5.2, y 5.4 finalmente para la planificación. La tarea 5.3, correspondiente al propio desarrollo de programas secundarios, resultó más extensa de lo que se esperaba porque se vio la necesidad de realizar programas que automaticen procesos para facilitar ciertas tareas y optimizar tiempos, sumado a que se tuvieron que desarrollar *scripts* para varios lenguajes.

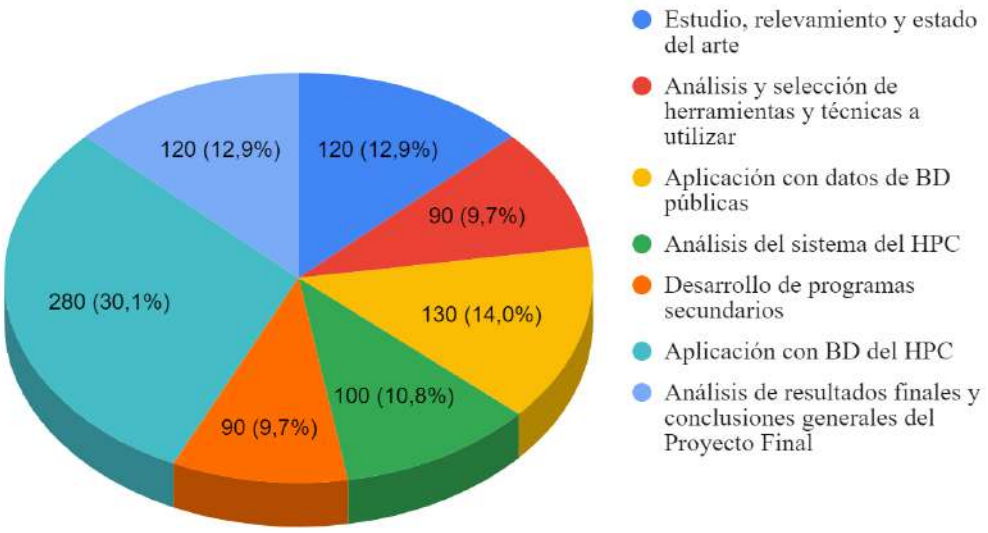
También se pueden ver diferencias entre la planificación y la ejecución en la etapa 6. Similar a lo que sucedió en la etapa 4, al contar con un archivo .csv ya listo para procesar, las tareas desde la 6.1 a 6.4 dejaron de ser necesarias. Luego, se decidió que las etapas de puesta a punto y ejecución de algoritmos para los datos del HPC se integren a la etapa 3. Esto es porque ya se había desarrollado un proceso automático para la ejecución y elección de parámetros en esa etapa y sólo fue necesario ejecutar dichos programas con el archivo del hospital.

Las tareas 5.1 y 5.2 se agregaron en el Gantt modificado porque se hizo un análisis y comparación (acompañado de gráficos, imágenes y tablas) de los resultados finales que se obtuvieron de las bases de datos públicas y del hospital. En la planificación se esperaba la ejecución de estas tareas en las etapas anteriores, pero los resultados más relevantes surgieron una vez que estas fueron finalizadas.

Es importante mencionar que hubo períodos de inactividad del desarrollo. En las primeras etapas la dedicación fue mayor y luego, en la medida del transcurso del tiempo, surgieron inconvenientes que hicieron que el ritmo de trabajo no fuese el mismo en todo momento. Situaciones inesperadas como viajes, problemas de salud, fallos en el hardware y trabajo son algunos ejemplos.

Gráfico 2

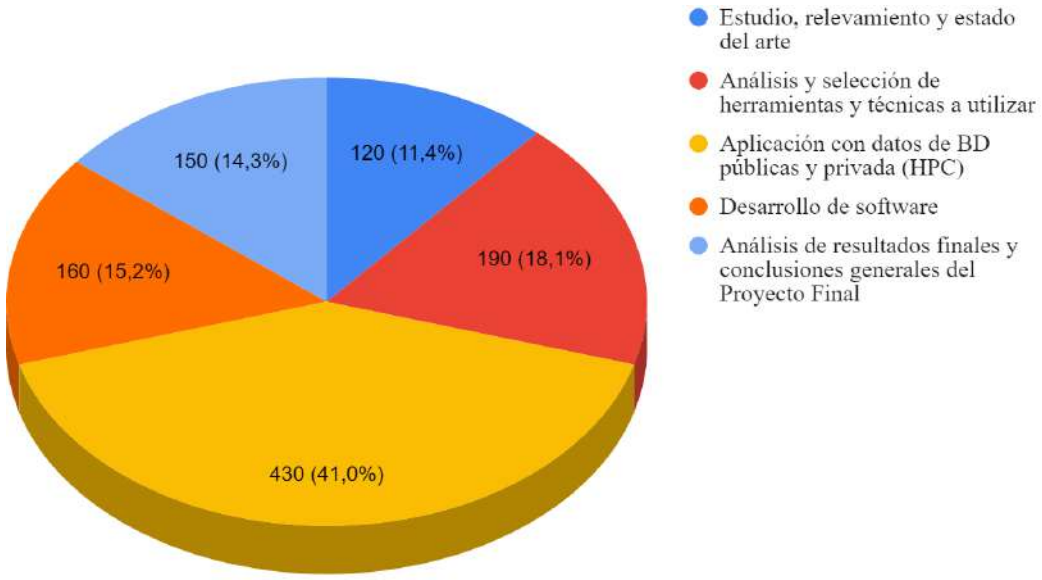
Gráfico de torta de la planificación inicial.



Nota: los valores en el gráfico están expresados en horas (tiempo total empleado para el proyecto: 930 horas).

Gráfico 3

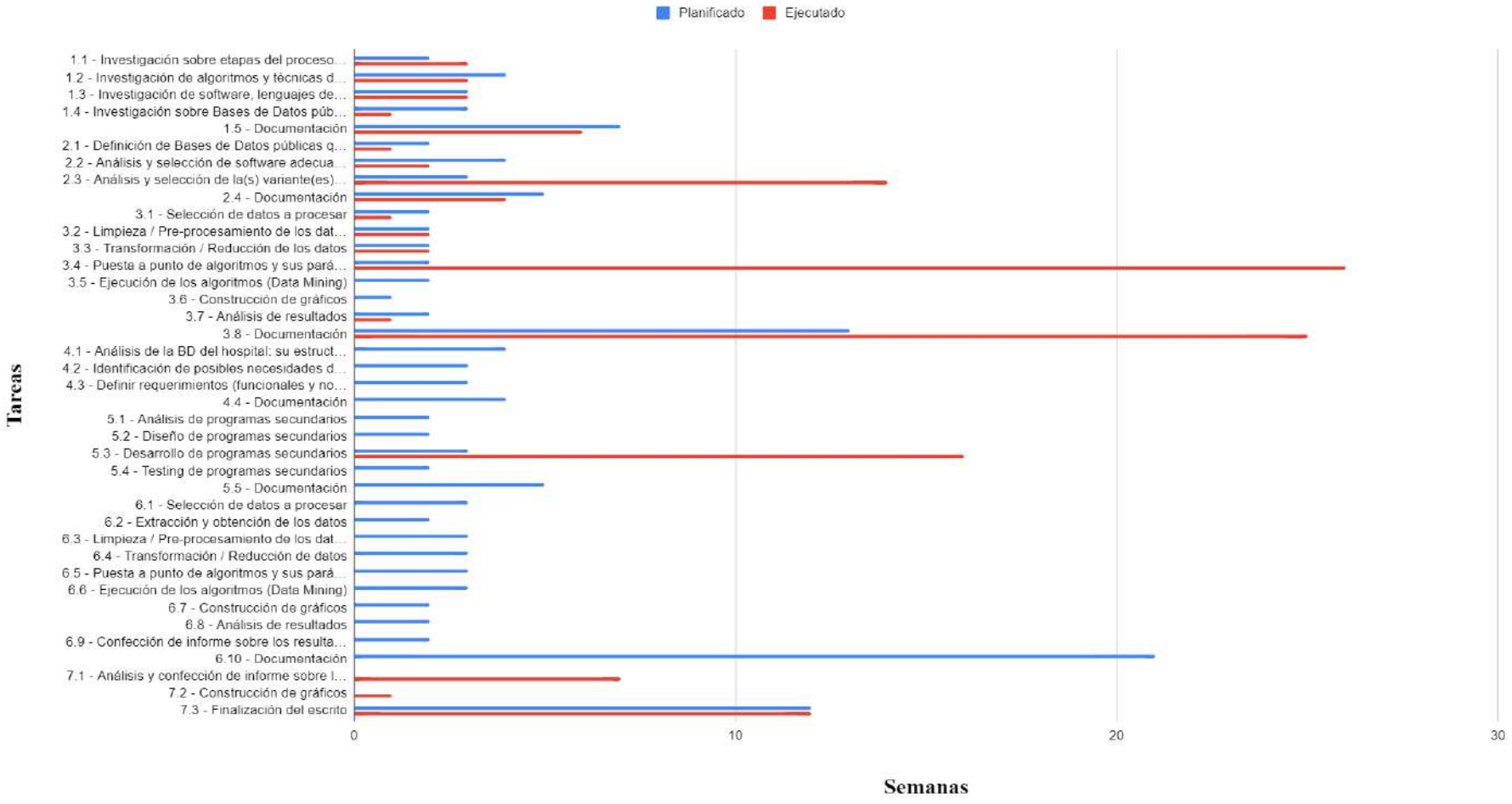
Gráfico de torta sobre la distribución de los tiempos de cada etapa del proyecto



Nota: los valores en el gráfico están expresados en horas (tiempo total empleado para el proyecto: 1050 horas).

Gráfico 4

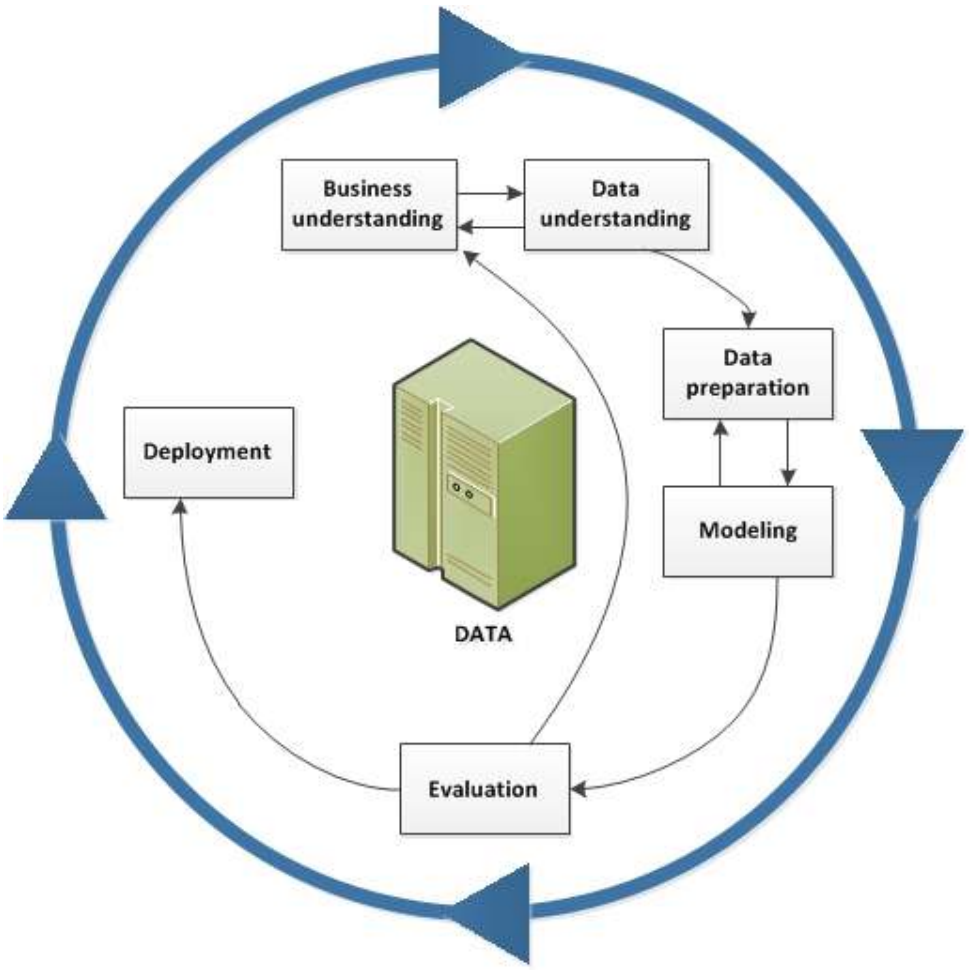
Gráfico de barras que compara la duración en semanas planificadas y las ejecutadas en cada una de las tareas del proyecto.



Durante el avance del proyecto resultó evidente que el solo uso del diagrama de Gantt como herramienta para estimar y gestionar un proyecto de minería de datos no es suficiente. El mismo no se adapta bien a los altos niveles de incertidumbre que se tiene hasta etapas avanzadas del proyecto por ser este orientado y severamente dependiente de los datos. Además, se siente algo rígido al no contemplar la iteración entre fases. Por estas razones se propone, de abordarse en el futuro un proyecto de características similares, utilizar algún enfoque alternativo que se adapte mejor a la naturaleza del proceso de minería de datos. Un ejemplo de esto sería el modelo conocido como CRISP-DM (*Cross-industry standard process for data mining*) [59], representado en la **Figura 80**.

Figura 80

Ciclo de vida de la minería de datos según CRISP-DM



Nota: tomado de https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

Este modelo, desconocido al inicio del proyecto, se encuentra entre los más utilizados en los procesos de minería de datos. Con el mismo pueden definirse tiempos estimados para cada fase y a su vez estimar una cantidad posible de iteraciones, ya sea entre fases o para un ciclo completo. Con eso se podría estimar tiempos mínimos y máximos del proyecto, los cuales dependen de los resultados que se obtengan en el medio. Esto es así por la fuerte dependencia que existe con los datos analizados, ya que no se sabe de antemano cómo será la respuesta de las técnicas y algoritmos frente a los datos presentados.

Este modelo puede adaptarse a este proyecto en particular integrando la fase de investigación con la de entendimiento del negocio y reemplazando la última fase por una de análisis de resultados y confección de gráficos. Además, podría utilizarse un enfoque mixto elaborando un diagrama de Gantt específico para cada fase. Naturalmente la estimación para los primeros ciclos será de mayor duración que los siguientes, ya que se irá ganando experiencia y la incertidumbre eventualmente irá disminuyendo.

7. Conclusiones y trabajos a futuro

En el presente trabajo final se presentó la aplicación de técnicas de descubrimiento de conocimiento (KDD) en bases de datos COVID-19 sobre una base de datos pública de Argentina, provista por el Ministerio de Salud, y otra de una institución privada de la ciudad de Mar del Plata, el HPC. En ambos casos los datos cubrieron el período de 2020 hasta mayo de 2022.

Se llevó a cabo una investigación de las etapas del KDD previa a su desarrollo, se hicieron pruebas y comparaciones entre distintos algoritmos y softwares para su selección y se generaron distintos subconjuntos de datos de interés para el análisis. En particular, en la etapa de *data mining*, se aplicaron técnicas de *clustering* incluyendo k-means, k-prototypes y mapas autoorganizados de Kohonen (SOM), siendo esta última una técnica de red neuronal artificial enmarcada en el paradigma de aprendizaje competitivo no supervisado. Por último, se combinaron las técnicas de SOM y k-means para mejorar los resultados en la etapa de formación de *clusters*.

Cabe destacar que el conjunto de datos utilizado incluyó datos numéricos y categóricos, lo cual incorporó cierta complejidad al problema, teniendo que adaptar datos y técnicas para su tratamiento. Además, la puesta a punto de los algoritmos utilizados fue uno de los mayores retos debido a la naturaleza de los datos. La dificultad para lograr grupos en los datos que pudieran ser utilizados para el análisis y potencial descubrimiento de información hizo que el tiempo requerido en ciertas etapas del proyecto se excediera de la planificación inicial. Siempre existió el deseo de incorporar otro tipo de datos (por ejemplo, datos clínicos de pacientes) y contar con la colaboración de un experto para darle más profundidad al trabajo, pero esto nunca estuvo en lo planificado ya que excedería en tiempo, esfuerzo y complejidad los alcances del proyecto.

Otro gran desafío fue la redacción del informe. Resultó difícil plasmar en un documento todo un proceso realizado que sea fácil de leer y de comprender con todas las pruebas, errores y vueltas a mismas etapas, siendo todas estas situaciones características de trabajos relacionados con el KDD.

En cuanto a lo aprendido de las herramientas utilizadas se puede destacar que el SOM es muy poderoso en sus posibilidades de clustering, análisis, representación y visualización de los resultados, posibilitando un análisis completo y profundo. Por otro lado, k-prototypes es una herramienta menos sofisticada pero que rápidamente puede presentar un panorama general de la distribución de las muestras.

Como parte del producto de este proyecto final, tal y como se detalla en los objetivos, se presenta la documentación y el conocimiento de todo un procedimiento de inicio a fin para la obtención potencial información útil de un conjunto de datos, pudiendo ser reproducido en otro con sus respectivas modificaciones.

Adicionalmente, se desarrolló un software que permite la visualización de los resultados obtenidos, en función de las variables de interés, *clusters* formados, cantidad de datos que activan cada neurona, entre otros, para que posteriormente los profesionales vinculados al área de la Salud puedan utilizarlo para la toma de decisiones en caso de reunir contenido útil según su conocimiento y experiencia. Además, SOM tiene el potencial de ser utilizado como una herramienta de clasificación. De esta manera, se podría predecir el comportamiento de casos de entrada mediante las características de los clusters ya formados.

Si bien KDD es una disciplina altamente compleja y teniendo en cuenta el alcance de un proyecto de grado, se pudieron cumplir la totalidad de los objetivos propuestos debido a que:

- Se incorporó una metodología para la gestión y desarrollo del proyecto mediante una planificación de tareas, definición de objetivos y alcances y

asignación de recursos. A esto se suma el control periódico a través de reuniones con los directores, donde se realizaba un monitoreo del progreso y evaluación de los resultados.

- Se ha llevado a cabo una investigación dedicada de grandes volúmenes de datos en el contexto de la pandemia y la gestión de la salud mediante el análisis de bases de datos, selección de atributos y aplicación de diversas técnicas de clustering, permitiendo evaluar su eficacia y eficiencia.

- Se seleccionó un software adecuado para la aplicación de las técnicas de clustering estudiadas, teniendo en cuenta sus características y funcionalidades. Con el mismo optimizaron los parámetros utilizados en las técnicas, y se han presentado, analizado y comparado los resultados obtenidos.

A modo personal se puede destacar el enorme aprendizaje adquirido de todo el proceso de extracción de conocimiento. Fue una experiencia enriquecedora haber abordado un proyecto en el que no se tenía ningún conocimiento previo del proceso de minería de datos. Además, el hecho de tener que trabajar con un conjunto limitado de datos, en un tiempo relativamente corto y con una temática compleja, propuso un desafío interesante. Específicamente, puede mencionarse una ganancia de aprendizaje en:

- Conceptos, comportamiento e implementación práctica de algoritmos de agrupamiento en minería de datos, partiendo de técnicas más simples como k-means y analizando otras más avanzadas tal como SOM, junto con sus respectivas variantes.
- Manejo de librerías de terceros que implementan técnicas de minería de datos. En muchas ocasiones se tuvo que dedicar tiempo a analizar la implementación de ciertas funciones para poder modificarlas y adaptarlas según necesidades.

- Manipulación de datos, tal como los procedimientos de limpieza y normalización y la capacidad de poder seleccionar atributos relevantes y generar nuevos a partir de atributos ya existentes.
- Gestión de entornos de ejecución para correr distintos paquetes y librerías de forma eficiente. A esto se suma el seguimiento y control de versiones de los scripts, lo que posibilitó realizar de forma más ordenada la ejecución de las distintas variantes de los algoritmos.
- La capacidad de realizar análisis profundos de artículos científicos complejos y desarrollar un juicio para evaluar la factibilidad de su aplicación o no en el proyecto.
- Elección de técnicas y software en base a criterios como la disponibilidad de recursos, complejidad y adaptabilidad.

Con respecto a los trabajos a futuro pueden realizarse ciertas modificaciones e implementaciones para seguir mejorando los resultados obtenidos y así obtener potencial información novedosa y útil. Algunas de estas propuestas son:

1. Implementar un SOM que pueda procesar datos numéricos y categóricos sin tener que realizar un proceso de binarización. Sería interesante analizar la formación de grupos que no están mayormente influenciados por la numerización de atributos nominales.

2. Tener en cuenta otros atributos que sean de interés para el entrenamiento de los mapas. Por ejemplo, se podrían utilizar datos relacionados con la aplicación de las vacunas existentes (no se disponía de esta información al comienzo del proyecto).

3. Agregar nuevas funcionalidades al software desarrollado de forma tal que se puedan cargar otros *datasets* y mapas y analizar sus resultados.

4. Sumar a un experto del área de salud para que participe activamente en las mejoras previamente mencionadas.

Bibliografía

[1] Monserrat, S. y Chiotti, O. (2013). Minería de Datos en Base de Datos de Servicios de Salud. *Congreso Nacional de Ingeniería Informática y Sistemas de Información.* http://conaiisi.frc.utn.edu.ar/PDFsParaPublicar/1/schedConfs/1/132-505-1-DR.pdf

[2] Hernández Orallo, J., Ramírez Quintana, M. J. y Ferri Ramírez, C. (2004). *Introducción a la Minería de Datos*. Pearson Prentice Hall.

[3] Demsar J., Curk T., Erjavec A., Gorup C., Hocevar T., Milutinovic M., Mozina M., Polajnar M., Toplak M., Staric A., Stajdohar M., Umek L., Zagar L, Zbontar J., Zitnik M., Zupan B. (2013). Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research.* https://orangedatamining.com/

[4] de Vos, N. J. (2015-2022). *kmodes categorical clustering library*. https://github.com/nicodv/kmodes

[5] Engardt S. (2018). *Unsupervised learning with mixed type data for detecting money laundering* [Tesis de Grado, Institute Of Technology KTH Royal, School of Electrical Engineering and Computer Science].

[6] Nieminen, I. (2012-2018). *SOMToolbox*. https://github.com/ilarinieminen/SOM-Toolbox

[7] Alhoniemi, E., Himberg, J., Parhankangas, J., Vesanto, J. (2000-2005). *SOMToolbox*. http://www.cis.hut.fi/somtoolbox/about.shtml

[8] (2022) *Plotly Open Source Graphing Library for Python*. https://plotly.com/python/

[9] (2022) *Git*. https://git-scm.com/

[10] Abdul Nazeer, K. A., Sebastian, M. P. (2009). Improving the Accuracy and Efficiency of the k-means Clustering Algorithm. *World Congress on Engineering 2009*, *Vol 1*. Newswood Limited.

[11] Vesanto, J., Alhoniemi, E. (2000). *Clustering of the Self-Organizing Map*. IEEE Transactions on Neural Networks, Vol. 11, No. 3.

[12] Huang, Z. (1998). *Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values*. Data Mining and Knowledge Discovery 2.

[13] Kohonen, T. (2000). *Self-Organizing Maps, 3rd Edition*. Springer.

[14] Rodríguez, M. D. (2000). *Mapas Autoorganizados para Clasificación en Tiempo Real: Implementación digital sobre FPGAs* [Tesis de Grado, Universidad del País Vasco, Facultad de Ciencia y Tecnología]

[15] Vesanto, J. , Himberg, J., Alhoniemi, E., Parhankangas, J. (2000). *SOM Toolbox for Matlab 5*. http://www.cis.hut.fi/somtoolbox/package/papers/techrep.pdf

[16] Seijas, L. M. (2002). *Reconocimiento de Dígitos Manuscritos Mediante Redes Neuronales: Una Técnica Híbrida*. [Tesis de Licenciatura, Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales].

[17] Ultsch, A., Siemon, H. P. (1990). Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis. *Proceedings of the International Neural Network Conference (INNC-90)*

[18] Umargono, E., Suseno, J., Gunawan, S. K. (2020). K-Means Clustering Optimization Using the Elbow Method and Early Centroid Determination Based on Mean and Median Formula. *Conference: The 2nd International Seminar on Science and Technology.* https://www.scitepress.org/Papers/2019/99084/99084.pdf

[19] Rousseeuw, P. J. (1987). Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis, *Journal of Computational and Applied Mathematics, Vol 20.* https://www.sciencedirect.com/science/article/pii/0377042787901257

[20] Davies, D. L., Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2.* https://ieeexplore.ieee.org/document/4766909

[21] Breard, G. T. (2017). *Evaluating Self-Organizing Map Quality Measures as Convergence Criteria* [Tesis de Grado, University of Rhode Island]. https://digitalcommons.uri.edu/theses/1033

[22] Bezdek, J., Ehrlich, R., Full, W. (1984). FCM—the Fuzzy C-Means clustering-algorithm. *Computers & Geosciences*. https://www.researchgate.net/publication/222868331_FCM-the_Fuzzy_C-Means_clustering-algorithm

[23] Zhexue Huang, Michael K. Ng (1999). A fuzzy k-modes Algorithm for Clustering Categorical Data. *IEEE Transactions on Fuzzy Systems, 1999, Vol. 7 No. 4.* https://hub.hku.hk/handle/10722/42992

[24] Ji, Jinchao, Pang, Wei, Zhou, Chunguang, Han, Xiao, Wang, Zhe. (2013). A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data. *Knowledge-Based Systems*. https://www.researchgate.net/publication/257391282_A_fuzzy_k-prototype_clustering_algorithm_for_mixed_numeric_and_categorical_data

[25] Arthur, D., Vassilvitskii, S. (2007). K-means++: The Advantages of Careful Seeding. *SODA '07*. https://api.semanticscholar.org/CorpusID:1782131

[26] Schubert, E., Rousseeuw, P. J. (2021). Fast and Eager k-medoids Clustering: O(k) Runtime Improvement of the PAM, CLARA, and CLARANS Algorithms. *Information Systems, Vol. 101*. https://doi.org/10.1016/j.is.2021.101804.

[27] Dasgupta, S., Frost, N., Moshkovitz, M., Rashtchian, C. (2020). Explainable k-Means and k-Medians Clustering. *ICML*. https://api.semanticscholar.org/CorpusID:211572790

[28] Hornik, K., Feinerer, I., Kober, M., & Buchta, C. (2012). Spherical k-Means Clustering. *Journal of Statistical Software*. https://doi.org/10.18637/jss.v050.i10

[29] Pelleg, D., Moore, A. (2002). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. *Machine Learning, p*. https://www.researchgate.net/publication/2532744_X-means_Extending_K-means_with_Efficient_Estimation_of_the_Number_of_Clusters

[30] Zhao, Z., Guo, S., Xu, Q., Ban, T. (2009). G-Means: A Clustering Algorithm for Intrusion Detection. *In: Köppen, M., Kasabov, N., Coghill, G. (eds) Advances in Neuro-Information Processing. ICONIP 2008. Lecture Notes in Computer Science, Vol 5506.* https://doi.org/10.1007/978-3-642-02490-0_69

[31] Coso, C., Fustes, D., Dafonte, C., Nóvoa, F., Rodríguez-Pedreira, J., Arcay, B. (2015). Mixing Numerical and Categorical Data in a Self-Organizing Map by Means of Frequency Neurons. *Applied Soft Computing*. https://www.researchgate.net/publication/282562812_Mixing_numerical_and_categorical_data_in_a_Self-Organizing_Map_by_means_of_frequency_neurons

[32] Chung-Chian, Hsu. (2006). Generalizing Self-Organizing Map for Categorical Data. *IEEE Transactions on Neural Networks, Vol. 17, No. 2.* https://www.comp.nus.edu.sg/~rudys/arnie/tnn-som-categoricaldata.pdf

[33] Frank, E., Hall, M. A., Witten, I. H. (2016). *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Morgan Kaufmann, Fourth Edition.

[34] Berthold, M. R. et al. (2008). KNIME: The Konstanz Information Miner. *Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization*. Springer.

[35] (2022) RapidMiner. https://rapidminer.com

[36] (2022) SAS Enterprise Miner. https://www.sas.com/es_ar/software/enterprise-miner.html

[37] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Vol. 1*.

[38] Steinhaus, H. (1957). Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*.

[39] Lloyd, S. P. (1957). Least square quantization in PCM. *Bell Telephone Laboratories Paper*.

[40] Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*.

[41] Cao, F., Liang, J, Bai, L. (2009). A new initialization method for categorical data clustering. *Expert Systems with Applications*.

[42] Liu, Y., Weisberg, R. (2011). A Review of Self-Organizing Map Applications in Meteorology and Oceanography. *Self Organizing Maps - Applications and Novel Algorithm Design*.
https://www.researchgate.net/publication/221910351_A_Review_of_Self-Organizing_Map_Applications_in_Meteorology_and_Oceanography

[43] Leys, C., et al. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology.*
https://www.researchgate.net/publication/256752600_Detecting_outliers_Do_not_use_standard_deviation_around_the_mean_use_absolute_deviation_around_the_median

[44] Velázquez, G. (2008). Las regionalizaciones argentinas: evolución de su capacidad de discriminación del bienestar de la población. *GeoFocus.*

[45] Haykin, S. (2011). *Neural Networks and Learning Machines, Third Edition*. Pearson.

[46] (2022) UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets.php

[47] (2022) Java SOMToolbox. http://www.ifs.tuwien.ac.at/dm/somtoolbox/datasets.html

[48] (2022) Ministerio de Salud Argentina.
http://datos.salud.gob.ar/dataset/covid-19-casos-registrados-en-la-republica-argentina

[49] Riese, F., Keller, S., Hinz, S. (2019). Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data. *Remote Sensing*.
https://www.researchgate.net/publication/338122250_Supervised_and_Semi-Supervised_Self-Organizing_Maps_for_Regression_and_Classification_Focusing_on_Hyperspectral_Data

[50] Kohonen, T. (2014). *MATLAB Implementations and Applications of the Self-Organizing Map*. Unigrafia Oy, Helsinki, Finland.

[51] Sulkava, M., Yli-Heikkilä, M., Latukka, A. (2013). Analysis of Farm Profitability and the Weighted Upscaling System Using the Self-Organizing Map. *Advances in Self-Organizing Maps*.

[52] Cottrell, M., Olteanu, M., Rossi, F., Vialaneix, N. (2016). Theoretical and Applied Aspects of the Self-Organizing Maps. *WSOM 2016*.
https://hal.archives-ouvertes.fr/hal-01270701

[53] MathWorks, Inc. (1996). *MATLAB: the language of technical computing: computation, visualization, programming: installation guide for UNIX version 5*. Natwick: Math Works Inc.

[54] Google LLC. (2022). *Google Docs*. https://www.google.com/docs/about/

[55] Google LLC. (2022). *Google Drive*. https://drive.google.com

[56] Github. (2022). *GitHub*. https://github.com/

[57] Kaufman, L., Rousseeuw, P. (2009). *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley.

[58] Fayyad et al. (1996). *Knowledge Discovery and Data Mining: Towards a Unifying Framework.*

[59] Shearer C. (2000). The CRISP-DM model: the new blueprint for data mining, *J Data Warehousing*.

Anexo 1

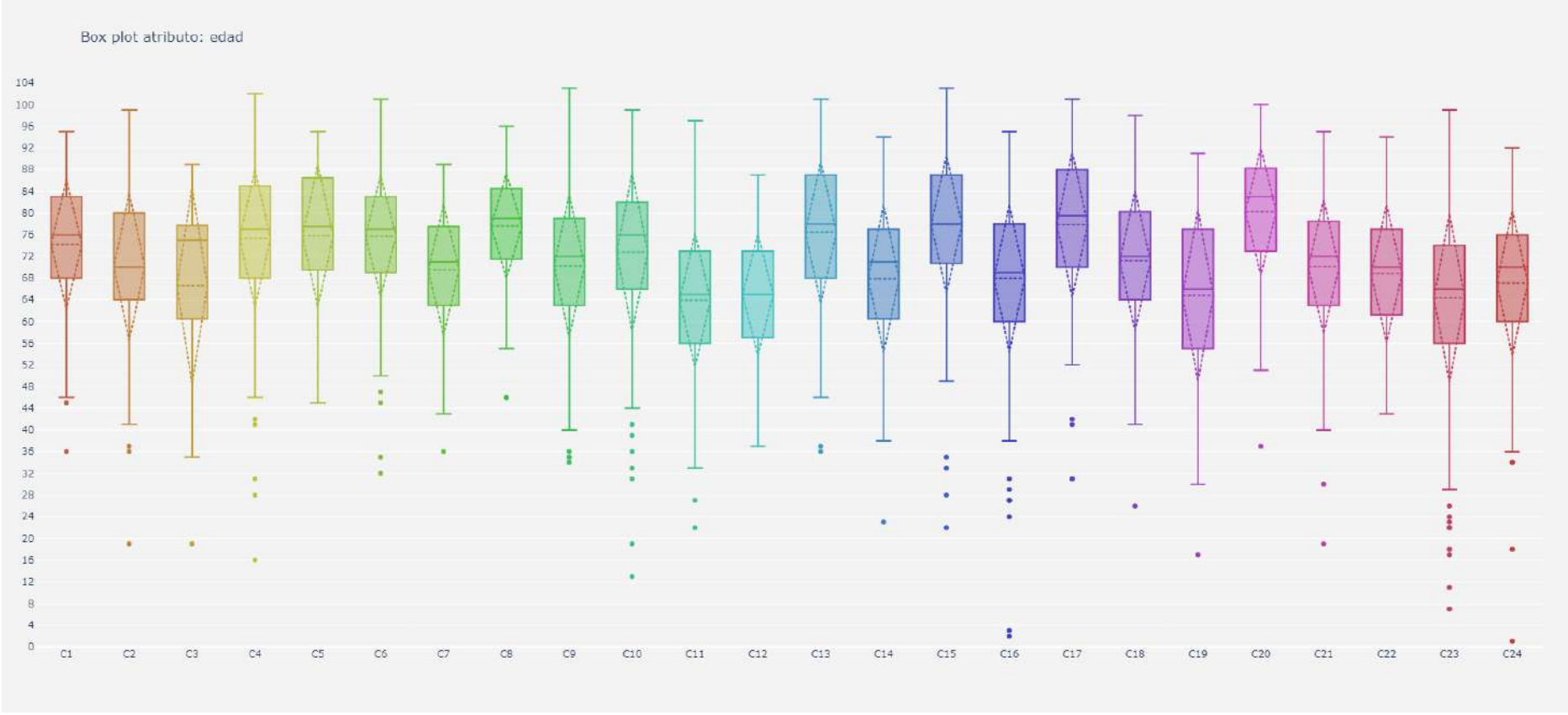
Gráficos Estadísticos

Índice

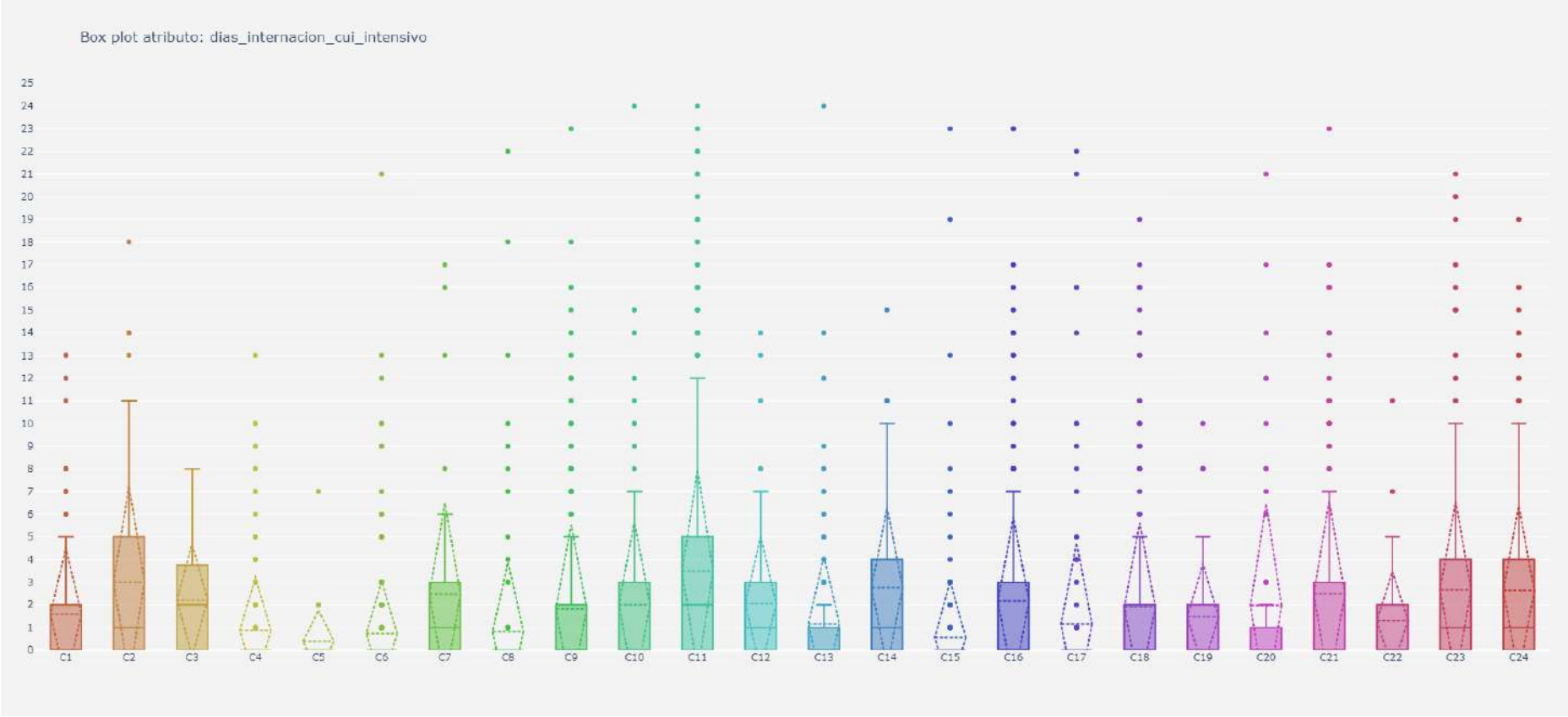
Dataset 1

Atributos numéricos

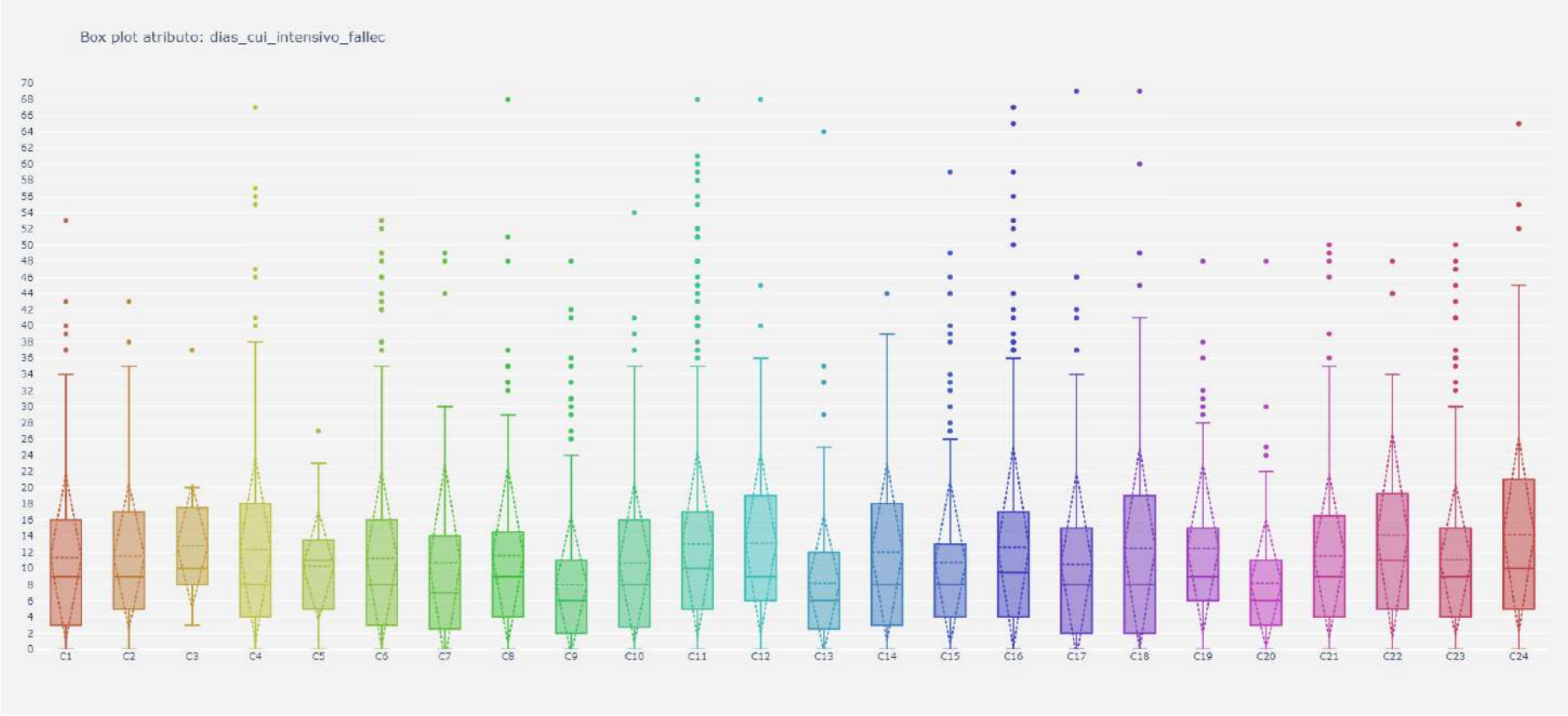
Edad



Dias_internacion_cui_intensivo

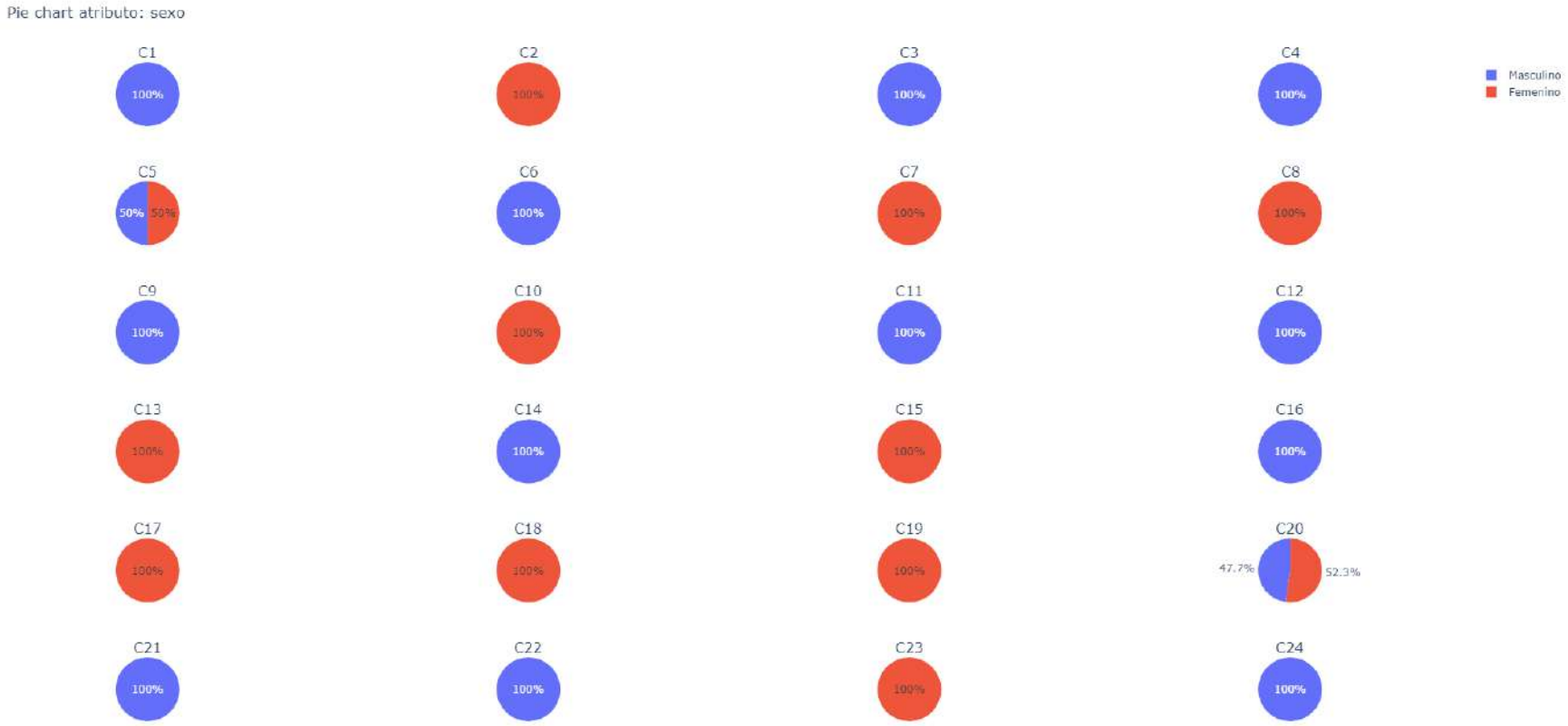


Dias_cui_intensivo_fallec

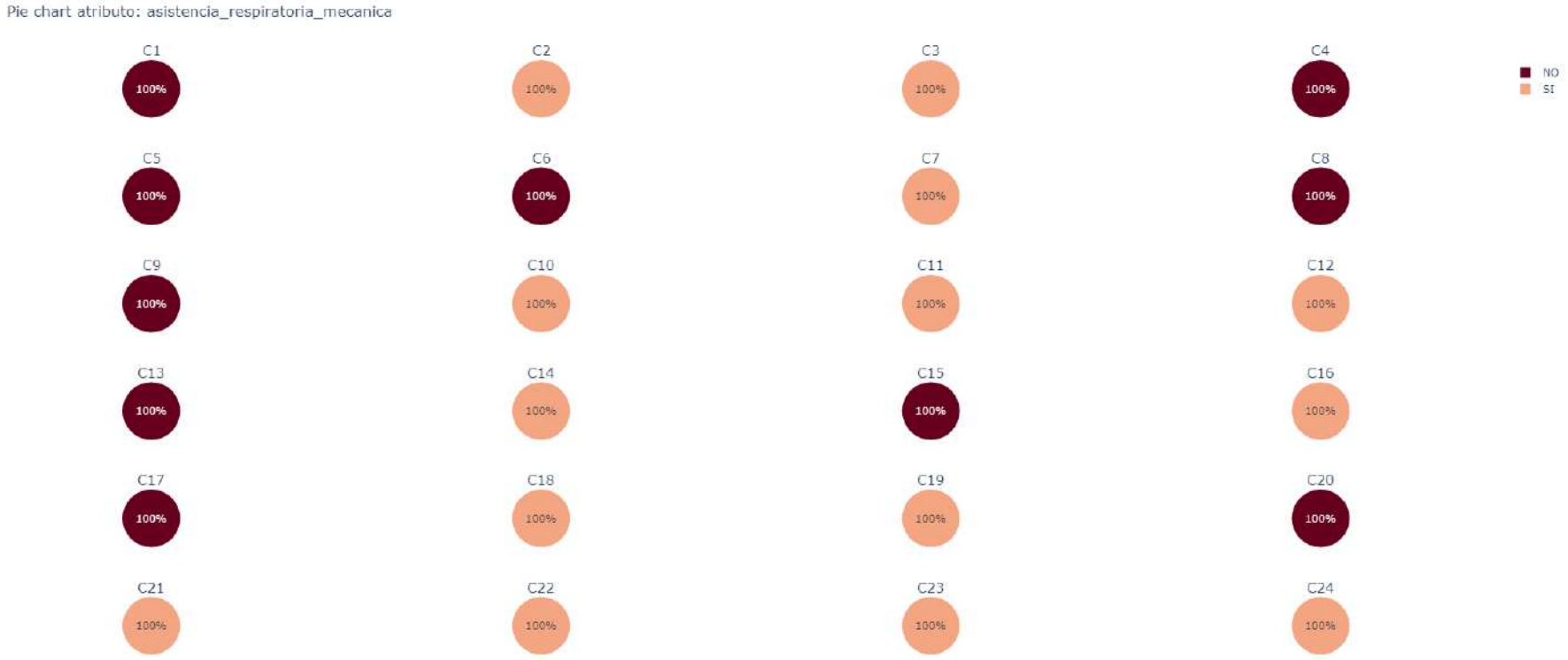


Atributos categóricos

Sexo



Asistencia_respiratoria_mecanica



Estacion


Pie chart atributo: estacion
C1
26.6%
73.4%
C2
8.33%
91.7%
C3
100%
C4
23.2%
76.8%
Verano
Invierno
Primavera
Otoño
C5
100%
C6
100%
C7
100%
C8
26.5%
73.5%
C9
100%
C10
22.1%
5.37%
72.5%
C11
1.73%
98.3%
C12
100%
C13
3.05%
97%
C14
100%
C15
100%
C16
100%
C17
29.1%
70.9%
C18
100%
C19
100%
C20
100%
C21
100%
C22
100%
C23
100%
C24
100%

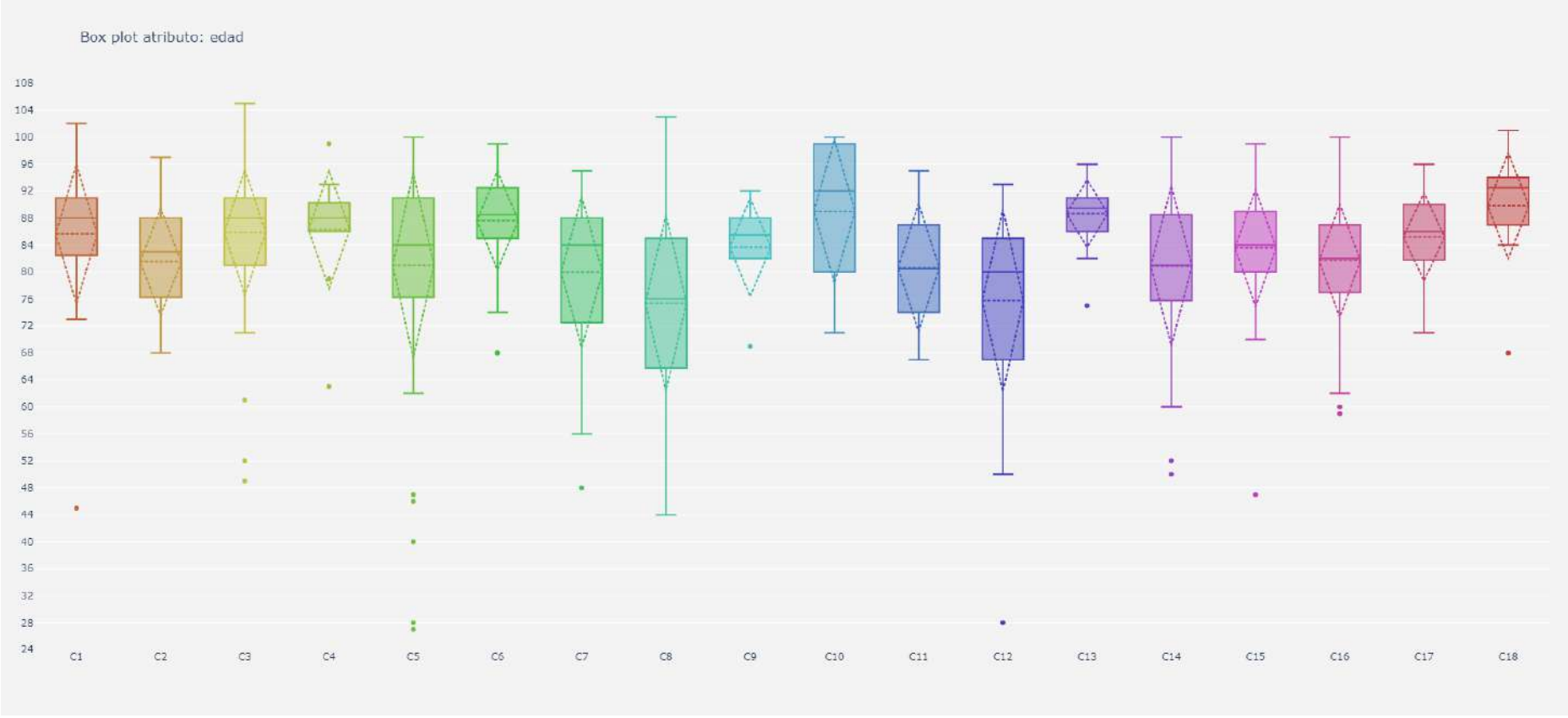
Año


Pie chart atributo: año
C1
100%
C2
100%
C3
100%
3.86%
C4
96.1%
2021
2020
2022
C5
100%
C6
100%
C7
100%
C8
100%
0.435%
C9
99.6%
C10
100%
C11
100%
C12
100%
C13
100%
C14
100%
C15
100%
C16
100%
C17
100%
C18
100%
C19
100%
C20
100%
C21
100%
4.26%
C22
95.7%
C23
100%
C24
100%

Dataset 2

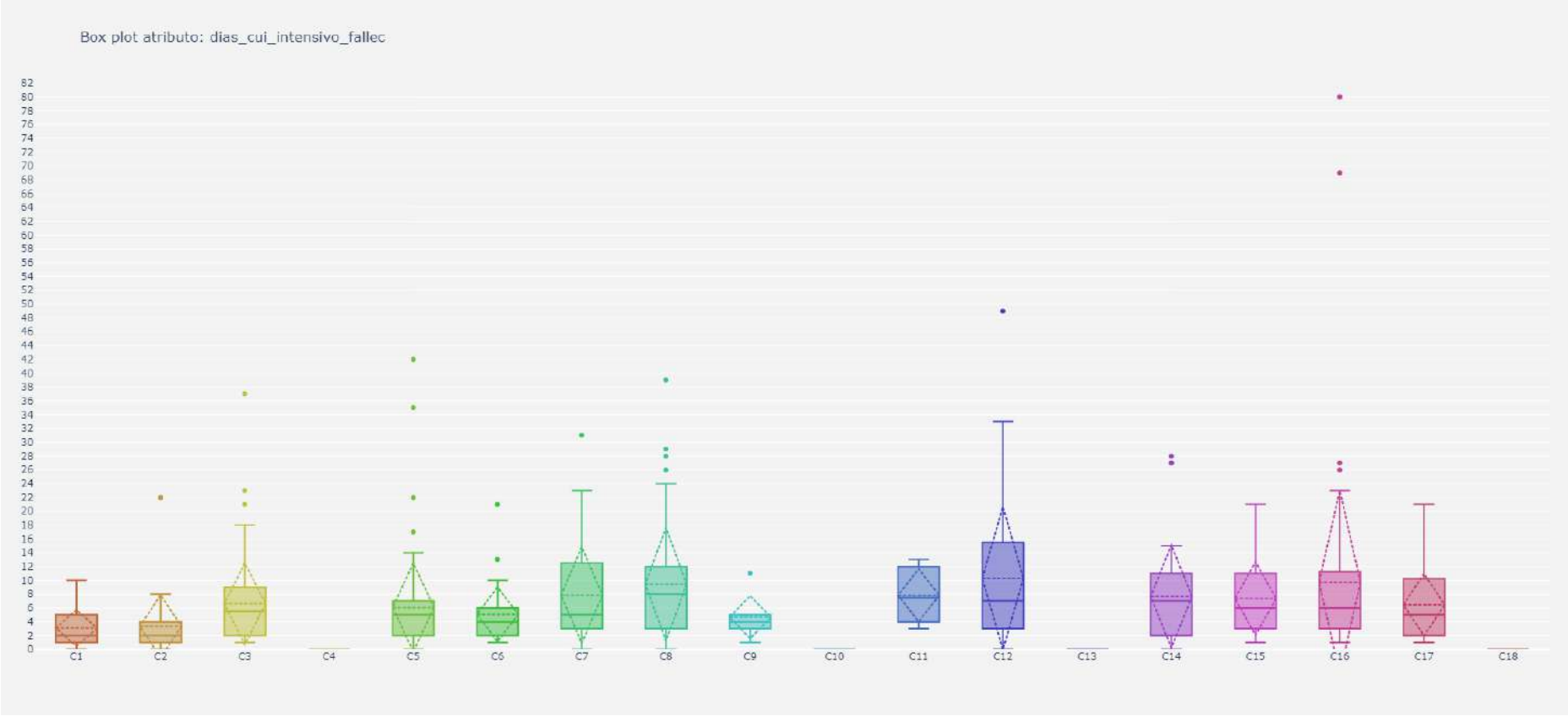
Atributos numéricos

Edad



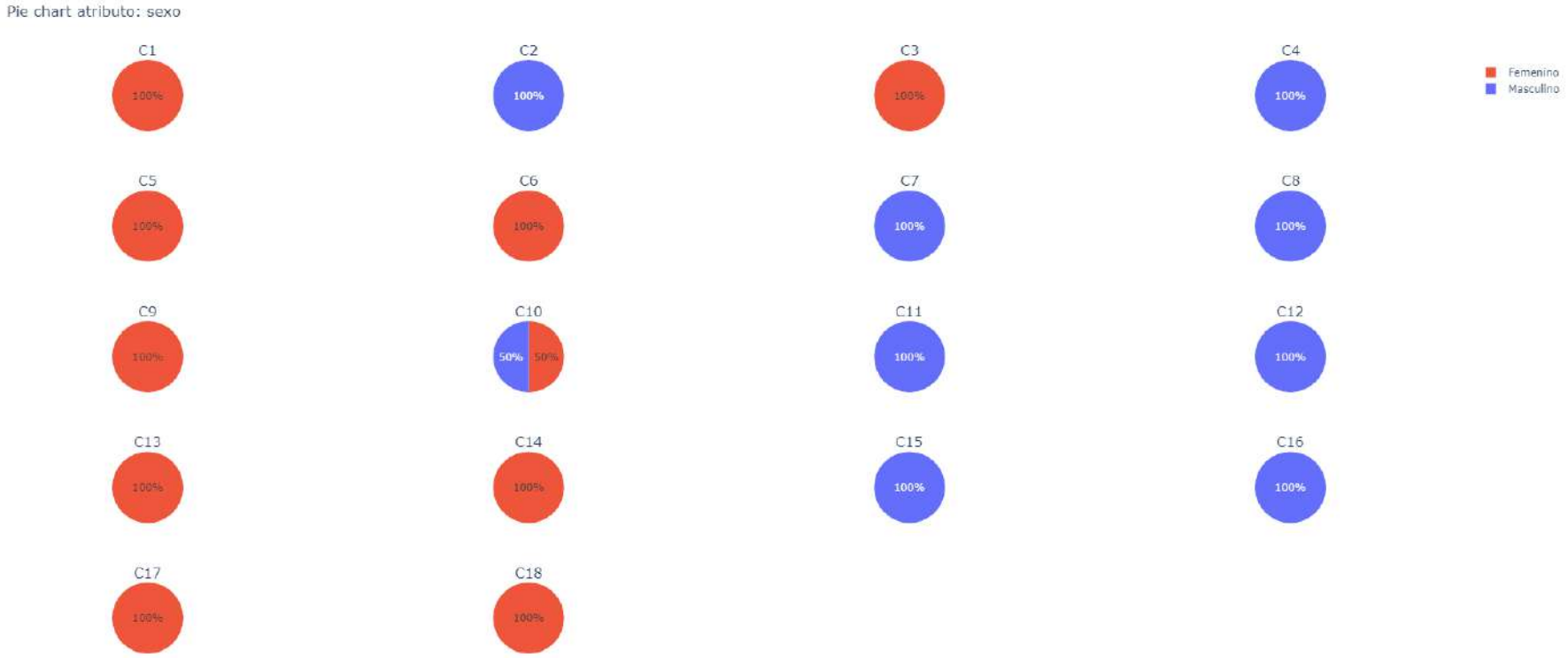
Dias_internacion_cui_intensivo

Dias_cui_intensivo_fallec

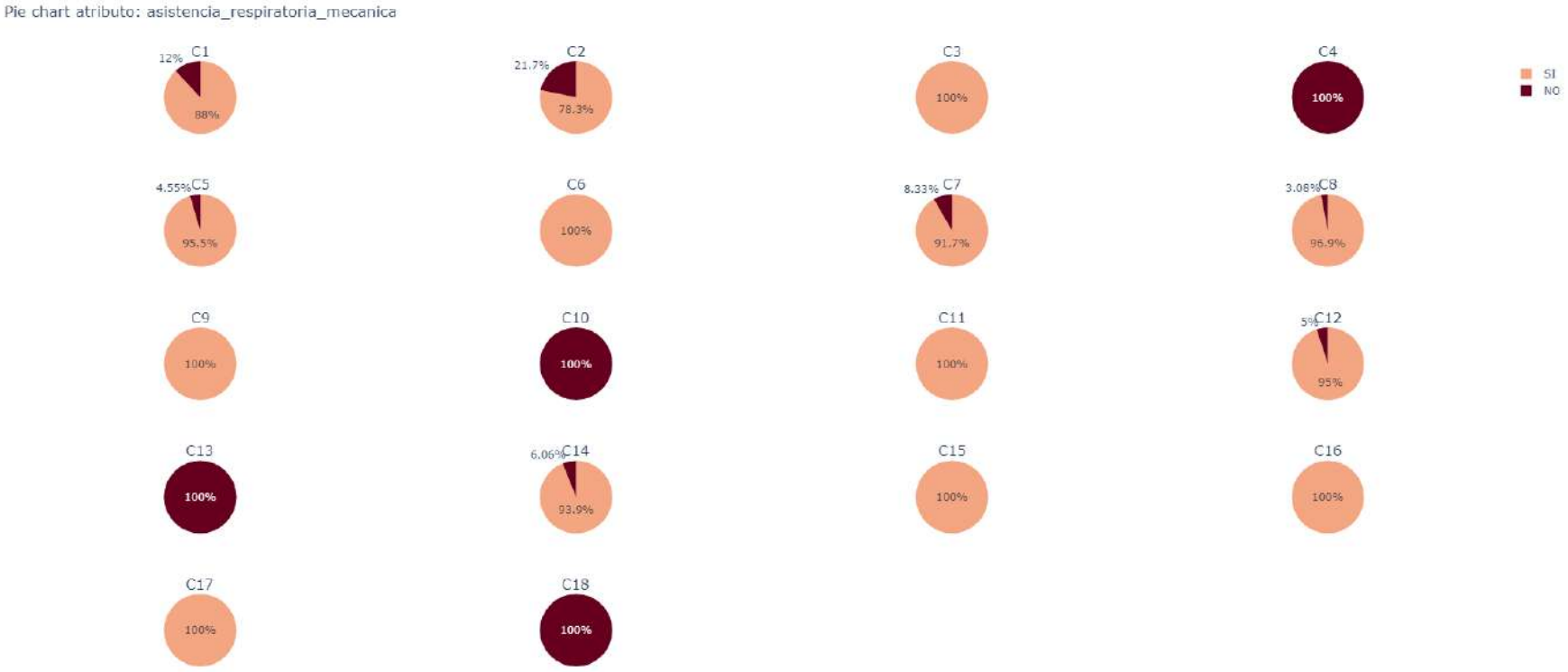


Atributos categóricos

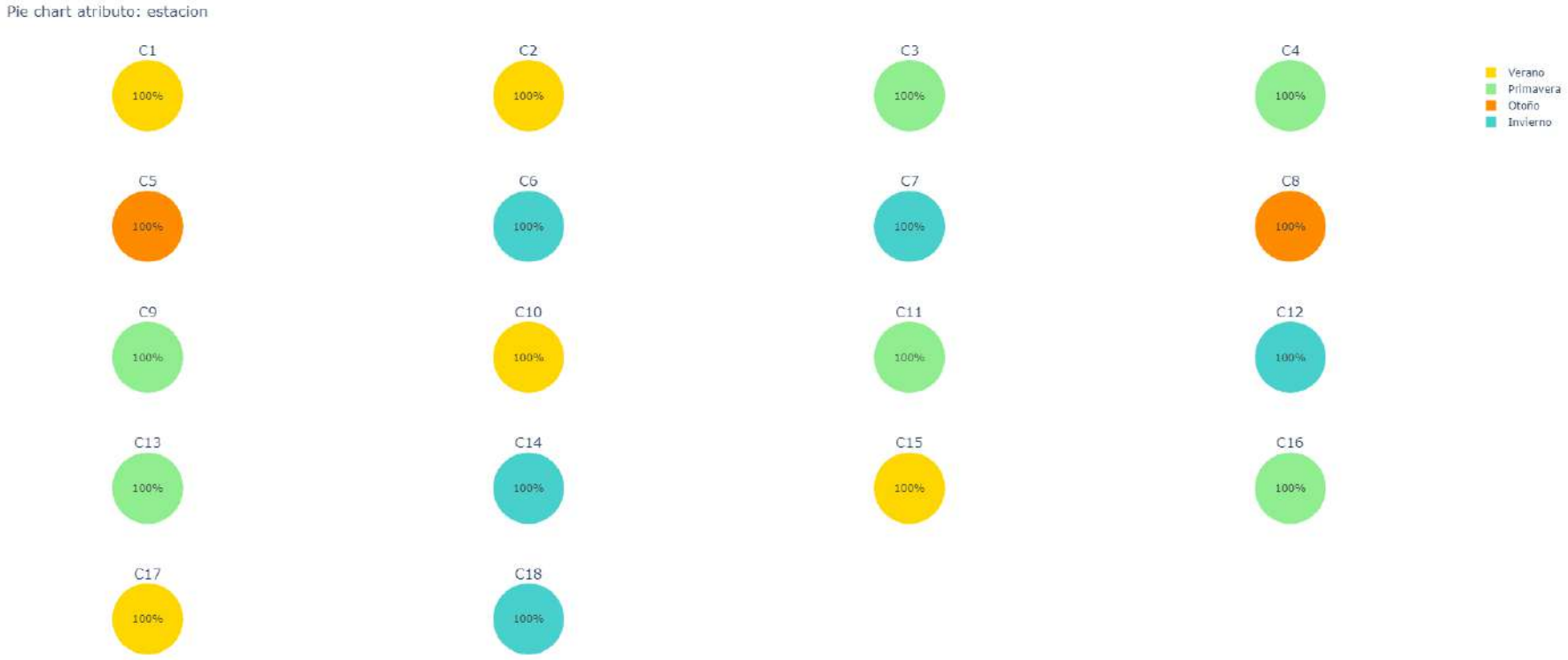
Sexo



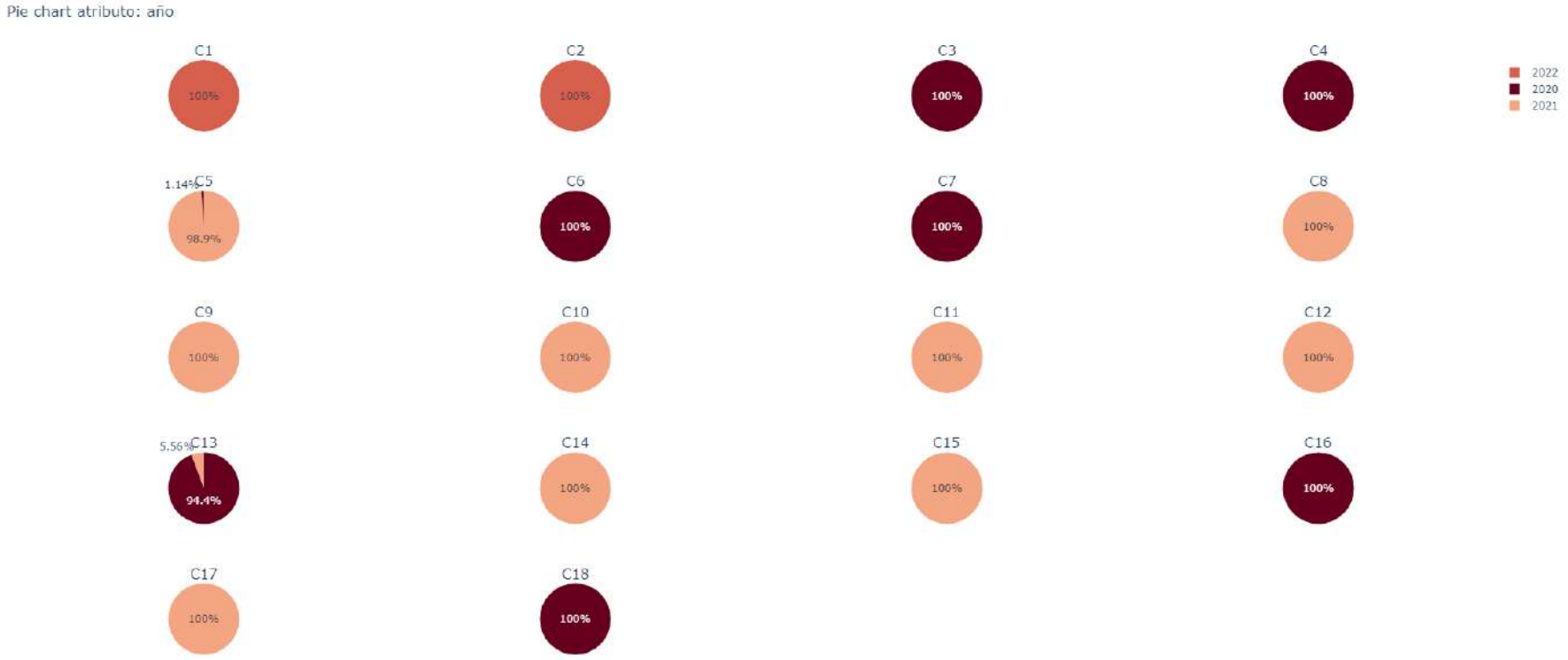
Asistencia_respiratoria_mecanica



Estacion


Pie chart atributo: estacion
C1
100%
C2
100%
C3
100%
C4
100%
Verano
Primavera
Otoño
Invierno
C5
100%
C6
100%
C7
100%
C8
100%
C9
100%
C10
100%
C11
100%
C12
100%
C13
100%
C14
100%
C15
100%
C16
100%
C17
100%
C18
100%

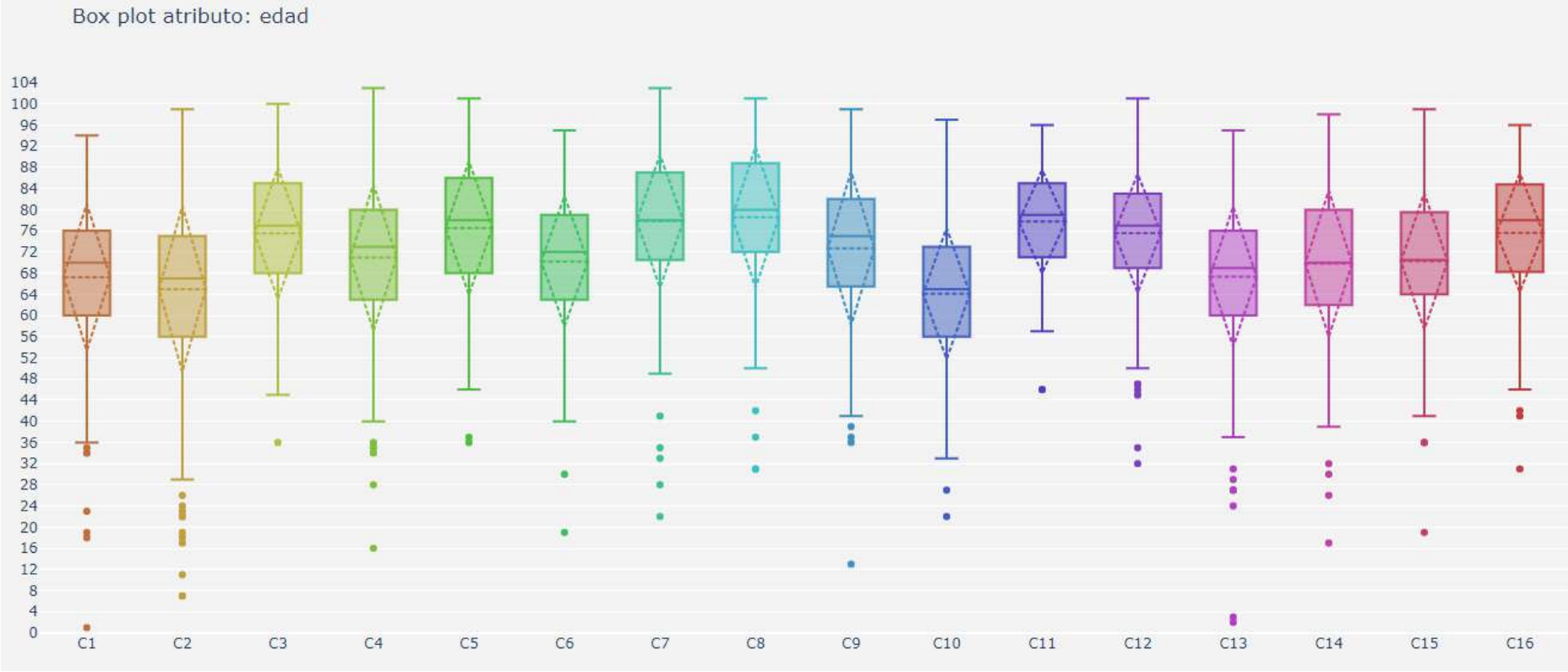
Año


Pie chart atributo: año
C1
100%
C2
100%
C3
100%
C4
100%
2022
2020
2021
1.14%
C5
98.9%
C6
100%
C7
100%
C8
100%
C9
100%
C10
100%
C11
100%
C12
100%
5.56%
C13
94.4%
C14
100%
C15
100%
C16
100%
C17
100%
C18
100%

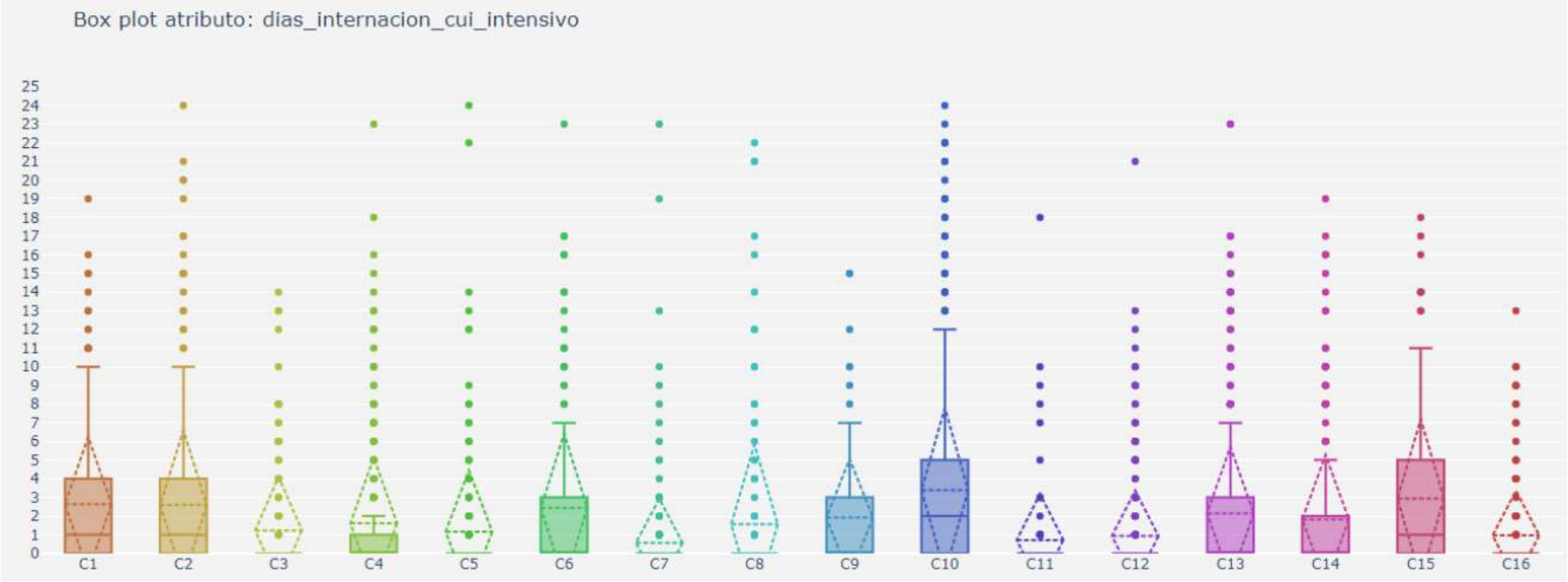
Dataset 3

Atributos numéricos

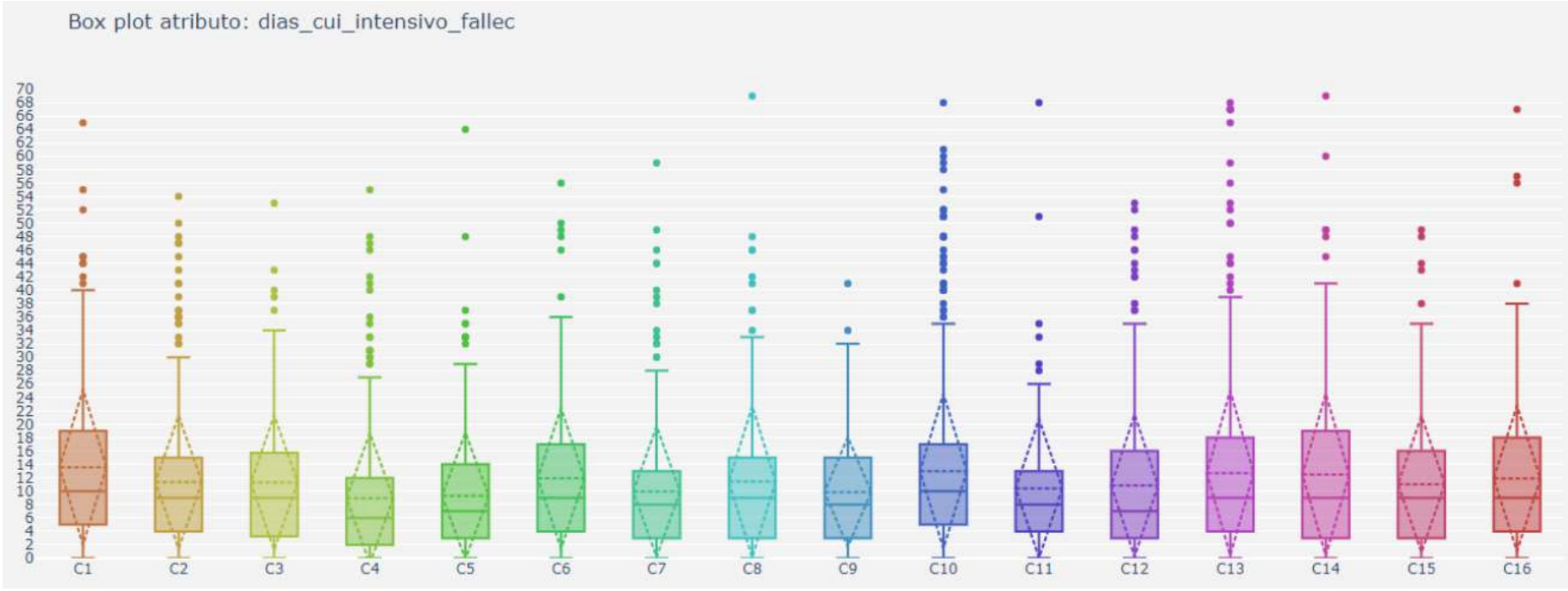
Edad



Dias_internacion_cui_intensivo



Dias_cui_intensivo_fallec

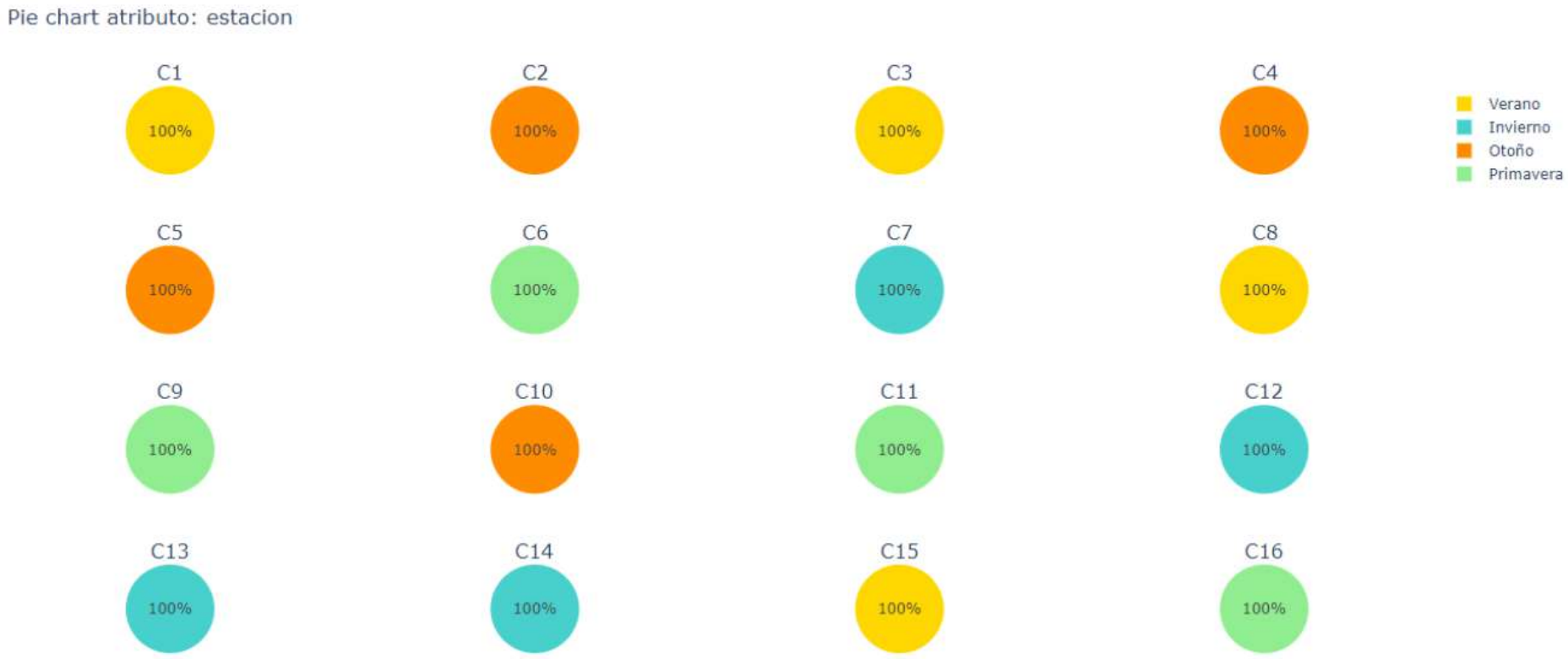


Atributos categóricos

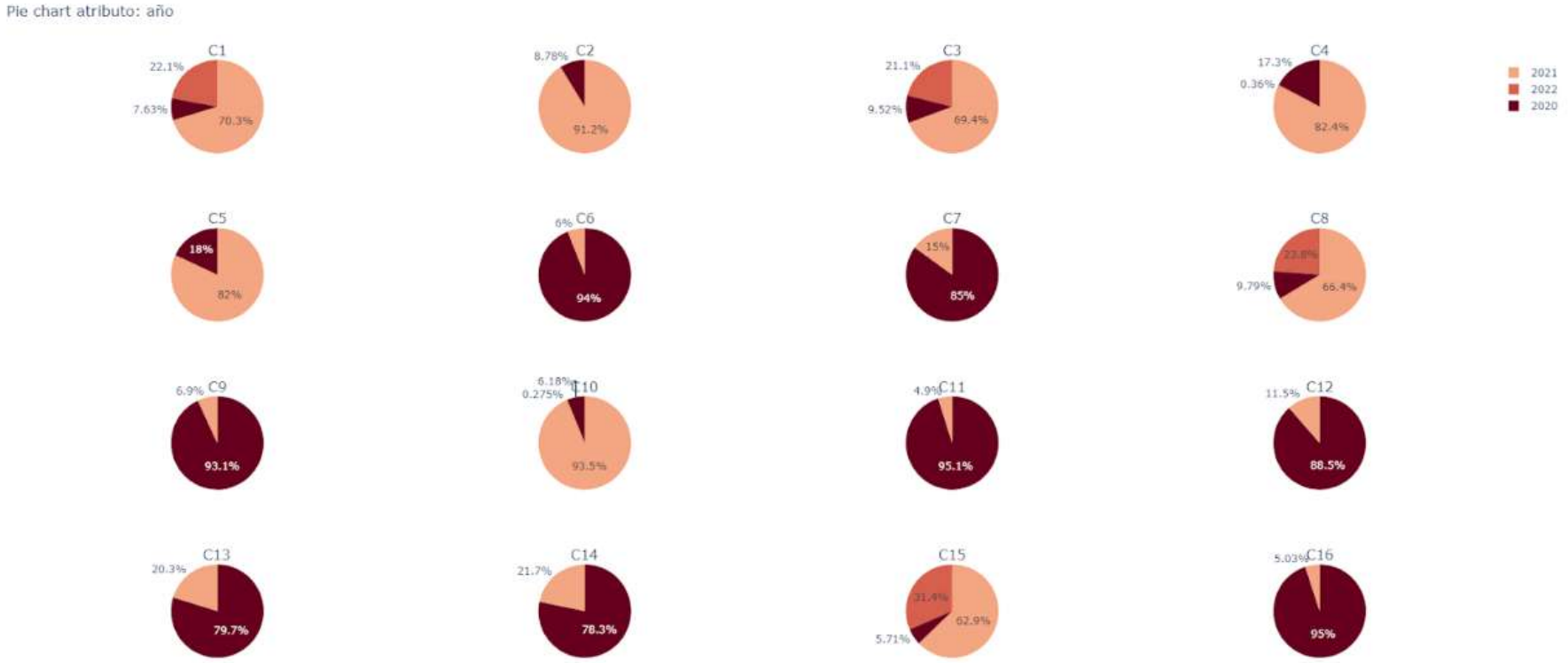
Sexo

Asistencia_respiratoria_mecanica

Estacion


Pie chart atributo: estacion
C1
100%
C2
100%
C3
100%
C4
100%
Verano
Invierno
Otoño
Primavera
C5
100%
C6
100%
C7
100%
C8
100%
C9
100%
C10
100%
C11
100%
C12
100%
C13
100%
C14
100%
C15
100%
C16
100%

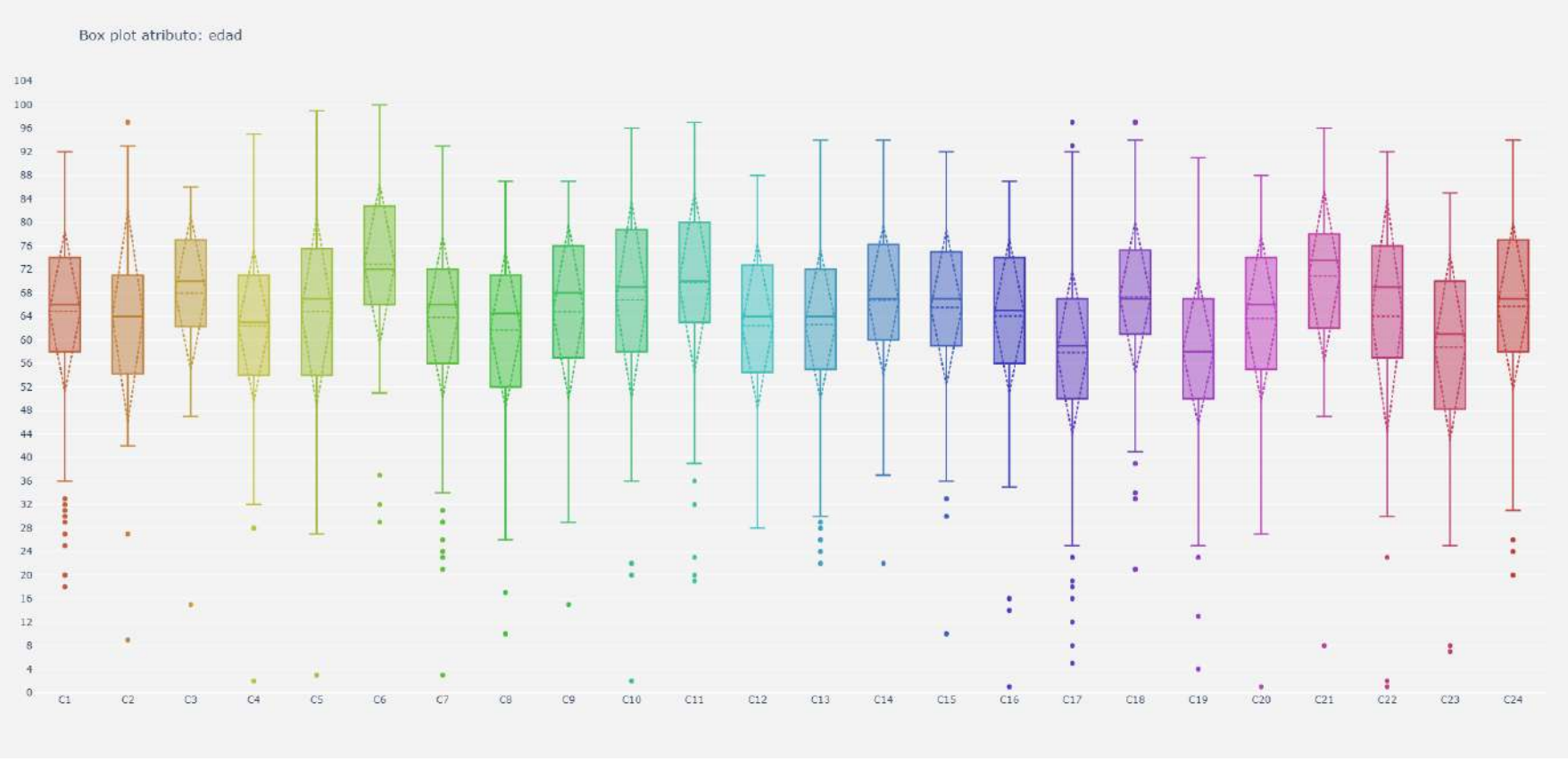
Año


Pie chart atributo: año
C1
22.1%
7.63%
70.3%
C2
8.78%
91.2%
C3
21.1%
9.52%
69.4%
C4
17.3%
0.36%
82.4%
2021
2022
2020
C5
18%
82%
C6
6%
94%
C7
15%
85%
C8
23.8%
9.79%
66.4%
C9
6.9%
93.1%
C10
6.18%
0.275%
93.5%
C11
4.9%
95.1%
C12
11.5%
88.5%
C13
20.3%
79.7%
C14
21.7%
78.3%
C15
31.4%
5.71%
62.9%
C16
5.03%
95%

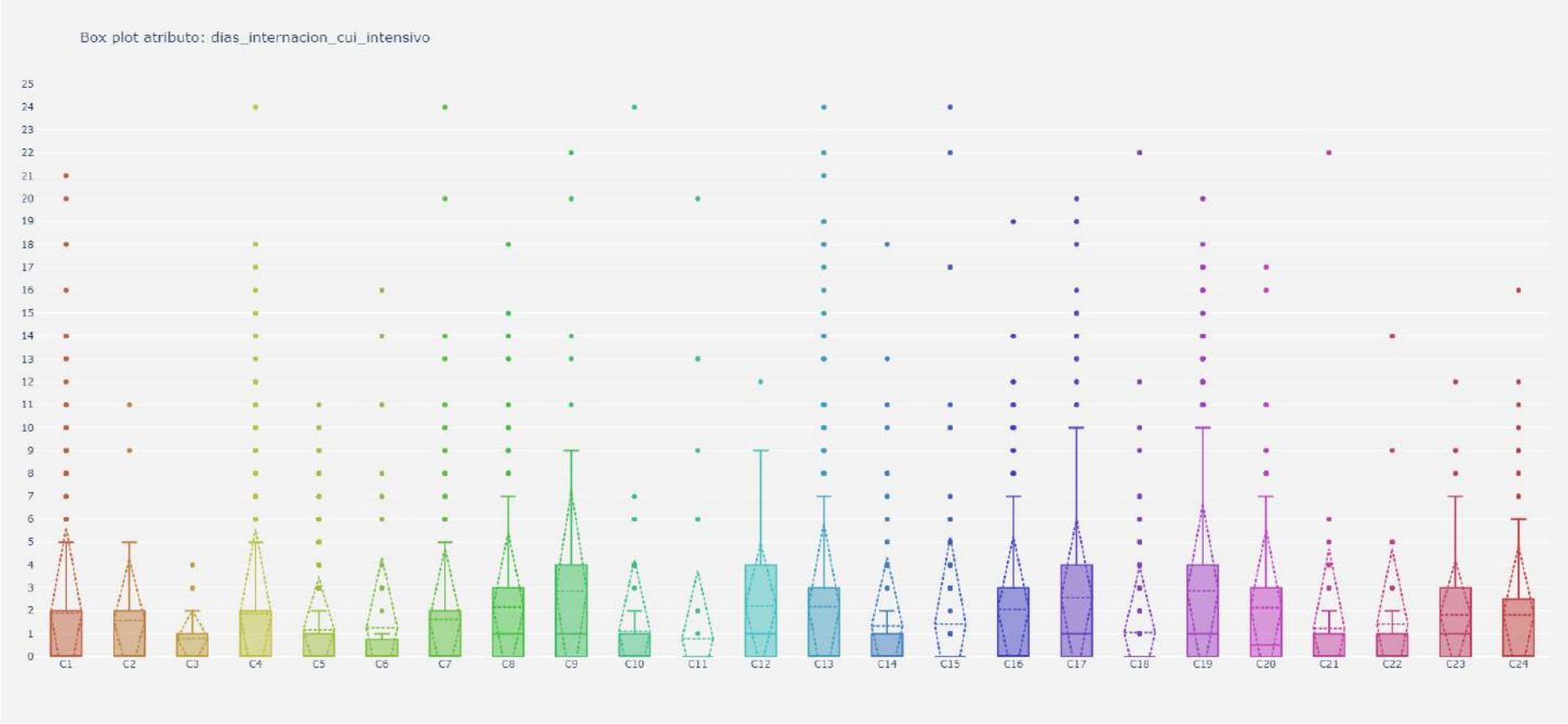
Dataset 4

Atributos numéricos

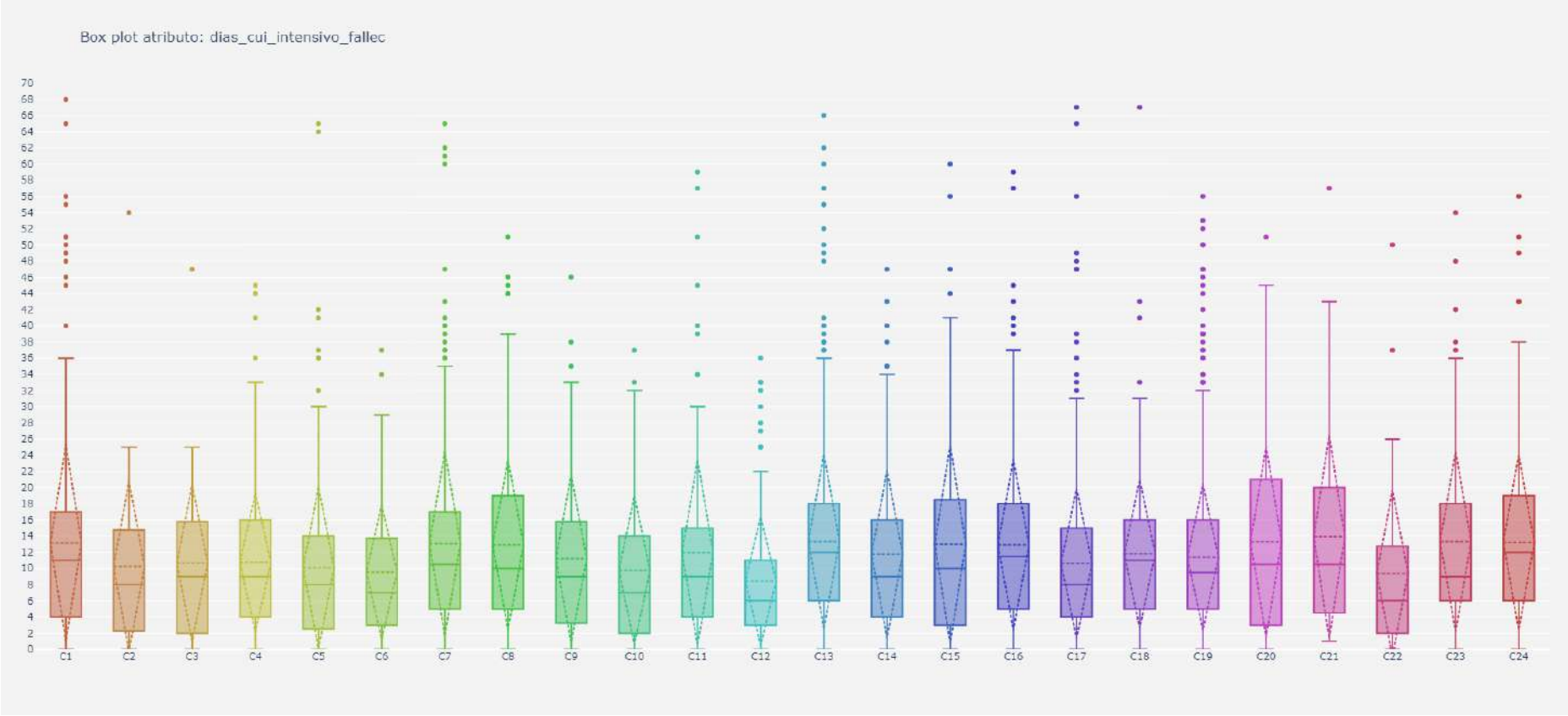
Edad



Dias_internacion_cui_intensivo

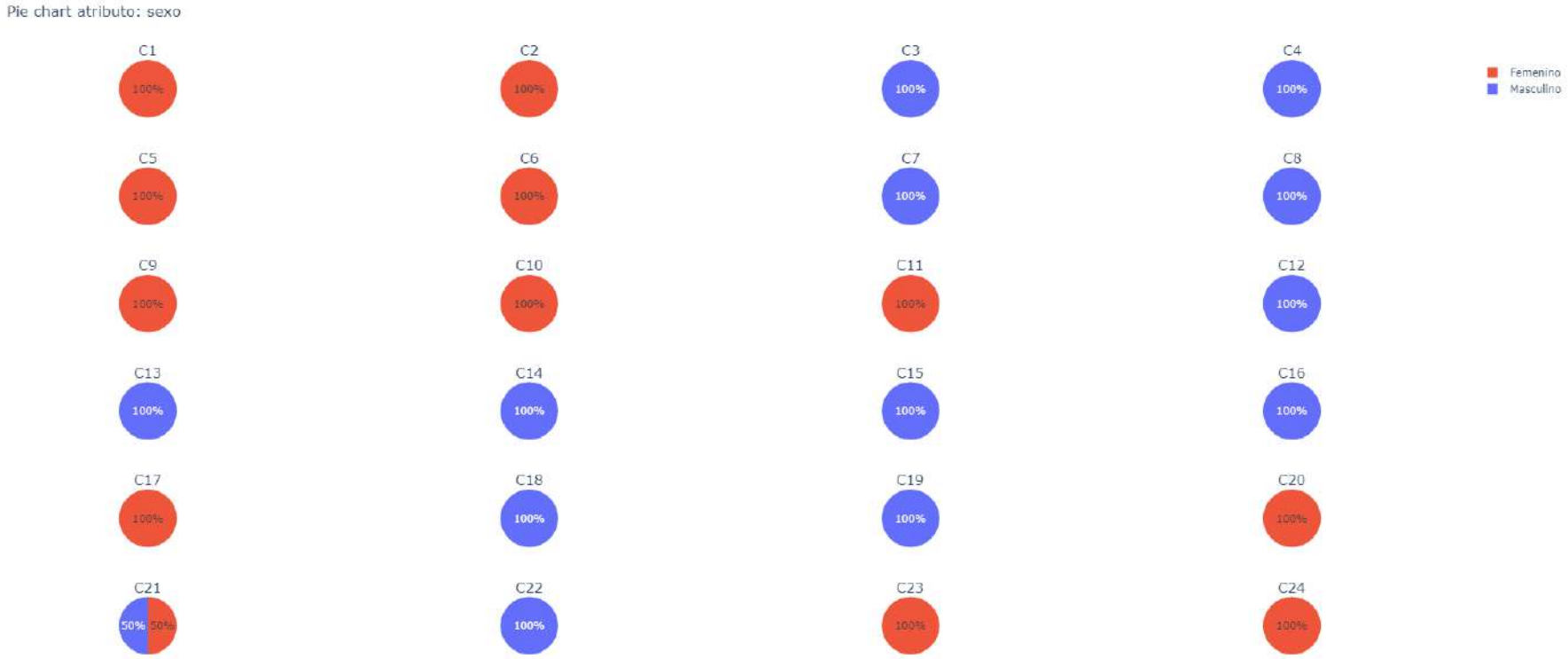


Dias_cui_intensivo_fallec



Atributos categóricos

Sexo

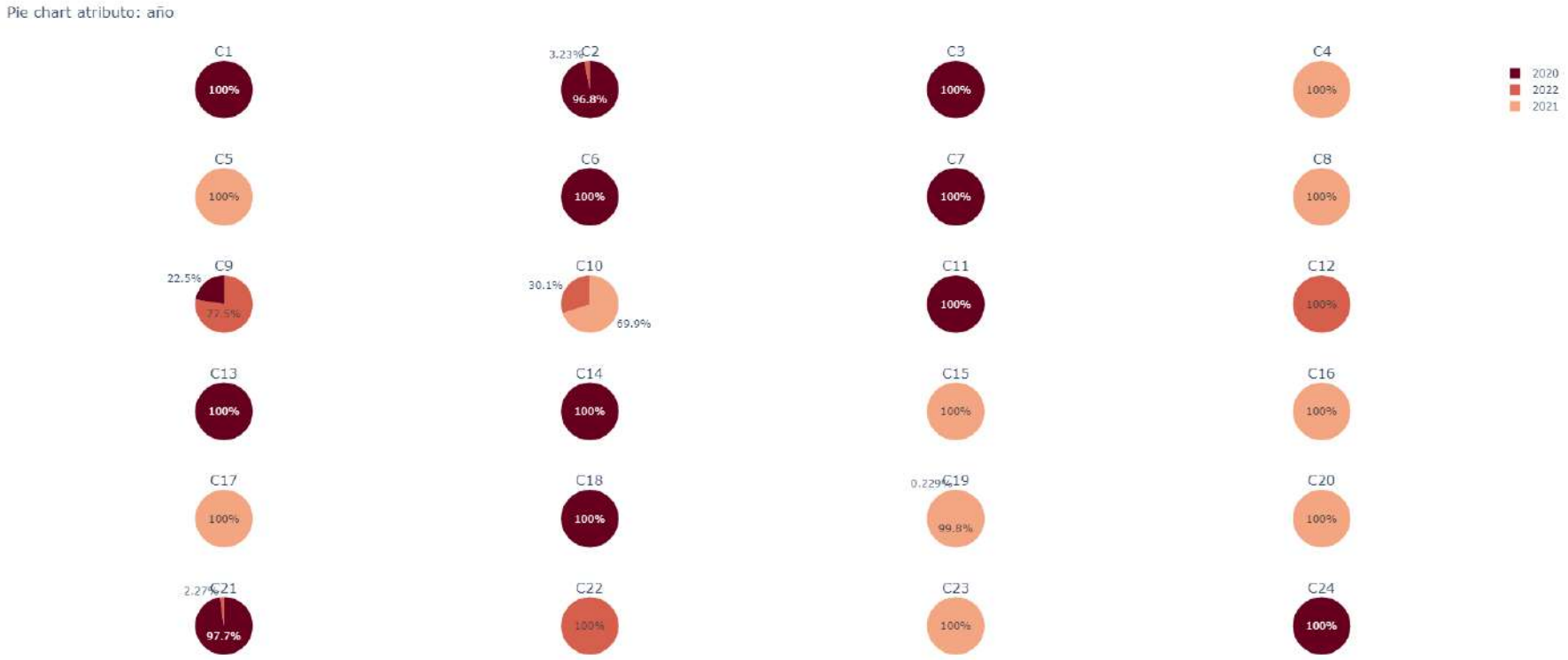


Asistencia_respiratoria_mecanica

Estacion


Pie chart atributo: estacion
C1
100%
C2
100%
C3
100%
C4
100%
Invierno
Otoño
Verano
Primavera
C5
20.7%
6.9%
72.4%
C6
9.89%
90.1%
C7
26.4%
73.6%
C8
8.67%
91.3%
C9
100%
C10
100%
C11
100%
C12
100%
C13
100%
C14
100%
C15
31.9%
8.33%
59.7%
C16
100%
C17
100%
C18
100%
C19
100%
C20
100%
C21
100%
C22
100%
C23
8.08%
91.9%
C24
100%

Año

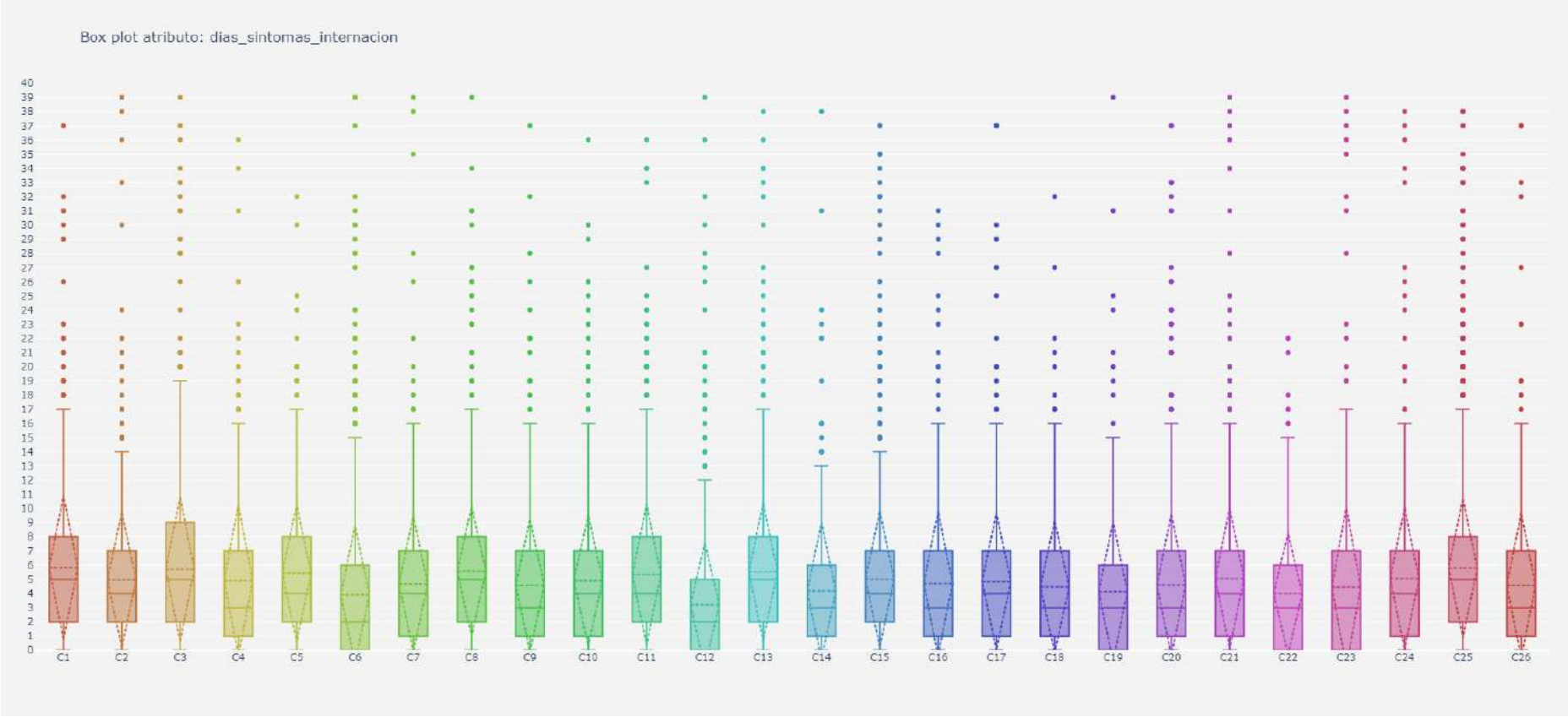

Pie chart atributo: año
C1
100%
C2
3.23%
96.8%
C3
100%
C4
100%
2020
2022
2021
C5
100%
C6
100%
C7
100%
C8
100%
C9
22.5%
77.5%
C10
30.1%
69.9%
C11
100%
C12
100%
C13
100%
C14
100%
C15
100%
C16
100%
C17
100%
C18
100%
C19
0.229%
99.8%
C20
100%
C21
2.27%
97.7%
C22
100%
C23
100%
C24
100%

Dataset 5

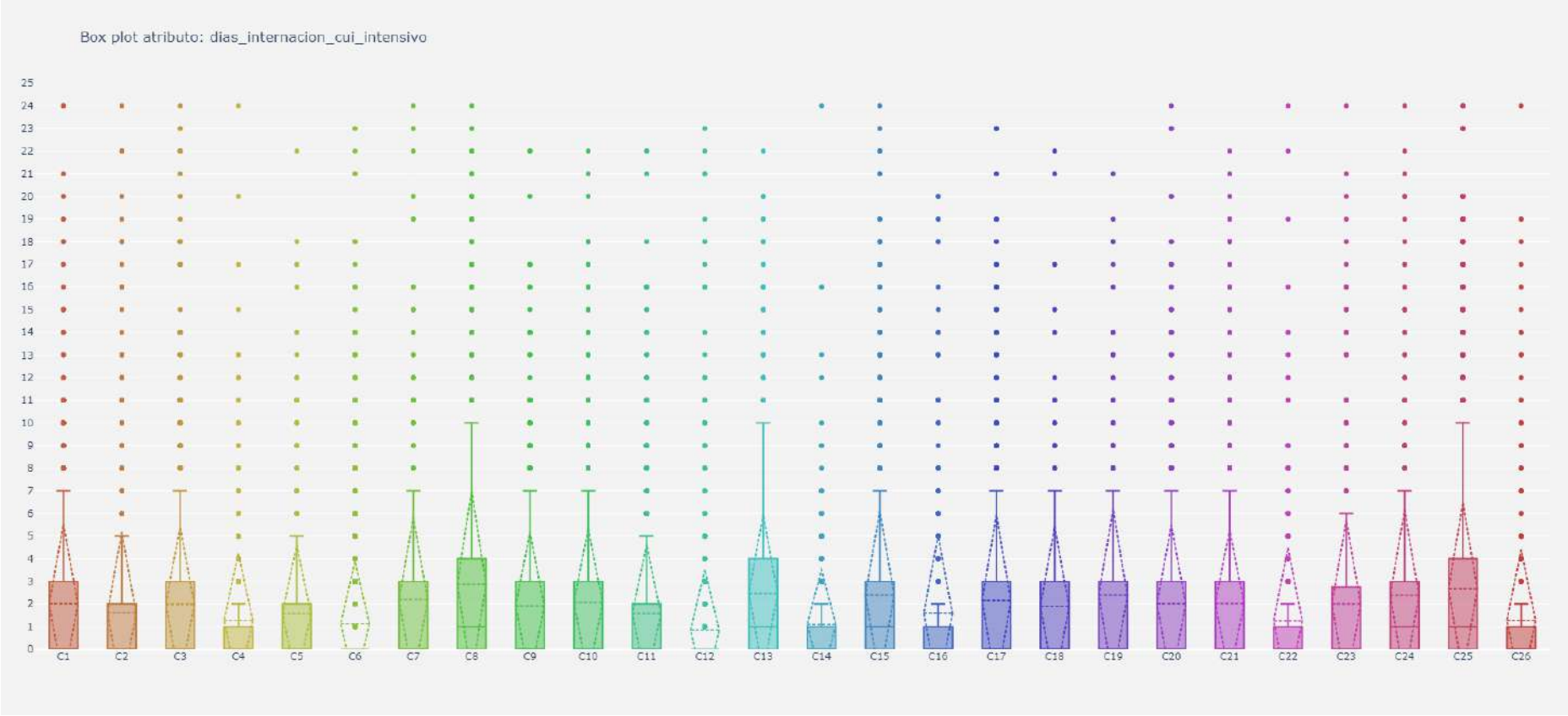
Atributos numéricos

Edad

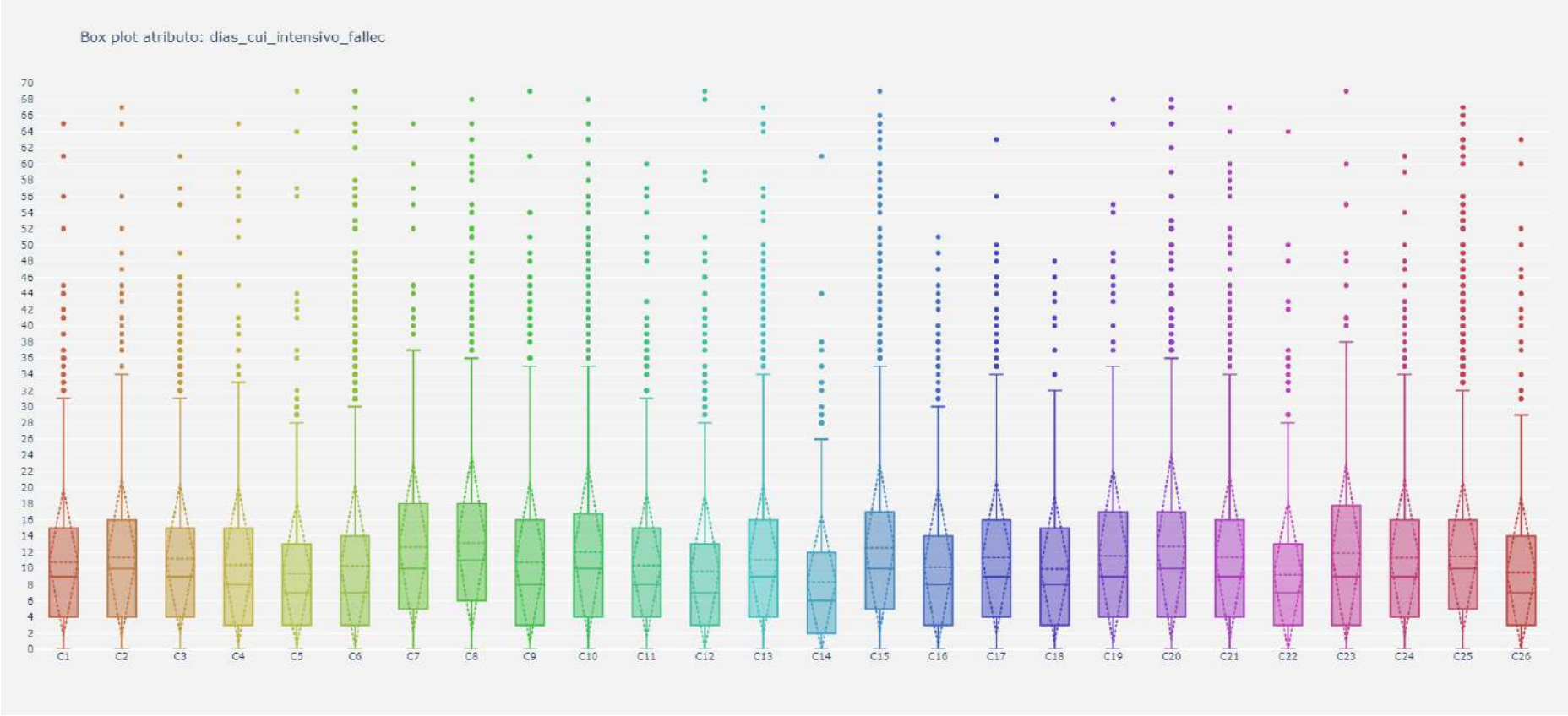
Dias_sintomas_internacion



Dias_internacion_cui_intensivo

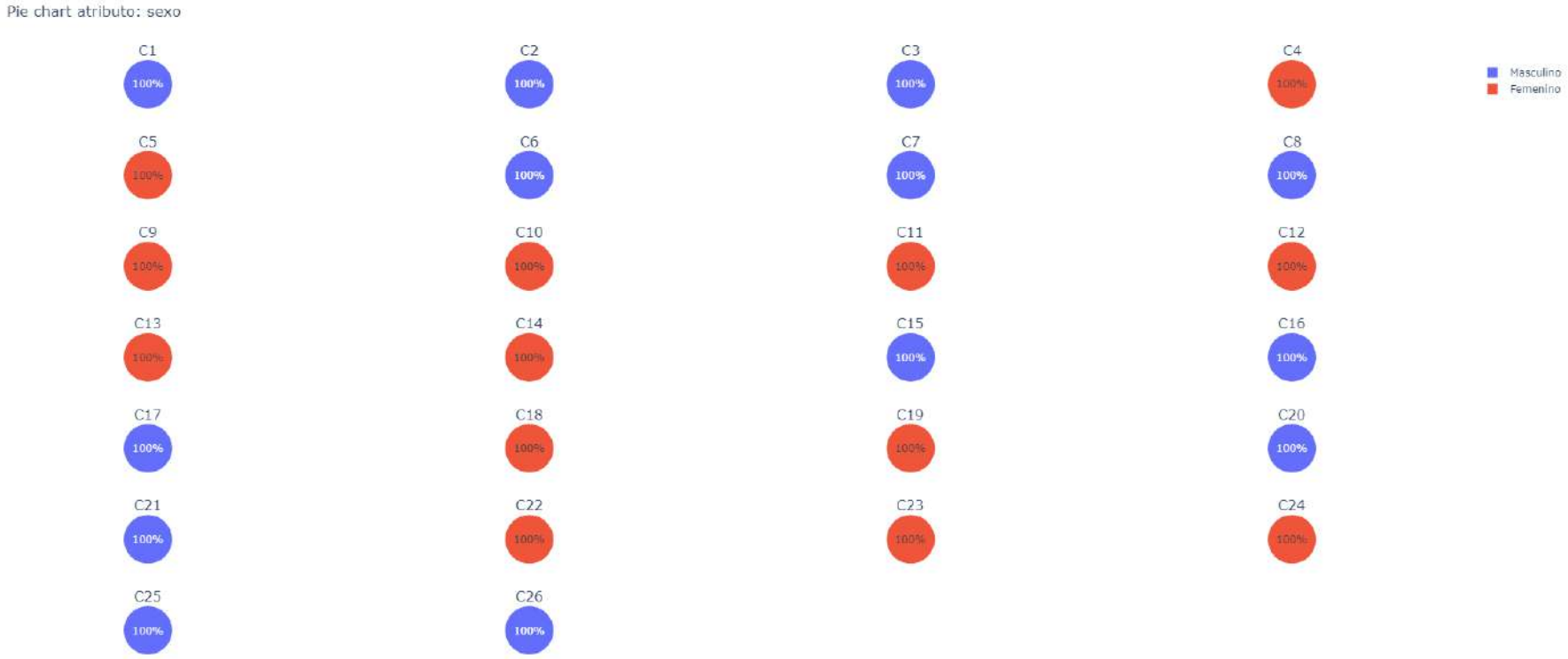


Dias_cui_intensivo_fallec

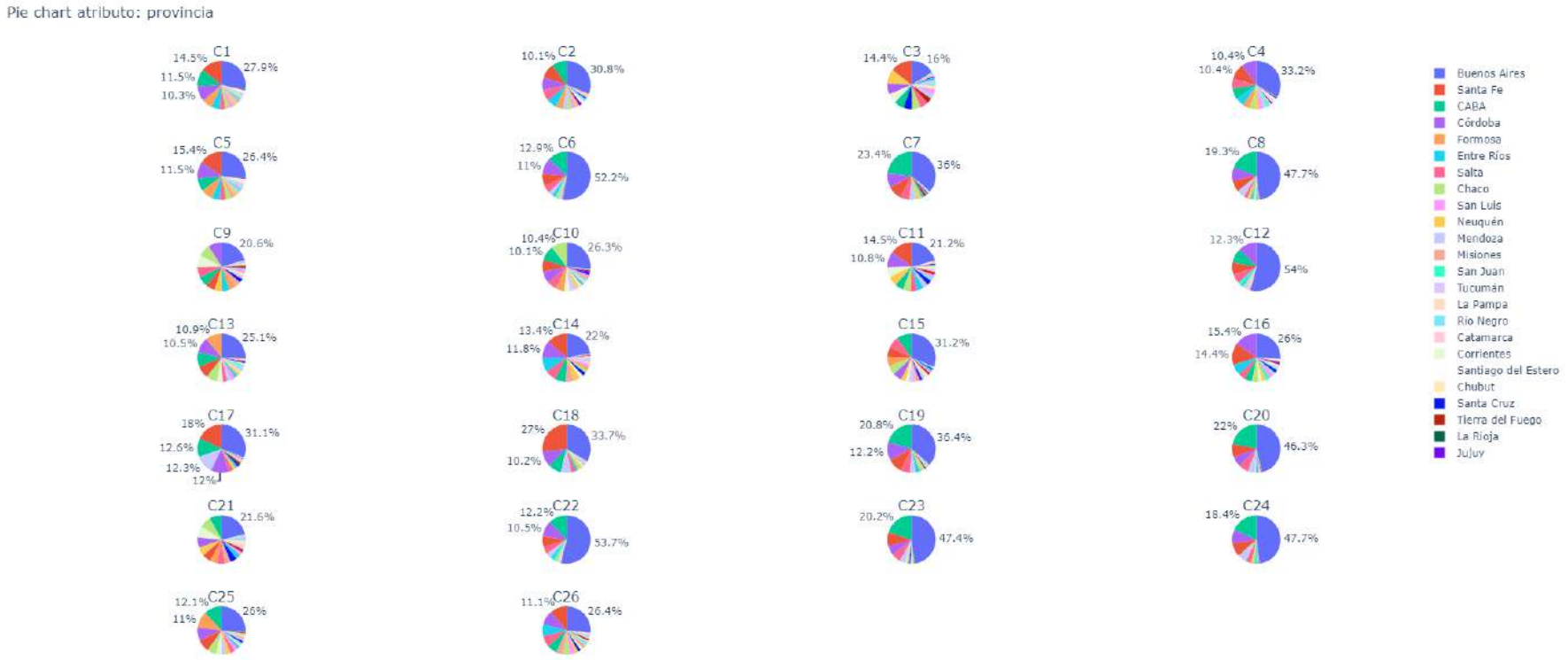


Atributos categóricos

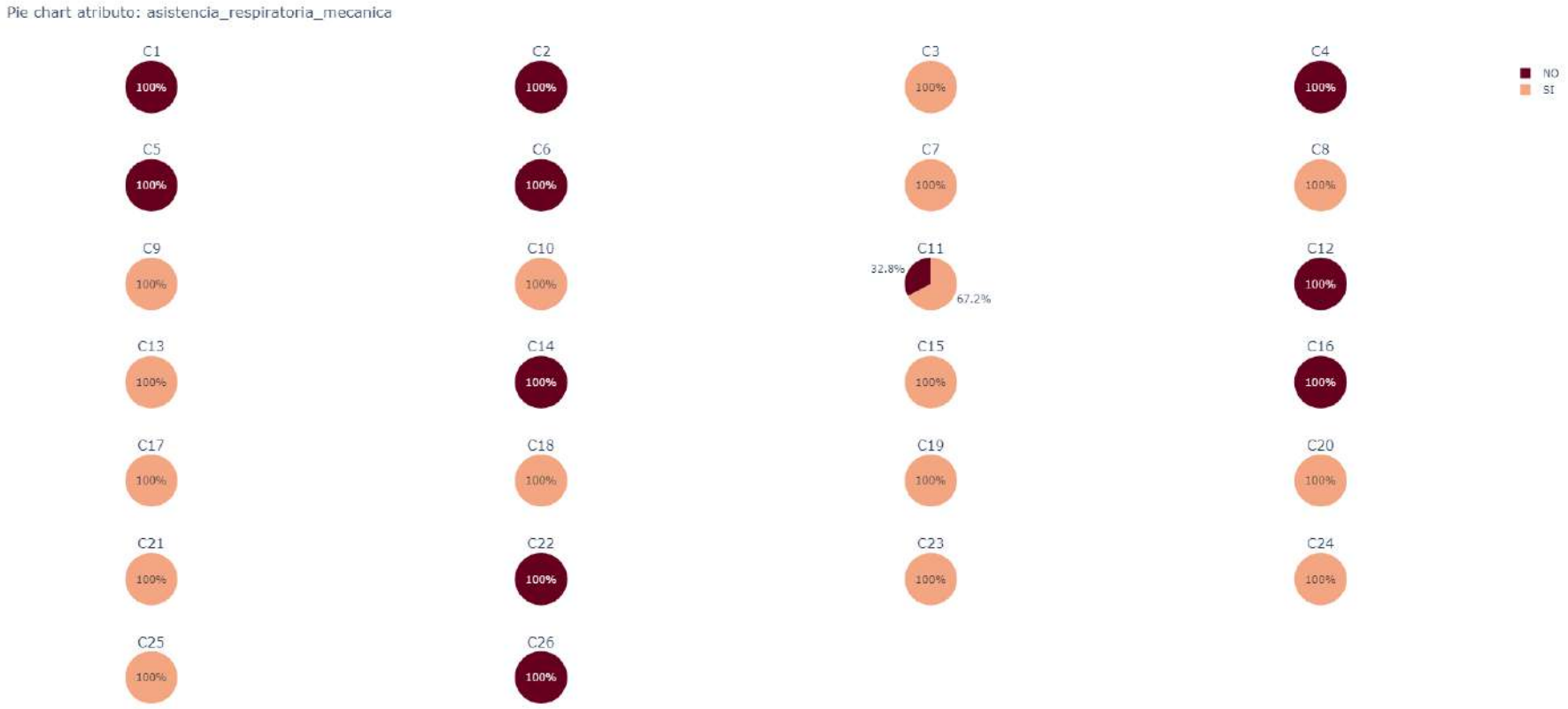
Sexo



Provincia

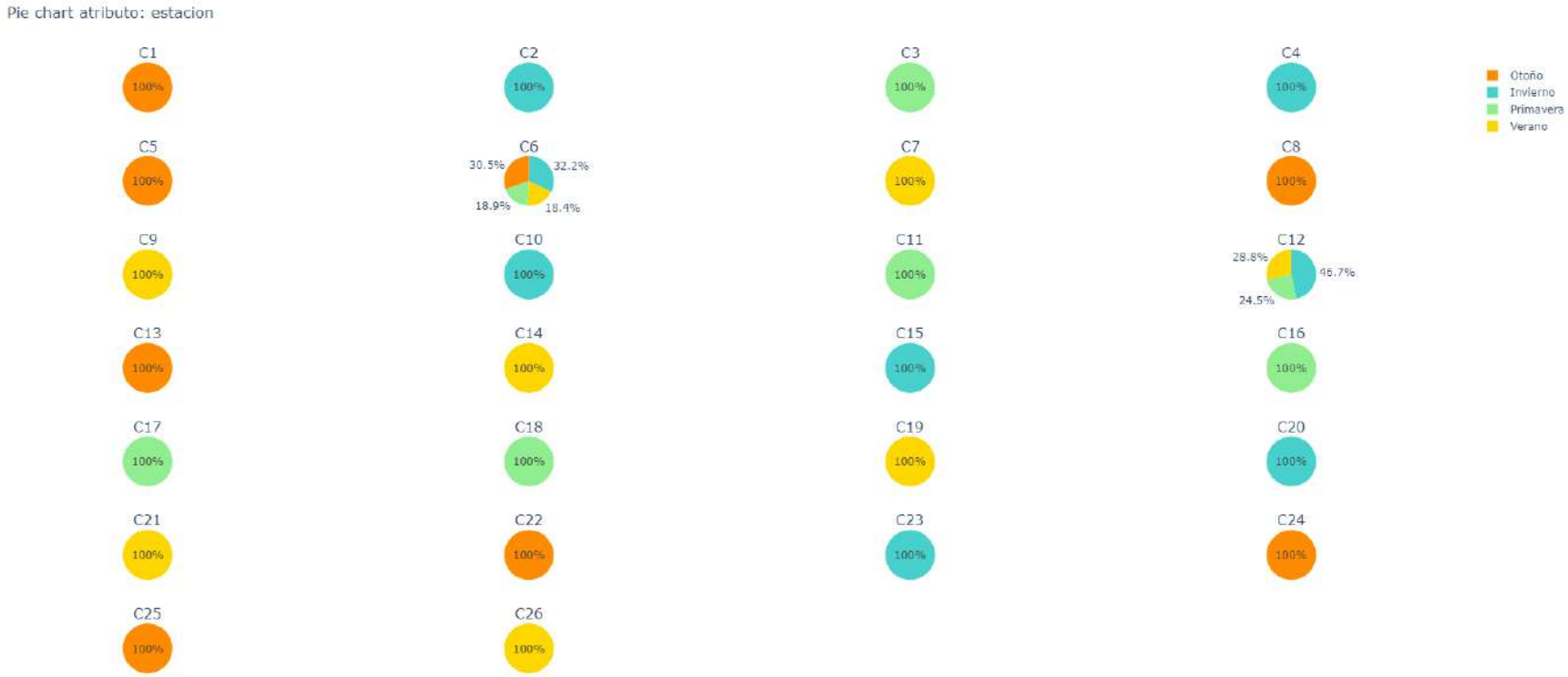

Pie chart atributo: provincia
C1
14.5%
27.9%
11.5%
10.3%
C2
10.1%
30.8%
C3
14.4%
16%
C4
10.4%
10.4%
33.2%
C5
15.4%
26.4%
11.5%
C6
12.9%
11%
52.2%
C7
23.4%
36%
C8
19.3%
47.7%
C9
20.6%
C10
10.4%
10.1%
26.3%
C11
14.5%
21.2%
10.8%
C12
12.3%
54%
C13
10.9%
10.5%
25.1%
C14
13.4%
11.8%
22%
C15
31.2%
C16
15.4%
14.4%
26%
C17
18%
31.1%
12.6%
12.3%
12%
C18
27%
33.7%
10.2%
C19
20.8%
36.4%
12.2%
C20
22%
46.3%
C21
21.6%
C22
12.2%
10.5%
53.7%
C23
20.2%
47.4%
C24
18.4%
47.7%
C25
12.1%
11%
26%
C26
11.1%
26.4%
Buenos Aires
Santa Fe
CABA
Córdoba
Formosa
Entre Ríos
Salta
Chaco
San Luis
Neuquén
Mendoza
Misiones
San Juan
Tucumán
La Pampa
Río Negro
Catamarca
Corrientes
Santiago del Estero
Chubut
Santa Cruz
Tierra del Fuego
La Rioja
Jujuy

Asistencia_respiratoria_mecanica

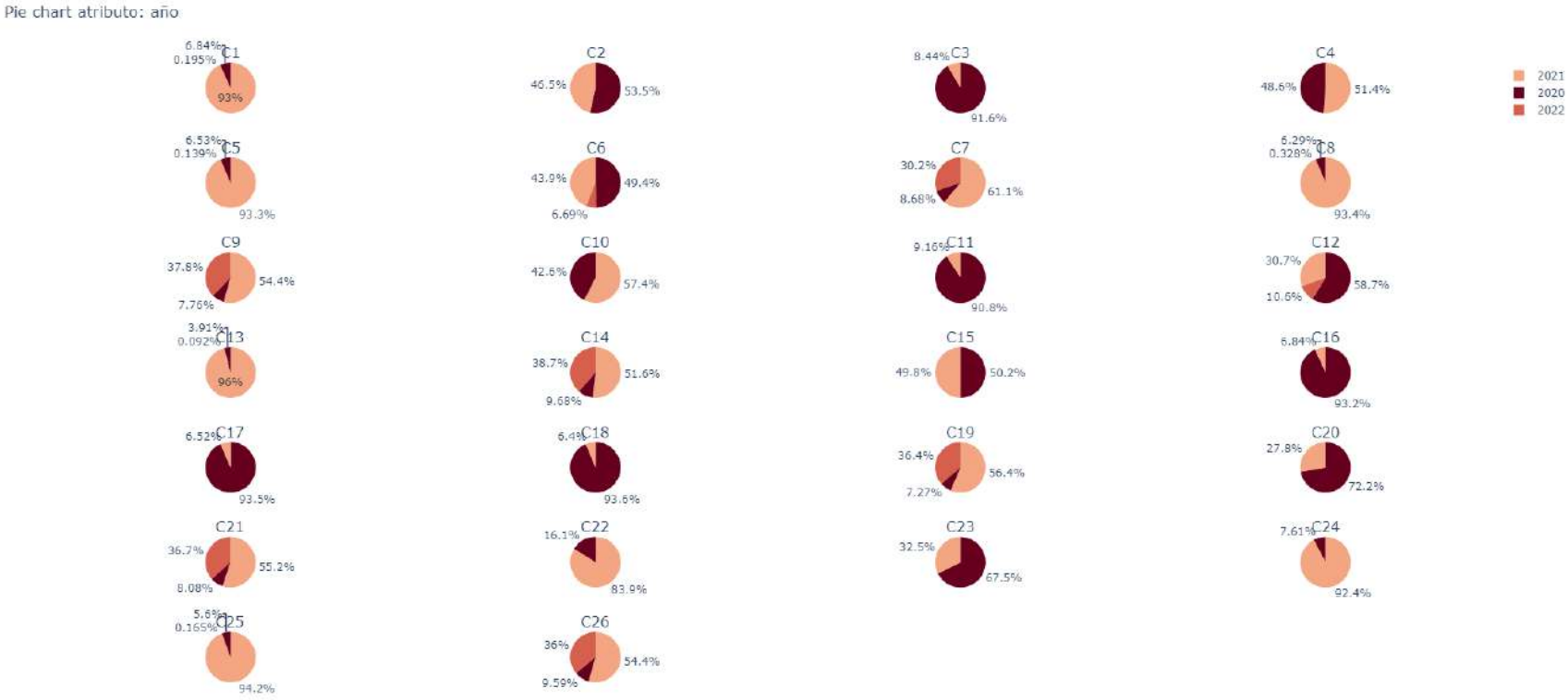


Origen_financiamiento

Estacion


Pie chart atributo: estacion
C1
100%
C2
100%
C3
100%
C4
100%
Otoño
Invierno
Primavera
Verano
C5
100%
C6
30.5%
32.2%
18.9%
18.4%
C7
100%
C8
100%
C9
100%
C10
100%
C11
100%
C12
28.8%
46.7%
24.5%
C13
100%
C14
100%
C15
100%
C16
100%
C17
100%
C18
100%
C19
100%
C20
100%
C21
100%
C22
100%
C23
100%
C24
100%
C25
100%
C26
100%

Año


Pie chart atributo: año
C1
6.84%
0.195%
93%
C2
46.5%
53.5%
C3
8.44%
91.6%
C4
48.6%
51.4%
C5
6.53%
0.139%
93.3%
C6
43.9%
49.4%
6.69%
C7
30.2%
8.68%
61.1%
C8
6.29%
0.328%
93.4%
C9
37.8%
54.4%
7.76%
C10
42.6%
57.4%
C11
9.16%
90.8%
C12
30.7%
10.6%
58.7%
C13
3.91%
0.092%
96%
C14
38.7%
51.6%
9.68%
C15
49.8%
50.2%
C16
6.84%
93.2%
C17
6.52%
93.5%
C18
6.4%
93.6%
C19
36.4%
56.4%
7.27%
C20
27.8%
72.2%
C21
36.7%
55.2%
8.08%
C22
16.1%
83.9%
C23
32.5%
67.5%
C24
7.61%
92.4%
C25
5.6%
0.165%
94.2%
C26
36%
54.4%
9.59%
2021
2020
2022

Anexo 2

Pruebas realizadas con SOM y métodos de representación explorados

Índice general

Índice de figuras

Índice de tablas

Primeras pruebas con SOM (JAVA SOMToolbox)

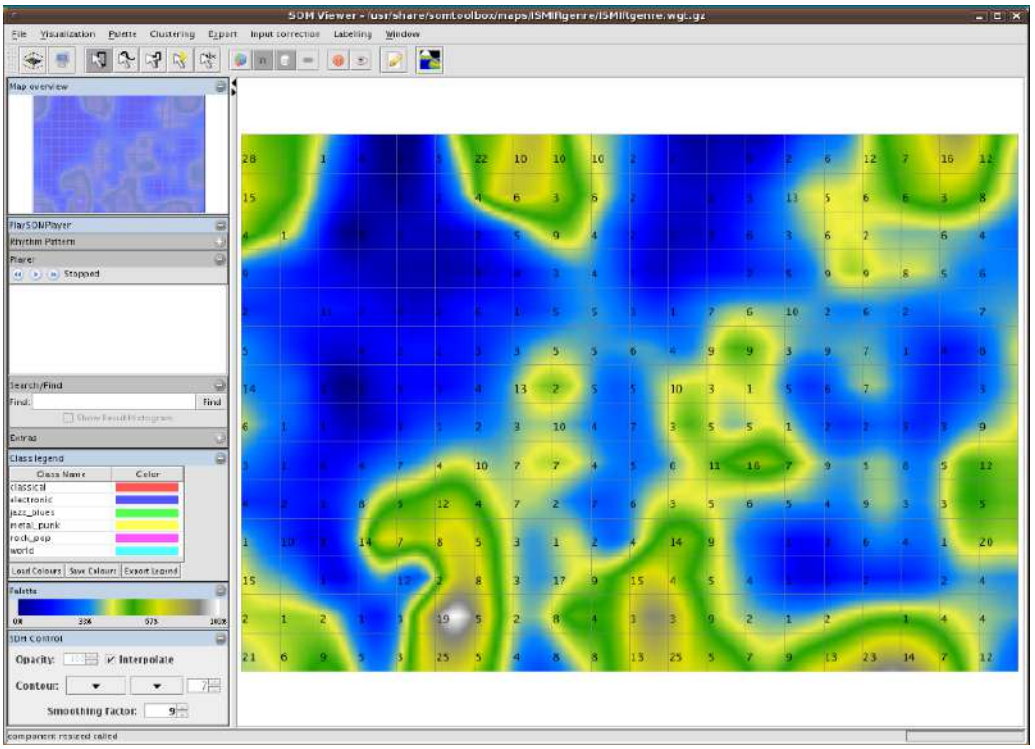
A continuación se detalla la experiencia, pruebas y resultados obtenidos con Java SOMToolbox para el entrenamiento de los mapas auto-organizados.

Introducción

Este software de código abierto fue desarrollado por el Instituto de Tecnología de Software y Sistemas Interactivos de la Universidad de Tecnología de Viena, Austria y tiene la licencia Apache, versión 2.0. Consiste en una implementación del modelo clásico de SOM realizada en Java. Esta alternativa fue la primera en ser elegida debido al gran atractivo que presentan las opciones de visualización del mapa.

Figura 1

Imagen del software Java SOMToolbox



Nota. Tomado de *Data Mining with the Java SOMToolbox*, por Universidad Tecnológica de Viena, 2011, http://www.ifs.tuwien.ac.at/dm/somtoolbox/

En la página web del software puede verse que cuenta con una interfaz dedicada a la inspección visual de los mapas auto-organizados entrenados (**Figura 1**). Esta interfaz en teoría permite cambiar entre varias vistas en tiempo real, pudiendo así analizar varios aspectos y realizar comparaciones de forma dinámica.

Desafortunadamente, en la práctica este programa presentó varios fallos y *bugs* que imposibilitaron el uso de todas las opciones disponibles. Aunque se destinó tiempo a entender el código fuente del software y se pudieron solucionar algunos problemas, la complejidad del código impidió que la herramienta se pudiera utilizar de manera satisfactoria. Además, el entrenamiento de los mapas mediante esta herramienta no contaba con la flexibilidad necesaria para este proyecto, ya que algunos parámetros de los mapas resultaban difíciles de cambiar mientras que otros no admitían modificaciones.

Pese a que el programa fue descartado tras el análisis para su utilización en el proyecto, sirvió como primera experiencia práctica con los mapas auto-organizados. Resultó útil para familiarizarse con los conceptos teóricos aprendidos, entender el funcionamiento de la técnica y visualizar la respuesta frente a los cambios realizados en sus parámetros.

Pruebas realizadas

A continuación se dará una breve explicación sobre las pruebas realizadas con Java SOMToolbox.

Como punto de partida para la elección de los valores de los parámetros utilizados en el entrenamiento del SOM se siguieron las heurísticas que se mencionan en *Neural Networks and Learning Machines* [45]. Dicho libro abarca varias técnicas referentes al *data mining*, incluyendo los mapas auto-organizados. Las heurísticas sugeridas por el autor y que utilizarían posteriormente como base son las siguientes:

- Función de decaimiento: $\sigma(t) = \sigma_0 e^{(\frac{-t}{\tau})}$, siendo el radio inicial $\sigma_0 = \frac{max\,(filas,columnas)}{2}$, τ la constante de vida media definida como $\tau = \frac{T}{log\,(\sigma_0)}$ y T la cantidad total de iteraciones, aproximada como $T = 500 * filas * columnas$.

- Función de aprendizaje: $\alpha(t) = \alpha_0 e^{(\frac{-t}{T})}$, siendo α_0 la tasa de aprendizaje inicial definida como $\alpha_0 = 0.1$.

- El tamaño de la grilla, es decir, la cantidad de neuronas que componen el mapa, se aproxima como $5\sqrt{n}$, siendo *n* la cantidad de casos presentes en el *dataset*.

Todos estos valores fueron utilizados con el fin de conocer la técnica de los mapas auto-organizados. Luego, para los *datasets* creados en posteriores etapas del proyecto (ver **Selección de *datasets* finales** en el **Informe**) se eligieron parámetros de acuerdo al resultado de distintas pruebas destinadas a mejorar la calidad del agrupamiento.

Para las primeras experiencias con SOM se utilizaron diferentes *datasets* de prueba que presentan una menor complejidad. Estos son conocidos en el ámbito del *data mining* y se utilizan a menudo para realizar *tests* y evaluar el desempeño de varios algoritmos [46, 47]. Concretamente, se realizaron pruebas con los siguientes *datasets*:

- Iris (150 casos, 4 atributos numéricos)
- Animals (16 casos, 13 atributos categóricos numerizados)
- Zoo (101 casos, 20 atributos categóricos numerizados)
- Chain link (1000 casos, 3 atributos numéricos)

- 10 clusters (850 casos, 10 atributos numéricos)

Los tres primeros corresponden a datos reales, mientras que los últimos dos contienen datos generados artificialmente para forzar topologías específicas. Todos los datos presentes en estos *datasets* se encuentran etiquetados. Esto permite comparar las clases reales con los *clusters* obtenidos tras el entrenamiento de un mapa y evaluar así su desempeño y la elección de los parámetros. Además, los mismos autores de Java SOMToolbox presentan sus propios mapas entrenados con los valores de los parámetros utilizados para cada uno de estos *datasets*. En algunos casos, mediante el refinamiento de parámetros, se han obtenido incluso mejores resultados que aquellos publicados en la página.

Por simplicidad, sólo se mostrarán los resultados significativos:

Dataset Iris

Este juego de datos es uno de los más conocidos en el dominio del aprendizaje automático. El *dataset* contiene 3 clases de plantas de iris, cada una con 50 instancias. Las plantas se describen por 4 atributos numéricos:

- Largo del sépalo
- Ancho del sépalo
- Largo del pétalo
- Ancho del pétalo

Una clase (**setosa**) es linealmente separable de las otras dos (**virginica** y **versicolor**), mientras que las últimas no son linealmente separables entre sí.

Se realizaron dos pruebas, la primera con los parámetros elegidos por los autores de Java SOMToolbox y la otra con los mencionados previamente del libro *Neural Networks and Learning Machines* (**Tabla 1**).

Tabla 1

Valores, según autor, de los distintos parámetros para las pruebas 1 y 2 (dataset Iris)

	Parámetros autores Java SOMToolbox (1)	Parámetros libro de Haykin S. (2)
Tamaño *x*	10	8
Tamaño *y*	10	8
Tasa de aprendizaje inicial (α_0)	0,7	0,1
Cantidad de iteraciones (T)	$\frac{10000}{8}$	33000
Radio de vecindad inicial (σ_0)	5	4
Constante de vida media (τ)	$\frac{10000}{5}$	$\frac{33000}{log(4)}$
Función de aprendizaje ($\alpha(t)$)	$0,7\ e^{\left(\frac{-t}{10000/8}\right)}$	$0,1\ e^{\left(\frac{-t}{33000}\right)}$

Para cada una de las pruebas, se utilizó la opción para visualizar la matriz-U resultante (**Figuras 2 y 3**):

Figura 2

Matriz-U resultante de la prueba 1

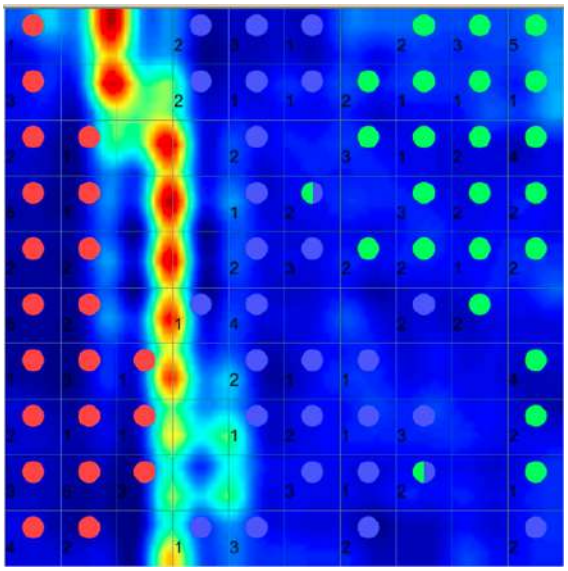
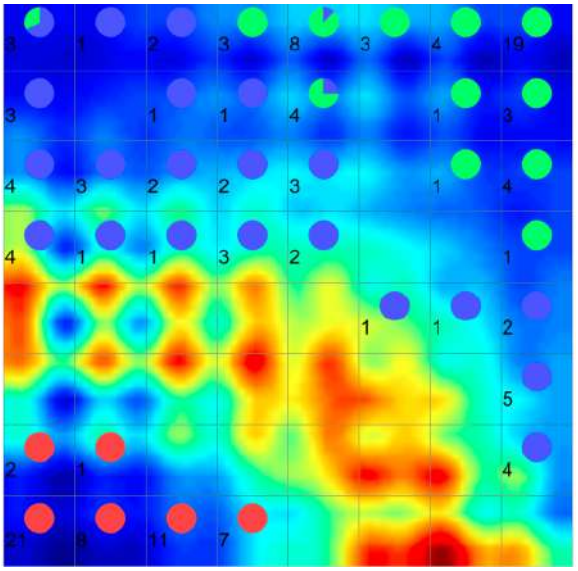


Figura 3

Matriz-U resultante de la prueba 2



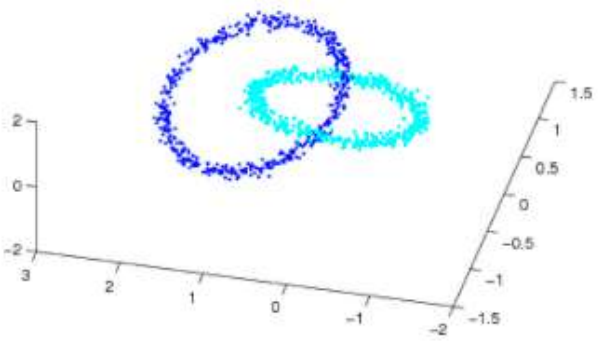
En cada celda se muestra la cantidad de *hits* (cantidad de veces que una neurona fue BMU) y la/s etiqueta/s perteneciente/s (indicada/s con un círculo de color) a los puntos de datos que hicieron match con dicha neurona. En cuanto a la escala de colores de la matriz-U los tonos en azul oscuro indican cercanía entre las neuronas (extremo inferior del rango de distancias), mientras que los tonos en rojo reflejan grandes distancias (extremo superior).

Si bien era esperable que entre las **virginicas** y las **versicolores** no se muestre una separación tan marcada como lo es con las **setosas**, en la **Figura 3** se puede apreciar que hay celdas de colores claros, en la parte superior derecha, que denotan una distinción entre las dos clases primeramente nombradas. En contraparte, en la **Figura 2** si no fuese por la etiqueta que indica la clase sería imposible encontrar una separación entre las **virginicas** y las **versicolores**. Por lo tanto, en este caso se observa que con los parámetros elegidos en la prueba 2 se obtiene un mejor resultado.

Dataset Chain Link

Chain Link es un ejemplo clásico de un *dataset* que provoca violaciones de preservación de topología. Contiene dos anillos, cada uno bidimensional, que se entrelazan en un espacio tridimensional (ver **Figura 4**). Al proyectar este conjunto de datos en un espacio de salida bidimensional, los anillos tienen que "romperse".

Figura 4
Visualización tridimensional de los puntos de datos de Chain Link



Nota. Tomado de *Data Mining with the Java SOMToolbox*, por Universidad Tecnológica de Viena, 2011, http://www.ifs.tuwien.ac.at/dm/somtoolbox/datasets.html

Al igual que en el caso anterior, se realizaron dos pruebas con diferentes parámetros (**Tabla 2**):

Tabla 2
Valores, según autor, de los distintos parámetros para las pruebas 1 y 2 (dataset Chain Link)

	Parámetros autores Java SOMToolbox (1)	Parámetros libro de Haykin S. (2)
Tamaño x	12	13
Tamaño y	18	13
Tasa de aprendizaje inicial (α_0)	0,7	0,1
Cantidad de iteraciones (T)	$\frac{10000}{8}$	84500
Radio de vecindad inicial (σ_0)	5	6,5

Constante de vida media (τ)	$\frac{10000}{5}$	$\frac{84500}{log(6{,}5)}$
Función de aprendizaje ($\alpha(t)$)	$0,7\ e^{\left(\frac{-t}{10000/8}\right)}$	$0,1\ e^{\left(\frac{-t}{84500}\right)}$

Figura 5

Matriz-U resultante de la prueba 1

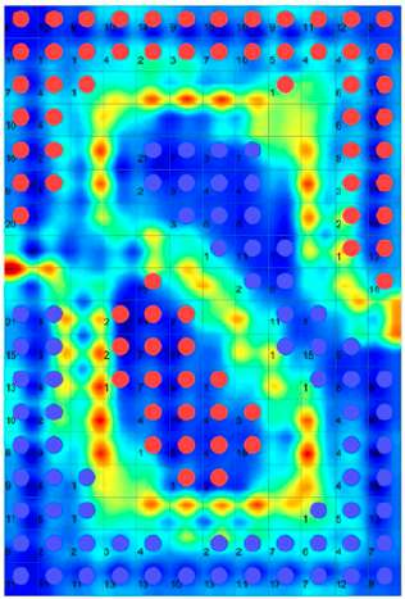
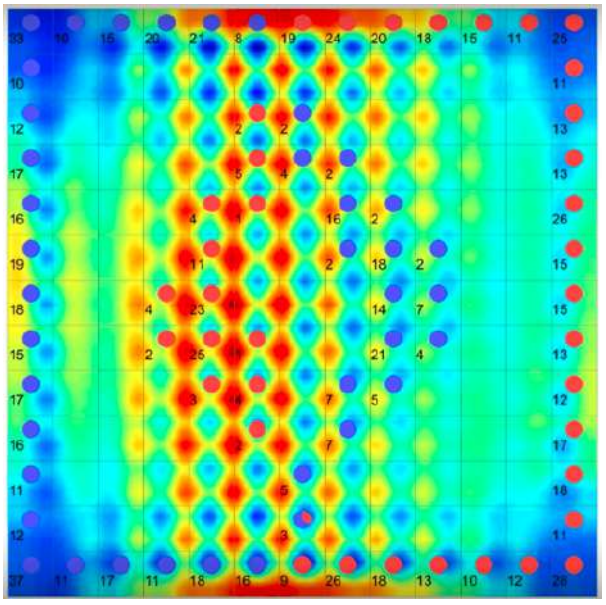


Figura 6

Matriz-U resultante de la prueba 2



En contraparte a lo que ocurrió en la prueba 2 del *dataset* Iris, se puede ver en la **Figura 6** que los parámetros utilizados no arrojaron un mejor resultado que los de la prueba 1 (**Figura 5**). Con esto se llega a la conclusión que los parámetros que se utilizan dependen en gran parte de la naturaleza de los datos y que no hay una "receta" a seguir para asegurar un resultado óptimo.

Dataset Zoo (101 animals)

Este conjunto de datos cuenta con 101 animales que se describen mediante 20 atributos con valores *booleanos* y se pueden etiquetar en siete clases diferentes. Resultados:

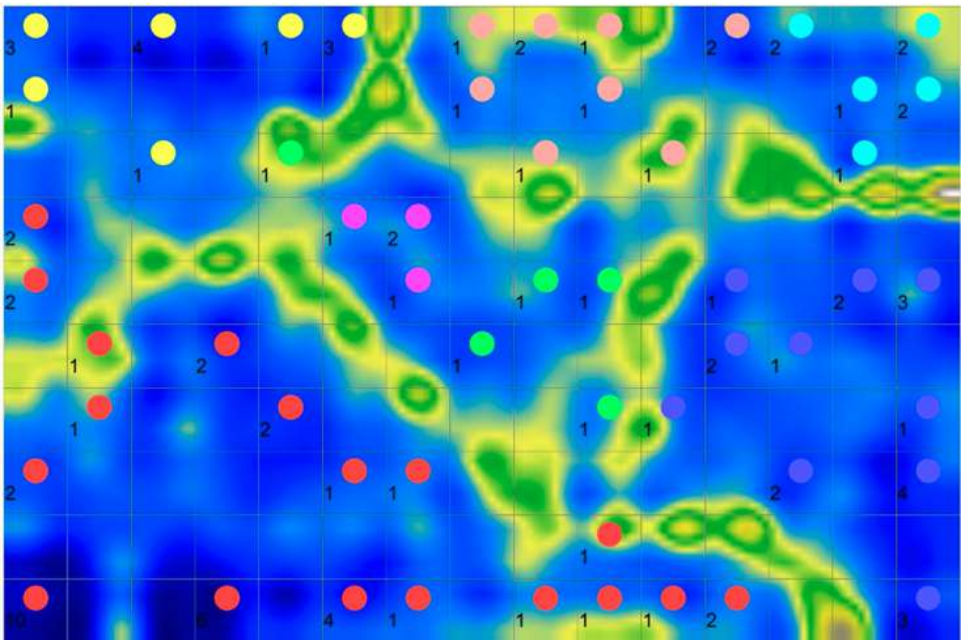
Tabla 3

Parámetros autores Java SOMToolbox (Dataset Zoo)

Tamaño *x*	15
Tamaño *y*	10
Tasa de aprendizaje inicial (α_0)	0,7
Cantidad de iteraciones (T)	50000
Radio de vecindad inicial (σ_0)	5
Constante de vida media (τ)	$\frac{50000}{5}$
Función de aprendizaje ($\alpha(t)$)	$0,7\ e^{\left(\frac{-t}{50000/8}\right)}$

Figura 7

Matriz-U resultante de procesar el dataset Zoo en Java SOMToolbox utilizando los parámetros de la Tabla 3



Se realizó una prueba con el *dataset* Zoo (parámetros usados en **Tabla 3**) con el objetivo de observar el comportamiento del SOM estándar con datos descritos por atributos de tipo categórico. Esto se debe a que este tipo de dato está presente en los *datasets* elegidos para las posteriores etapas del **Informe**.

Al analizar la **Figura 7** se puede notar la formación de clases bien delimitadas donde en su mayoría intentan agrupar una única categoría. Si bien, en algunos casos, existe el solapamiento de dos categorías diferentes en un mismo grupo y la separación de una misma categoría en dos grupos, este agrupamiento indicaría que se pueden obtener resultados aceptables.

Finalmente, se realizó una primera evaluación con los datos de COVID-19 publicados por el Ministerio de Salud [48] en la versión correspondiente en esta instancia del trabajo (**Tabla 8** del **Informe**). Se realizaron dos pruebas con distintos parámetros (**Tabla 4**):

Tabla 4

Valores de los distintos parámetros para las pruebas 1 y 2 (dataset público COVID-19)

	Parámetros prueba 1	Parámetros prueba 2
Tamaño x	28	20
Tamaño y	30	15
Tasa de aprendizaje inicial (α_0)	0,1	0,7
Cantidad de iteraciones (T)	421000	250000
Radio de vecindad inicial (σ_0)	15	10
Constante de vida media (τ)	$\frac{421000}{log(15)}$	$\frac{250000}{log(10)}$
Función de aprendizaje ($\alpha(t)$)	$0,1\ e^{(\frac{-t}{421000})}$	$0,7\ e^{(\frac{-t}{250000})}$

Figura 8

Matriz-U resultante de la prueba 1

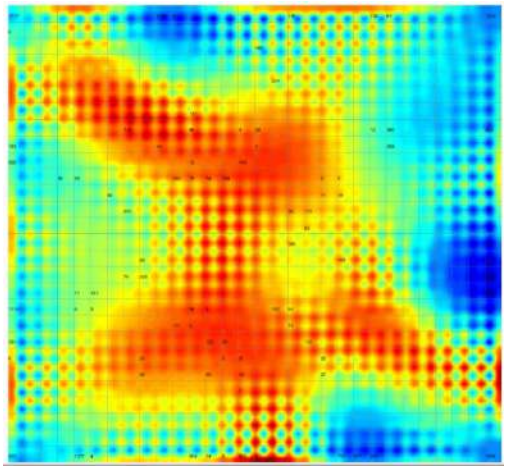
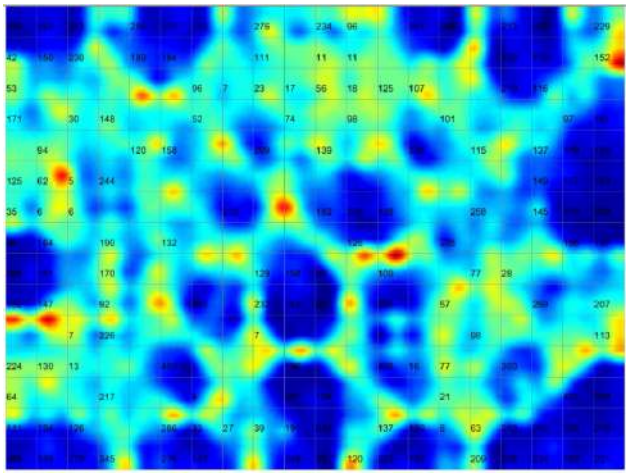


Figura 9

Matriz-U resultante de la prueba 2



En las **Figuras 8** y **9** se puede observar la presencia de algunos *clusters*, pero los resultados no parecen prometedores. Si bien en la matriz-U se ven conglomerados, los datos parecerían no "caer" en estos estando distribuídos en casi la totalidad de las neuronas. En la **Sección 4.2.10.** del **Informe** se aborda que la elección de los atributos, parámetros y además el uso de una topología toroidal juegan un papel clave para la obtención de un *clustering* de calidad en el proyecto final.

Librería minisom de Python

Introducción

Dado que Python es un lenguaje frecuentemente utilizado en el ámbito de *data mining*, cuenta con una cantidad considerable de paquetes y librerías desarrolladas para este fin. Para elegir la librería de SOM adecuada se investigó en diversos artículos. Uno en particular realiza una comparación de estas librerías teniendo en cuenta factores tales como la documentación disponible, la calidad del código, si es de simple instalación, entre otros [49] (ver **Figura 10**). En base a este artículo y luego de investigar cada librería, se decidió utilizar **minisom** para continuar con las pruebas de SOM sobre el *dataset*. La documentación de esta

librería resultó lo suficientemente clara como para entender su código y poder hacer las modificaciones pertinentes.

Figura 10

Análisis y comparaciones entre distintos paquetes de SOM

Package Reference	SuSi	SOMPY [36]	SimpSOM [37]	MiniSom [38]	TensorFlow SOM [39]	PyMVPA [40]	NeuPy [41]	kohonen [35]	SS-SOM [33]
Simple syntax	✓		✓	✓		✓	✓		
Useful documentation	✓			✓		✓	✓		✓
Well documented code	✓		✓	✓	✓		✓		
Unsupervised clustering	✓	✓	✓	✓	✓	✓	✓	✓	✓
Supervised regression	✓							✓	
Supervised classification	✓							✓	✓
Semi-supervised regression	✓								
Semi-supervised classification	✓								✓
Simple installation	✓		✓	✓		✓	✓	✓	
Python version	3	2	3	3	3	2	3	R	C

Nota. Tomado de *Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data,* (p.4), 2019.

El código de este paquete, a diferencia de Java SOMToolbox, permite realizar en forma simple una automatización para hacer varias ejecuciones sucesivas de SOM variando sus parámetros (cantidad de iteraciones, radio de vecindad, tamaño de la grilla, entre otros). Además, por la simplicidad del código es fácil cambiar otros factores relevantes en el algoritmo en sí, tal como el tipo de función de aprendizaje. Posibilita, por ejemplo, dividir el SOM en dos fases y entrenar un mapa ya pre-entrenado. Estas variantes serán abordadas con más detalle a continuación.

Pruebas realizadas

Una de las primeras modificaciones fue en la inicialización de los pesos de las neuronas. Estos dejaron de ser aleatorios y en su lugar pasaron a tomar valores cercanos al promedio de cada atributo. Con esto se buscó acelerar el proceso de ordenamiento del mapa.

Lo siguiente que se experimentó utilizando esta librería fue separar el entrenamiento del SOM en dos fases distintas: una de ordenamiento y otra de refinamiento o convergencia.

En la primera fase, se busca que las neuronas se ordenen topológicamente y el mapa tome su forma inicial. Para esto, se utiliza un σ_0 aproximadamente igual al radio del mapa y un α_0 relativamente mayor al de la segunda fase. Este proceso no suele durar más que unas pocas miles de iteraciones.

En la segunda fase, se refina el valor de cada neurona de modo que la precisión del mapa aumente y se represente fielmente el espacio de entrada. Para esto, se utiliza tanto un σ_0 como un α_0 pequeños. Este proceso dura muchas más iteraciones que la fase anterior (al menos 500 veces el número de neuronas).

Las características del *dataset* utilizado en esta instancia del trabajo se encuentran detalladas en la **Tabla 9** del **Informe**.

A continuación, en las **Figuras 11** y **12**, se muestran los resultados luego de entrenar un SOM con los siguientes parámetros (ver **Tabla 5**):

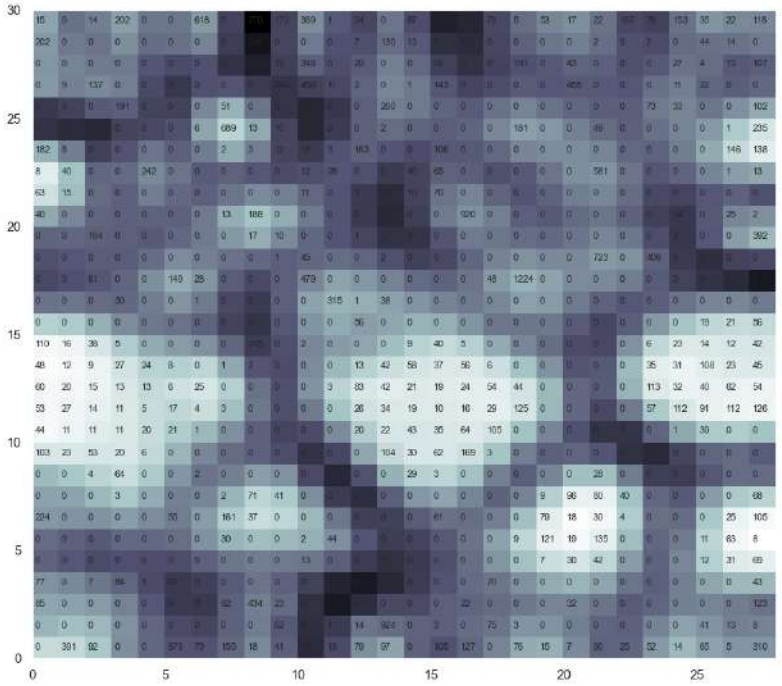
Tabla 5

Valores de los distintos parámetros para las fases de ordenamiento y convergencia (función de aprendizaje exponencial)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	28	28
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,7	0,05
Cantidad de iteraciones (T)	1000	420000
Radio de vecindad inicial (σ_0)	15	1
Función de aprendizaje ($\alpha(t)$)	$0,7.e^{\frac{-t}{1000}}$	$0,05.e^{\frac{-t}{420000}}$

Figura 11

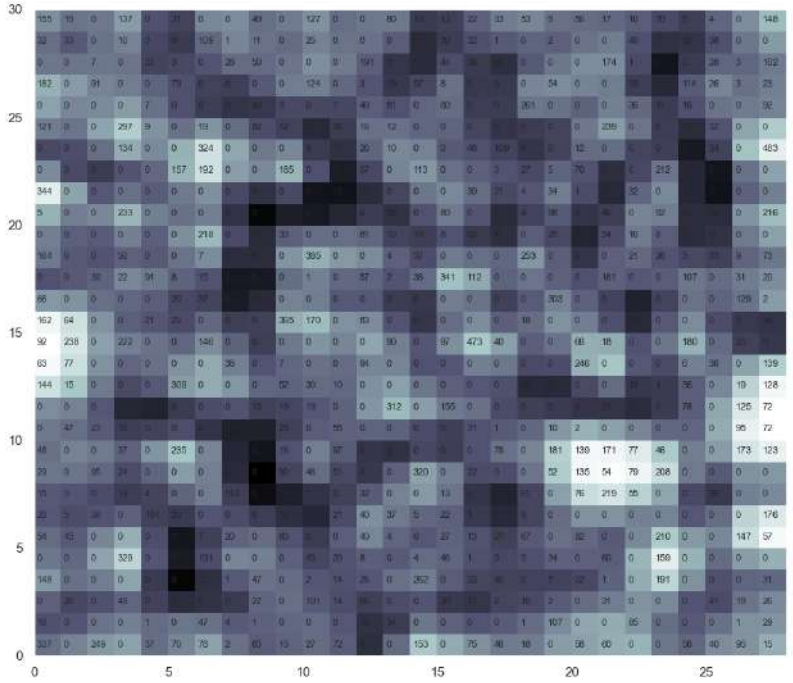
Matriz-U del SOM luego de la fase de ordenamiento



Nota. El color blanco indica cercanía entre neuronas mientras que el negro indica grandes distancias. Los números señalan la cantidad de veces que una neurona resultó BMU.

Figura 12

Matriz-U del SOM luego de la fase de convergencia



Nota. El color blanco indica cercanía entre neuronas mientras que el negro indica grandes distancias. Los números indican la cantidad de veces que una neurona resultó BMU.

Al terminar la primera fase, los valores de QE y TE del mapa eran 1,091 y 0,044, respectivamente. Si bien a primera vista pareciera que hay algunos grupos formados, existen una gran cantidad de neuronas que no resultaron BMU de ningún dato (neuronas muertas) y hay otras aisladas que tienen una alta concentración de datos (por ejemplo: 1224). Por lo tanto es evidente que el mapa aún requiere más ordenamiento. Al terminar la fase de convergencia el valor de QE era de 0,499 mientras que el de TE era 0,010.

Otra alternativa que se utilizó fue cambiar el tipo de función de aprendizaje, ya que hasta ahora solo se había utilizado una función de tipo exponencial. A continuación, en las **Figuras 13** y **14,** se presentan los resultados luego de entrenar un SOM con los mismos parámetros, pero utilizando una función de aprendizaje lineal (ver **Tabla 6**):

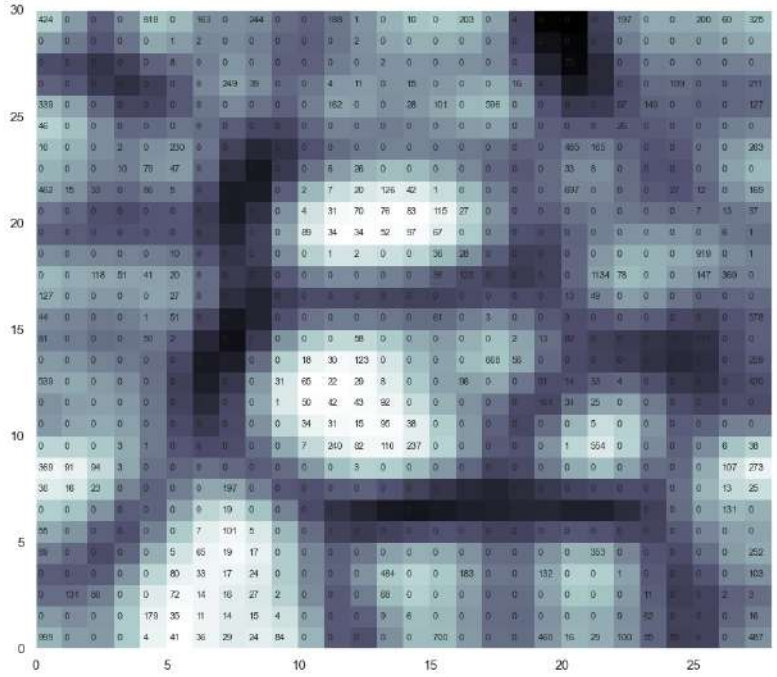
Tabla 6

Valores de los distintos parámetros para las fases de ordenamiento y convergencia (función de aprendizaje lineal)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	28	28
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,7	0,05
Cantidad de iteraciones (T)	1000	420000
Radio de vecindad inicial (σ_0)	15	1
Función de aprendizaje ($\alpha(t)$)	$0,7.(1-\frac{t}{1000})$	$0,05.(1-\frac{t}{420000})$

Figura 13

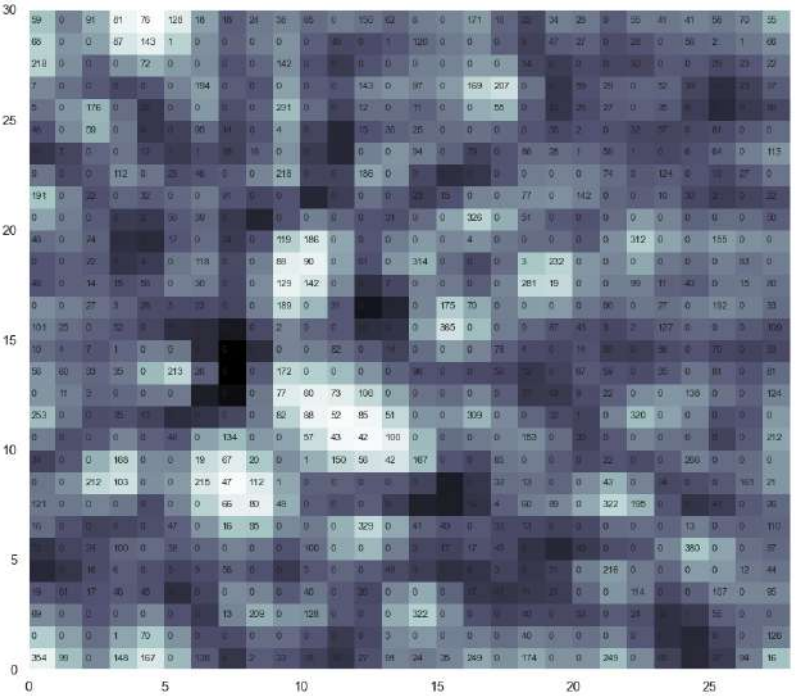
Matriz-U del SOM luego de la fase de ordenamiento



Nota. El color blanco indica cercanía entre neuronas mientras que el negro indica grandes distancias. Los números indican la cantidad de veces que una neurona resultó BMU.

Figura 14

Matriz-U del SOM luego de la fase de convergencia



Nota. El color blanco indica cercanía entre neuronas mientras que el negro indica grandes distancias. Los números indican la cantidad de veces que una neurona resultó BMU.

Al terminar la primera fase, los valores de QE y TE del mapa eran 1,160 y 0,019, respectivamente. Al terminar la fase de convergencia, el valor de QE era de 0,4889, mientras que el de TE era 0,008.

Como puede apreciarse, los dos resultados obtenidos anteriormente (funciones exponencial y lineal) no son muy distintos. Además de que los valores de las medidas de calidad son parecidos, ambas matrices-U son muy similares: en ninguna puede distinguirse formaciones claras de *clusters* y puede verse el efecto borde (predominancia de los datos de hacer *match* con neuronas situadas en los límites del mapa) .

Si se utilizan los mismos parámetros con la función de aprendizaje inversamente proporcional a las iteraciones (ver **Tabla 7**), se obtiene el siguiente resultado (ver **Figuras 15** y **16**):

Tabla 7

Valores de los distintos parámetros para las fases de ordenamiento y convergencia (función de aprendizaje inversa)

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	28	28
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,7	0,05
Cantidad de iteraciones (T)	1000	420000
Radio de vecindad inicial (σ_0)	15	1
Función de aprendizaje ($\alpha(t)$)	$0,7.(\frac{1}{t})$	$0,05.(\frac{1}{t})$

Figura 15

Matriz-U del SOM luego de la fase de ordenamiento

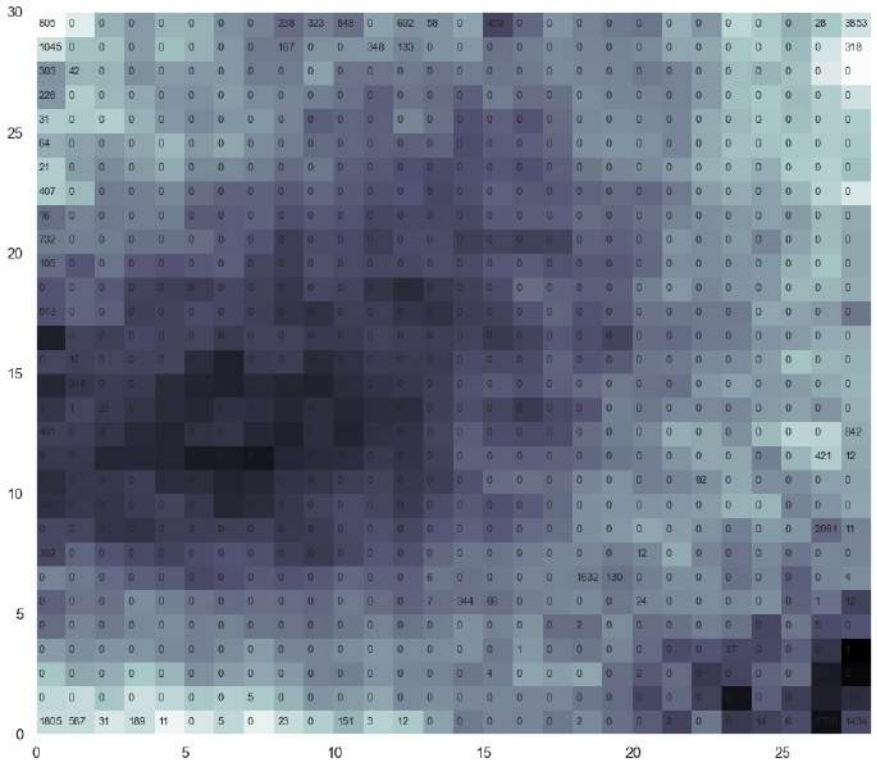
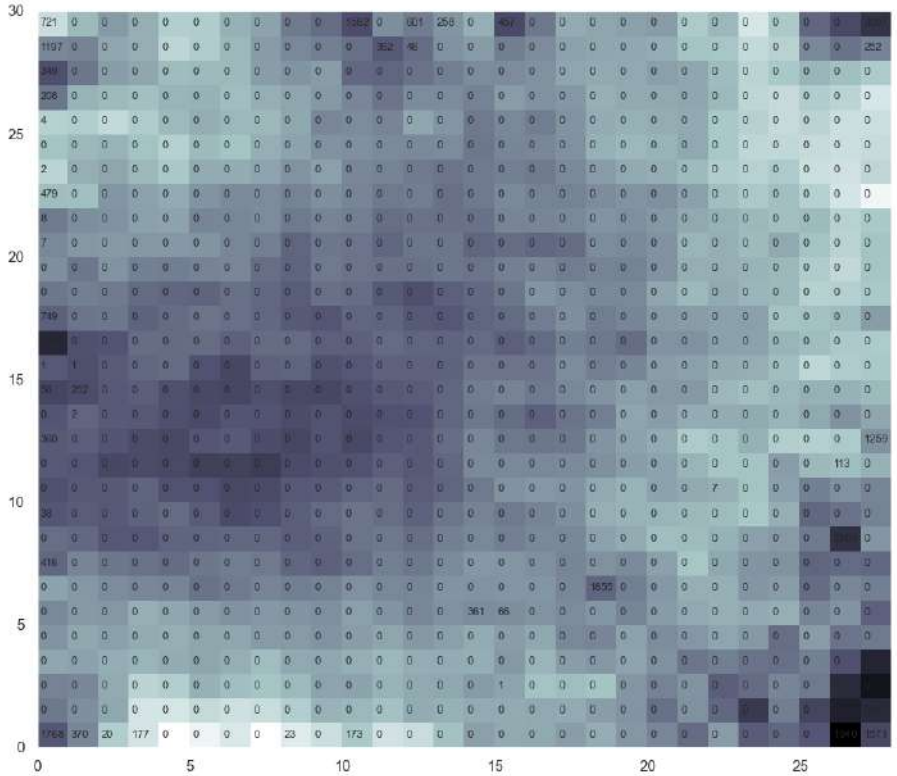


Figura 16

Matriz-U del SOM luego de la fase de convergencia



Al terminar la primera fase, los valores de QE y TE del mapa eran 1,675 y 0,136, respectivamente. Al terminar la fase de convergencia, el valor de QE era de 1,653, mientras que el de TE era 0,088. En este caso no solo los valores de ambas medidas de calidad fueron más altos, sino que también que el mapa sufrió en mayor medida las consecuencias del efecto borde.

Por último, se muestra un ejemplo de entrenamiento de mapa (ver **Figuras 17** y **18**) utilizando las funciones de aprendizaje y distancia usadas en otro proyecto [16] (ver **Tabla 8**):

Tabla 8

Valores de los distintos parámetros para las fases de ordenamiento y convergencia (funciones de aprendizaje y distancia exponenciales [16])

	Parámetros fase de ordenamiento (1)	Parámetros fase de convergencia (2)
Tamaño *x*	28	28
Tamaño *y*	30	30
Tasa de aprendizaje inicial (α_0)	0,7	0,05
Tasa de aprendizaje final(α_f)	0,05	0,001
Cantidad de iteraciones (T)	1000	420000
Radio de vecindad inicial (σ_0)	15	1
Radio de vecindad final(σ_f)	1	1

Figura 17

Matriz-U del SOM luego de la fase de ordenamiento

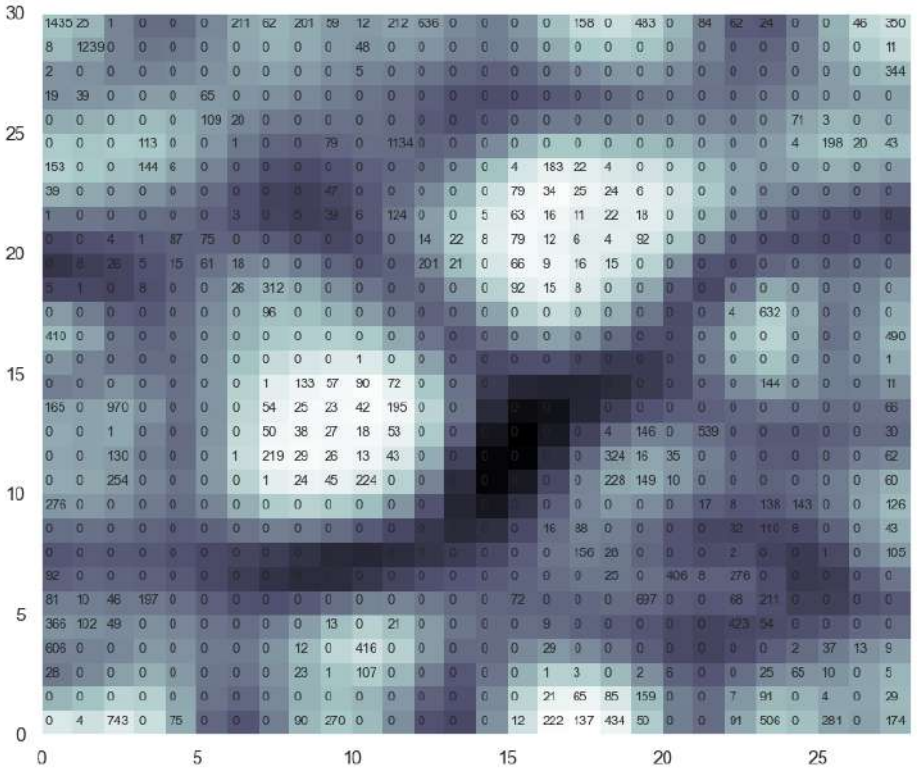
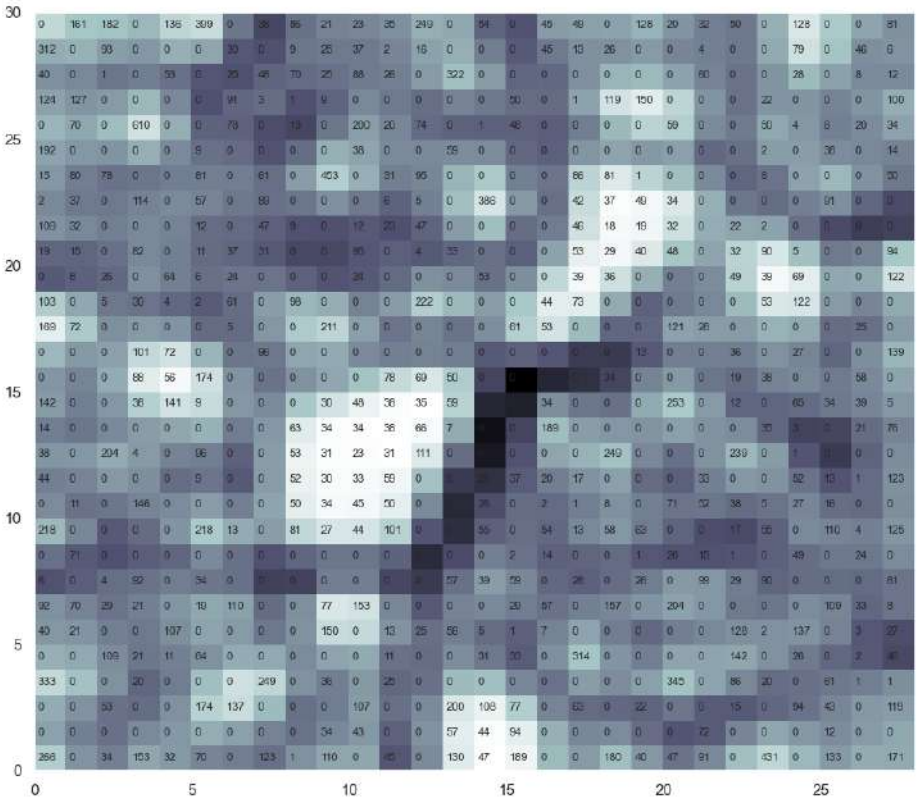


Figura 18

Matriz-U del SOM luego de la fase de convergencia



Al terminar la primera fase, los valores de QE y TE del mapa eran 1,204 y 0,007, respectivamente. Al terminar la fase de convergencia, el valor de QE era de 0,510, mientras que el de TE era 0,010.

Esta clase de resultados se mantuvo frente a numerosas variaciones de los parámetros. Se realizaron varias pruebas alternando las distintas funciones de aprendizaje y cambiando el tamaño del mapa y valores de α_0, σ_0 y *T*. También se experimentó entrenar mapas que ya habían sido entrenados previamente con el objetivo de facilitar el ordenamiento de los mismos (ver **Figuras 19** y **20**). En ninguno de estos casos se obtuvo un resultado significativamente distinto.

Figura 19

Matriz-U resultante del mapa pre-entrenado

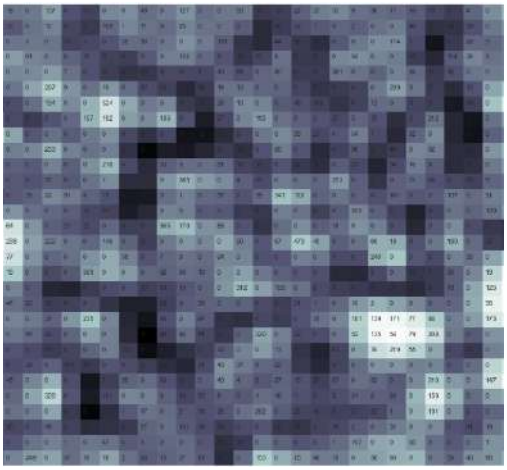
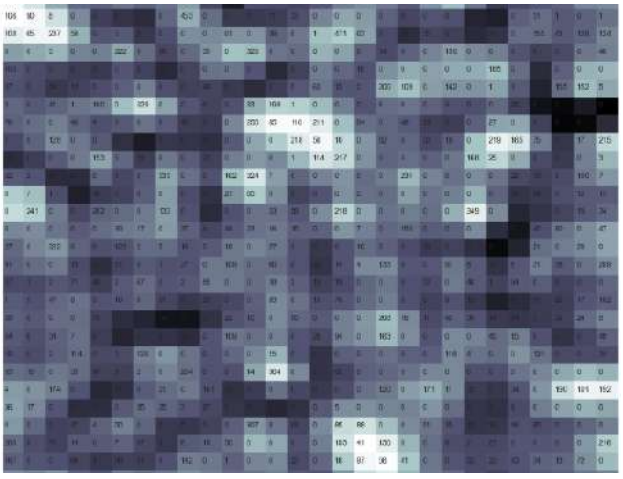


Figura 20

Matriz-U resultante del mapa re-entrenado



Métodos de interpretación de clusters

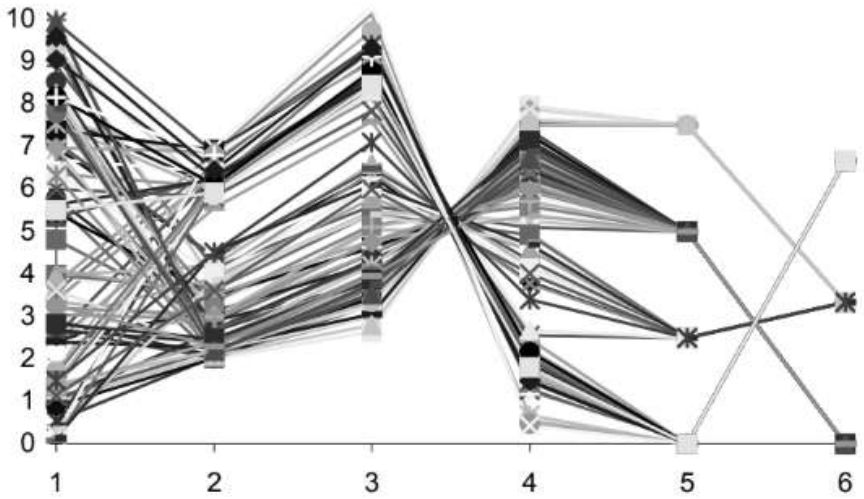
A continuación se explicará la experiencia en la utilización de métodos para la detección de patrones y para entender el significado de los *clusters*.

Primero se hicieron pruebas con los gráficos de coordenadas paralelas [2]. Estos son los más conocidos dentro de las técnicas de visualización de datos multidimensionales. Se mapea el espacio *k*-dimensional en dos dimensiones mediante el uso de *k* ejes de ordenadas

(escalados linealmente) por uno de abscisas. Cada punto en el espacio *k*-dimensional se hace corresponder con una línea poligonal (polígono abierto), donde cada vértice de la línea poligonal intersecta los *k* ejes en el valor para la dimensión. La **Figura 21** muestra un espacio 6-dimensional representado de esta forma.

Figura 21

Gráfica de seis coordenadas paralelas con muchos ejemplos

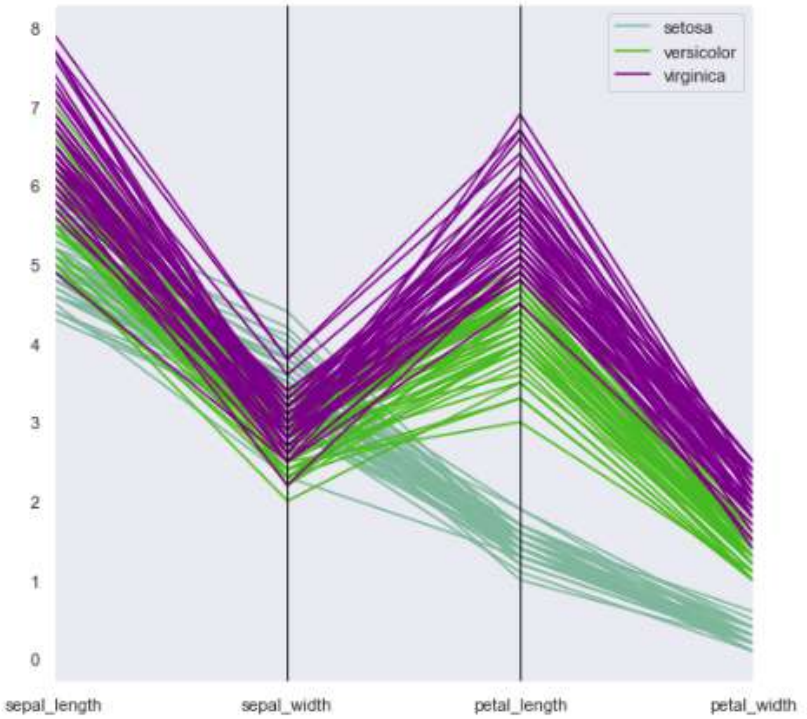


Nota. Tomado de *Introducción a la Minería de Datos* (p. 105), por J. Hernandéz Orallo, M. J. Ramírez Quintana y C. Ferris Ramírez, 2004, Pearson Prentice Hall.

La idea es que cada punto de dato en el gráfico tenga un color asociado correspondiente al grupo al que pertenece. Al observar cómo se comparan los valores de las variables entre los grupos, se puede tener una idea de lo que representan realmente. A continuación se ilustra un ejemplo de la utilización de estos gráficos para el conjunto de datos Iris (ver **Figura 22**).

Figura 22

Gráfico de coordenadas paralelas de los 3 grupos que componen el dataset Iris



Nota. Tomado de *La función parallel_coordinates*, por InteractiveChaos, https://interactivechaos.com/en/node/996

Se probó esta técnica en el *dataset* 2 (por ser el que tiene menos *clusters*) y desafortunadamente no fue de utilidad, como se ve en la **Figura 23**, en la identificación de patrones. Esto es fundamentalmente por la cantidad de *clusters* y además por la cantidad de casos, de atributos y por la binarización. También, como se muestra en la **Figura 24**, se intentó utilizar únicamente los centroides de cada grupo para la implementación de estos gráficos así como también sólo sus atributos numéricos (ver **Figura 25**), pero aún así los resultados no fueron satisfactorios.

Figura 23

Gráfico de coordenadas paralelas para todos los puntos de datos del dataset 2 (18 clusters) con todos sus atributos

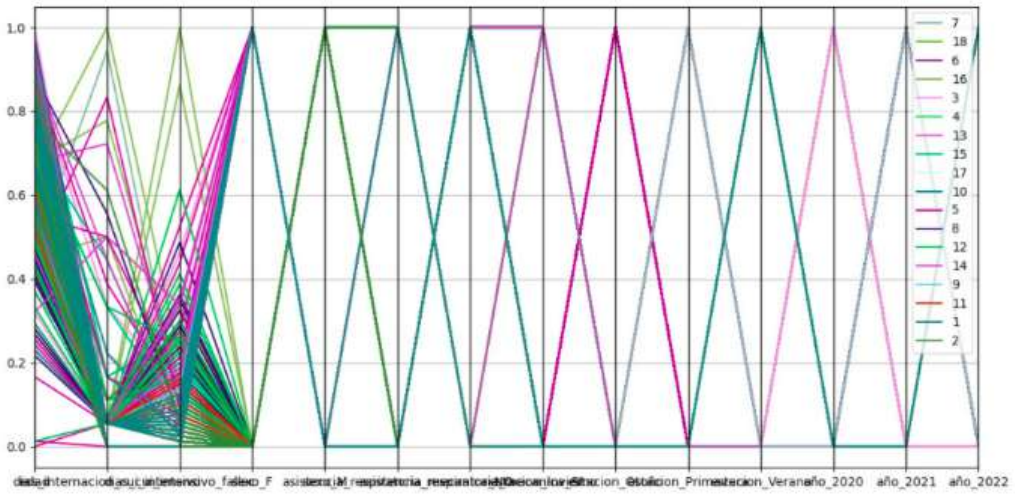


Figura 24

Gráfico de coordenadas paralelas para los centroides de los 18 grupos del dataset 2 con todos sus atributos

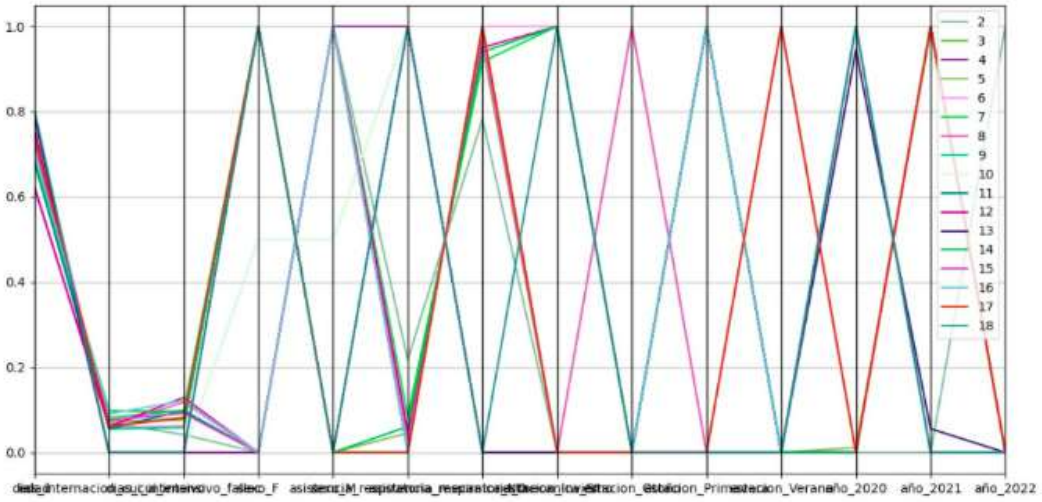
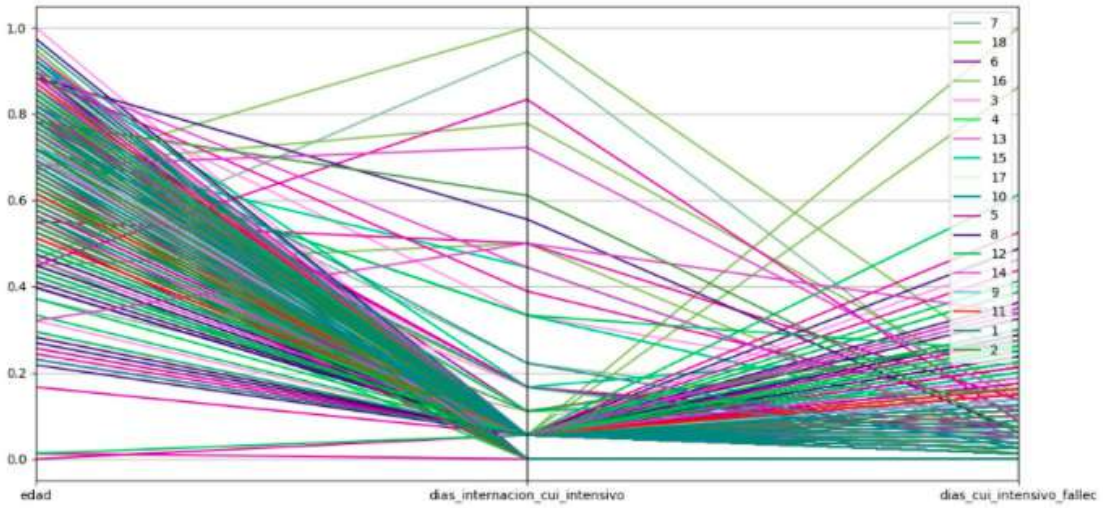


Figura 25

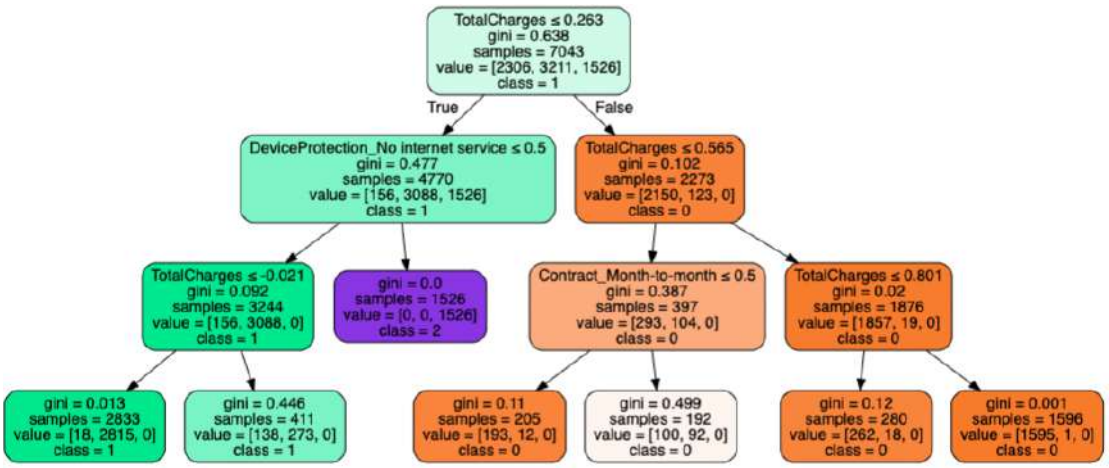
Gráfico de coordenadas paralelas para los centroides de los 18 grupos del dataset 2 con sólo atributos numéricos



En segundo lugar se evaluó la posibilidad de utilizar árboles de decisiones para lograr una descripción y etiquetado de los *clusters* (ver ejemplo en la **Figura 26**). El problema fue el mismo que en el caso anterior, la cantidad de *clusters* complejiza el proceso y la legibilidad de las etiquetas por lo que no se profundizó en el tema.

Figura 26

Ejemplo de árbol de decisión para etiquetado de 7 clusters. Cuanto más cantidad de clusters el árbol resulta más complejo de interpretar.



Nota. Tomado de *Interpreting Cluster — mix of data science and intuition*, por Pranay D., https://towardsdatascience.com/interpreting-clusters-29975099eea1

Por último, se decidió utilizar gráficos estadísticos como el diagrama de caja y bigote para descripción de atributos numéricos y gráficos de torta para los categóricos. Se busca que a través de estos se pueda visualizar de una manera un poco más sencilla las características de cada *cluster* y así facilitar su análisis. Los gráficos de cada *dataset* se encuentran en el **Anexo 1**.