# CLASIFICACIÓN DE DATOS MEDIANTE TDABC
## MONOGRAFÍA DE GRADO

CAMILO ZAMORA BERRÍO
**ASESOR:** PROF. DR. BERNARDO URIBE JONGBLOED

Abstract. Topological data analysis is a branch of computational topology that aims to extract information from data by expressing it as a topological space. Kindelan et al. propose a classifier that uses topological data analysis to classify data, which will be applied and analyzed in this document.

Resumen. El análisis topológico de datos es una rama de la topología computacional que tiene como propósito inferir información de los datos, representándolos como un espacio topológico. Kindelan et al. proponen un método de clasificación de datos por análisis topológico de datos, el cual será aplicado y analizado en este documento.

## 1. Introducción

El análisis topológico de datos (TDA por sus siglas en inglés) es una subárea de la topología computacional que se enfoca en el estudio de datasets mediante el uso de herramientas topológicas, principalmente mediante el estudio de la homología persistente.[19]

Resulta extraño pensar en esta noción de "los datos tienen forma", aunque, en realidad, la mayoría de los datasets pueden ser expresadas en términos de nubes de puntos en un espacio métrico haciendo posible analizar sus propiedades topológicas mediante el uso de complejos simpliciales abstractos, como el complejo de Vietoris-Rips, el complejo Čech, entre otros.

Una de las muchas aplicaciones del análisis topológico de datos es la clasificación mediante modelos de machine learning. No todos los problemas de clasificación de datos son tan sencillos como parecen, a veces nos podemos encontrar con datasets en las que se presenta una o más clases "minoritarias", en otras palabras, de las que se poseen muy pocas muestras provocando un desbalance de datos. Un ejemplo posible de un problema de este tipo es intentar clasificar entre pacientes que presentan una enfermedad y los que no, teniendo una cantidad muy limitada de casos positivos. También, podemos encontrarnos con datos que se han clasificado incorrectamente o datos faltantes. Esos tipos de problemas son bastante comunes y no son sencillos de solucionar, es por ésto que es importante presentar métodos para solucionarlos.

Kindelan et al. (2021) resaltan que, generalmente el análisis topológico de datos se usa como un paso preliminar al machine learning, y presenta la posibilidad de aprovechar por completo de el pipeline del análisis topológico de datos y no sólamente de sus resultados ayude a resolver algunos de los problemas anteriormente mencionados.

Así, presentan el método TDABC por sus siglas en inglés, un método de clasificación por análisis topológico de datos que permite clasificar datos desbalanceados sin necesidad de capas adicionales de machine learning.

En este sentido, el propósito de éste documento es estudiar, aplicar y analizar el método TDABC, comparándolo en efectividad y eficiencia con otros algoritmos clasificadores como el k-ésimo vecinos más cercanos (k-NN) o k-ésimo vecinos más cercanos con distancia ponderada (wk-NN).

## 2. Topología

### 2.1. **Introducción.**

Al pensar en las propiedades cualitativas de los espacios topológicos, una de las primeras y más obvias que vienen a la mente es su descomposición en componentes conexas por caminos. Al hacer esto en nuestro espacio topológico, lo que tenemos es una partición del mismo, y su cardinalidad es una invariante topológica cualitativa, que nos brinda información relevante acerca del espacio topológico. De la misma forma, se puede contar el número de ciclos únicos que tiene un espacio, veamos

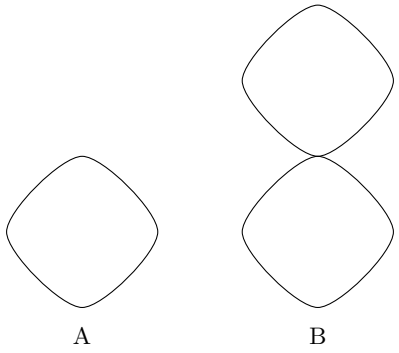



Figura 1. Dos espacios topológicos con distintos tipos de homotopía.

Podemos ver que, solo con deformaciones nos es imposible transformar $B$ en $A$, necesitamos un corte.

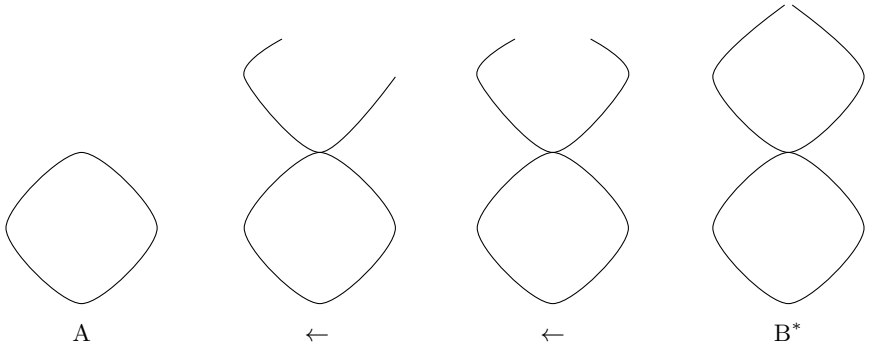



Figura 2. Homotopía entre $B^*$ y $A$.

Por lo que en este sentido, algunos ciclos pueden representar los "huecos" que tiene el espacio, de la misma manera que, las diferentes componentes conexas por caminos representan los "huecos" 0-dimensionales del espacio. Luego, es la topología algebráica la que se encarga de generalizar estos conceptos.

### 2.2. **Homología singular.**

En esta sección sentaremos las bases teóricas de la homología singular, la más general de las teorías de la homología, y a su vez, la más difícil de computar de manera algorítmica. Las siguientes definiciones fueron tomadas en su mayoría de [2], [14] y [20].

**Definición 2.1.** Sean $X, Y$ espacios topológicos y $f, g : X \to Y$ continuas, se dice que son ***homótopas*** si existe una función continua $H : X \times [0,1] \to Y$ tal que $H(x,0) = f(x)$ y $H(x,1) = g(x)$ para todo $x \in X$. En este caso, se dice que $H$ es una ***homotopía*** y lo denotaremos por $f \simeq g$. Además, $\simeq$ es una relación de equivalencia entre espacios topológicos.

**Definición 2.2.** Dos espacios topológicos $X, Y$ son del mismo ***tipo de homotopía*** si existen $f : X \to Y$ y $g : Y \to X$ continuas tales que

$$g \circ f \simeq 1 : X \to X$$

$$f \circ g \simeq 1 : Y \to Y$$

En este caso, se dice que $X$ y $Y$ son ***homotópicamente equivalentes*** y lo denotaremos por $X \simeq Y$. Llamaremos a $f$ y $g$ ***equivalencias homotópicas***.

**Definición 2.3.** Un conjunto de $n+1$ puntos en $\mathbb{R}^m$ con $n \leq m$

$$X := \{x_i \in \mathbb{R}^m | i = 0, 1, 2, \ldots, n\}$$

es ***linealmente independiente*** si $\{\vec{v}_j | \vec{v}_j = v_j - v_0, j = 1, 2, \ldots, n\}$ es un conjunto de vectores linealmente independientes.

**Definición 2.4.** Dado un conjunto de puntos $X$ en un espacio euclídeo $E$, decimos que $X$ es ***convexo*** si contiene todos los segmentos de línea que conectan cualquier par de puntos $x_i, x_j \in X$.

**Definición 2.5.** Dado un conjunto de puntos $X$ en un espacio euclídeo $E$, se define la ***envolvente convexa*** de $X$ como la intersección de todos los subconjuntos convexos $\Omega$ de $E$ tal que $\Omega \subseteq X$.

$$conv(X) := \bigcap_{\substack{\Omega \subseteq E \\ \Omega \text{ convexo}}} \Omega$$

**Definición 2.6.** Un ***n-simplex*** es la envolvente conexa de un conjunto de $n+1$ puntos linealmente independientes.
Denotamos a este simplex

$$[v_0, v_1, \ldots, v_n] := \left\{ \sum_{i=0}^{n} t_i v_i \middle| \sum_{i=0}^{n} t_i = 1 \text{ y } t_i \geq 0, i = 0, 1, \ldots, n \right\}$$

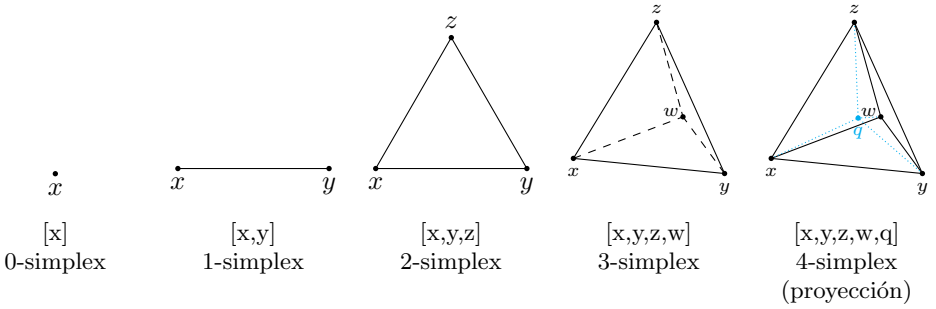



FIGURA 3. Ejemplos de símplices de dimensiones bajas

**Definición 2.7.** Llamamos ***n-simplex estándar*** $\triangle_n$ al simplex generado por la base canonica de $\mathbb{R}^{n+1}$:

$$\triangle_n = [e_0, e_1, \ldots, e_n] = \left\{ (x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1} \middle| 0 \leq x_i \leq 1, \sum_{i=0}^{n} x_i = 0 \right\}$$

**Definición 2.8.** Una ***orientación*** de $\triangle_n$ es un ordenamiento lineal de sus vértices, un recorrido de sus vértices. Por ejemplo, veamos la orientación $e_2 < e_1 < e_0$ de $\triangle_2$ nos da un recorrido en el sentido de las manecillas del reloj.

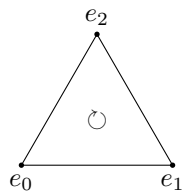



FIGURA 4. Orientación de $\triangle_2$ en el sentido de las manecillas del reloj.

Es claro que varios ordenamientos pueden dar la misma orientación, en este caso $e_2 < e_1 < e_0$ es equivalente a $e_1 < e_0 < e_2$ y a $e_0 < e_2 < e_1$.

**Definición 2.9.** Dos ordenamientos de $\triangle_n$ son ***equivalentes*** si al verlos como permutaciones de $\{e_0, e_1, \ldots, e_n\}$ tienen la misma paridad, de lo contrario, son ***opuestos***.

Dado una orientación de $\triangle_n$, se define una ***orientación inducida*** de sus caras, orientando la $i$-ésima cara de la siguiente manera

$$(-1)^i [e_0, \ldots, \hat{e_i}, \ldots, e_n]$$

donde $-[e_0, \ldots, \hat{e_i}, \ldots, e_n]$ equivale a eliminar el vértice $e_i$ y asignarle la orientación opuesta.

También, se define la ***frontera*** de $\triangle_n$ como $\bigcup_{i=0}^{n} [e_0, \ldots, \hat{e_i}, \ldots, e_n]$ y su ***frontera orientada*** como $\bigcup_{i=0}^{n} (-1)^i [e_0, \ldots, \hat{e_i}, \ldots, e_n]$.

**Definición 2.10.** Sea $X$ un espacio topológico y $\triangle_n$ el n-símplex estándar, definimos un ***n-simplex singular*** como una función continua $\sigma_n : \triangle_n \to X$. Es importante destacar que el hecho que la función sólo requiera ser continua y nada más hace prácticamente imposible computar la homología singular en la mayoría de casos.

**Definición 2.11.** Sea $X$ un espacio topológico. Para todo $n \geq 0$, se define $S_n(X)$ como el grupo abeliano libre cuya base son los $n-$simplices singulares en $X$. Defina $S_{-1}(X) = 0$.

Los elementos de $S_n(X)$ son llamados ***n-cadenas (singulares)*** en $X$.

**Definición 2.12.** Para cada $n, i$, definimos el mapeo de la ***i-ésima cara*** de $\triangle_n$ como

$$\epsilon_i = \epsilon_i^n : \triangle_{n-1} \to \triangle_n$$

como el mapa afín que envía los vértices $\{e_0, \ldots, e_{n-1}\}$ a $\{e_0, \ldots, \hat{e_i}, \ldots, e_n\}$, preservando el ordenamiento.

$$\epsilon_i^n : (t_0, \ldots, t_{n-1}) \to \begin{cases} (0, t_0, \ldots, t_{n-1}) & \text{si } i = 0 \\ (t_0, \ldots, t_{i-1}, 0, t_i, \ldots, t_{n-1}) & \text{si } i \geq 1 \end{cases}$$

**Definición 2.13.** Sea $\sigma : \triangle_n \to X$ continua y $n > 0$, entonces se define la ***frontera*** de $\sigma$ como

$$\partial_n \sigma = \sum_{i=0}^{n} (-1)^i \sigma \epsilon_i^n \in S_{n-1}(X)$$

si $n = 0$, entonces $\partial_0 \sigma = 0$.

Nótese que si $X = \triangle_n$ y $\delta : X \to \triangle_n = 1 : \triangle_n \to \triangle_n$, se tiene que

$$\partial_n(\delta) = \sum_{i=0}^{n} (-1)^i \epsilon_i^n$$

**Teorema 2.14.** *Para todo $n \geq 0$ existe un homomorfismo único*

$$\partial_n : S_n(X) \to S_{n-1}(X)$$

*con $\partial_n \sigma = \sum_{i=0}^{n} (-1)^i \sigma \epsilon_i$ para cada $n-$símplex singular $\sigma$ en $X$.*

*Demostración.* Sean $\sigma, \delta \in S_n(X)$ , se sigue que

$$\begin{aligned}
\partial_n(\sigma + \delta) &= \sum_{i=0}^{n} (-1)^i (\sigma + \delta)\epsilon_i^n \\
&= \sum_{i=0}^{n} (-1)^i (\sigma \epsilon_i^n + \delta \epsilon_i^n) \\
&= \sum_{i=0}^{n} \left[ (-1)^i \sigma \epsilon_i^n + (-1)^i \delta \epsilon_i^n \right] \\
&= \sum_{i=0}^{n} (-1) \sigma \epsilon_i^n + \sum_{i=0}^{n} (-1) \delta \epsilon_i^n \\
&= \partial_n(\sigma) + \partial_n(\delta) \in S_{n-1}(X)
\end{aligned}$$

La unicidad se sigue de la hipótesis. □

Los homomorfismos $\delta_n : S_n(X) \to S_{n-1}(X)$ se les llama ***operadores de frontera***. Así, para cada $X$, tenemos una secuencia de grupos abelianos libres y homomorfismos

$$\ldots \to S_n(X) \xrightarrow{\partial_n} \ldots \to S_1(X) \xrightarrow{\partial_1} S_0(X) \xrightarrow{\partial_0} 0$$

al que llamaremos ***complejo singular*** de $X$ denotado por $(S_*(X), \partial)$ o simplemente $S_*(X)$

**Lema 2.15.** *Sea $k < j$, se cumple que*

$$\epsilon_j^{n+1} \epsilon_k^n = \epsilon_k^{n+1} \epsilon_{j-1}^n : \triangle_{n-1} \to \triangle_{n+1}$$

*Demostración.* Sabemos que $\epsilon_k^n$ mapea $e_t \to e_t$ para $t < k$ y $e_p \to e_{p+1}$ para $k \leq p \leq n-1$. De la misma forma, se sigue que $\epsilon_j^{n+1} \epsilon_k^n$ mapea $e_t \to e_t$ para $t < k$, $e_t \to e_{t+1}$ para $k \leq t+1 < j$ y $e_t \to e_{t+2}$ para $j \leq t+1 \leq n-1$.

Del mismo modo, $\epsilon_{j-1}^n$ mapea $e_t \to e_t$ para $t < j-1$ y $e_t \to e_{t+1}$ para $j-1 \leq t \leq n-1$. También, $\epsilon_k^{n+1}$ mapea $e_p \to e_p$ si $p < k$ y $e_p \to e_{p+1}$ si $k \leq p \leq n-1$. Como $k < j$, se sigue que $\epsilon_k^{n+1} \epsilon_{j-1}^n$ mapea $e_t \to e_t$ cuando $t < k$, cuando $k \leq t < j-1$ mapea $e_t \to e_{t+1}$ lo cual es equivalente a $\epsilon_j^{n+1} \epsilon_k^n$ ya que $t+1 < j \Rightarrow t < j-1$. Por último, cuando $j-1 \leq t \leq n-1$, mapea $e_t \to e_t + 2$ lo cual es equivalente a $\epsilon_j^{n+1} \epsilon_k^n$ ya que $j \leq t+1 \Rightarrow j-1 \leq t$. □

**Teorema 2.16.** *Para todo $n \geq 0$, se tiene que $\partial_n \partial_{n+1} = 0$*

*Demostración.* Dado que $S_{n+1}(X)$ es generado por los $(n+1)$-simplices $\sigma$, basta con mostrar que $\partial\partial\sigma = 0$ para todo $\sigma$.

$$\begin{aligned}
\partial\partial\sigma &= \partial\left(\sum_j (-1)^j \sigma\epsilon_j^{n+1}\right) \\
&= \sum_{j,k} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n \\
&= \sum_{j\leq k} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n + \sum_{k<j} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n \\
&= \sum_{j\leq k} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n + \sum_{k<j} (-1)^{j+k} \sigma\epsilon_k^{n+1}\epsilon_{j-1}^n
\end{aligned}$$

Sea $p = k$ y $q = j-1$

$$\begin{aligned}
&= \sum_{j\leq k} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n + \sum_{p\leq q} (-1)^{p+q+1} \sigma\epsilon_p^{n+1}\epsilon_q^n \\
&= \sum_{j\leq k} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n + (-1)\sum_{p\leq q} (-1)^{p+q} \sigma\epsilon_p^{n+1}\epsilon_q^n \\
&= \sum_{j\leq k} (-1)^{j+k} \sigma\epsilon_j^{n+1}\epsilon_k^n - \sum_{p\leq q} (-1)^{p+q} \sigma\epsilon_p^{n+1}\epsilon_q^n \\
&= 0
\end{aligned}$$

□

**Definición 2.17.** El grupo de ***n-ciclos*** (singulares) en $X$ está denotado y dado por

$$Z_n(X) = ker\ \partial_n$$

El grupo de ***n-fronteras*** en $X$, está denotado y dado por

$$B_n(X) = im\ \partial_{n+1}$$

Es claro que ambos son subgrupos de $S_n(X)$ para todo $n \geq 0$.

**Corolario 2.18.** *Para todo espacio topológico $X$ y todo $n \geq 0$,*

$$B_n(X) \subset Z_n(X) \subset S_n(X)$$

*Demostración.* Sea $\beta \in B_n(X)$, entonces $\beta = \partial_{n+1}\alpha$ para algún $\alpha \in S_{n+1}(X)$. Así, se sigue que $\partial_n(\beta) = \partial_n\partial_{n+1}\alpha = 0$, y por tanto $\beta \in Z_n(X)$ □

Al pensar en la noción presentada anteriormente, podemos considerar los ciclos que no son fronteras (las fronteras son ciclos triviales) como una medida de los "huecos" de un espacio topológico, lo que nos lleva a la siguiente definición.

**Definición 2.19.** Para todo $n \geq 0$, se define el $n$-ésimo ***grupo de homología*** (singular) de un espacio topológico $X$ como

$$H_n(X) := \frac{Z_n(X)}{B_n(X)} = \frac{ker\ \partial_n}{im\ \partial_{n+1}}$$

Dado $z_n \in Z_n(X)$, la clase lateral $z_n + B_n(X)$ es llamada la ***clase de homología*** de $z_n$, y la denotaremos por $clsz_n$.

**Definición 2.19.1.** Sean $\mathcal{C}, \mathcal{D}$ categorías, entonces un ***functor*** $T : \mathcal{C} \to \mathcal{D}$ es una función, es decir, se cumplen las siguientes condiciones:

- $C \in obj\ \mathcal{C}$ implica que $TC \in obj\ \mathcal{D}$
- Si $f : C \to C'$ es un morfismo en $\mathcal{C}$, entonces $Tf : TC \to TC'$ es un morfismo en $\mathcal{D}$ tal que:
  - Si $f, g$ son morfismos en $\mathcal{C}$ y $g \circ f$ está definida, entonces
$$T(g \circ f) = (Tg) \circ (Tf)$$
  - $T(1_C) = 1_{TC}$ para todo $C \in obj\ \mathcal{C}$

Nuestro siguiente objetivo es mostrar que $H_n$ es un functor que envía espacios topológicos a grupos abelianos. (**Top**→**Ab**).

Sean $X, Y$ espacios topológicos y $f : X \to Y$ continua. Si $\sigma : \triangle_n \to X$ es un $n-$simplex en $X$, entonces $f \circ \sigma : \triangle_n \to Y$ es un $n-$simplex en $Y$. Al expandir por linearidad tenemos un homomorfismo $f_\# : S_n(X) \to S_n(Y)$, dado por

$$f_\#(\sum m_\sigma \sigma) = \sum m_\sigma (f \circ \sigma), \text{ donde } m_\sigma \in Z$$

Note que para cada $n \geq 0$ existe un homomorfismo $f_\#$.

**Lema 2.20.** *Sean $X, Y$ dos espacios topológicos y $f : X \to Y$ continua, entonces $\partial_n f_\# = f_\# \partial_n$. En otras palabras, para cada $n \geq 0$, tenemos el siguiente diagrama conmutativo*

$$\begin{array}{ccc} S_n(X) & \xrightarrow{\partial_n} & S_{n-1}(X) \\ \downarrow{\scriptstyle f_\#} & & \downarrow{\scriptstyle f_\#} \\ S_n(Y) & \xrightarrow{\partial_n} & S_{n-1}(Y) \end{array}$$

*Demostración.* Es claro que basta con mostrar que se cumple en la composición con cualquier generador $\sigma$ de $S_n(X)$. Veamos

$$\begin{aligned} f_\# \partial_n \sigma &= f_\#(\sum_{i=0}^{n} (-1)^i \sigma \epsilon_i^n) \\ &= \sum_{i=0}^{n} (-1)^i f_\#(\sigma \epsilon_i^n) \\ &= \sum_{i=0}^{n} (-1)^i f(\sigma \epsilon_i^n) \\ &= \sum_{i=0}^{n} (-1)^i (f\sigma) \epsilon_i^n) \\ &= \partial_n (f\sigma) \\ &= \partial_n f_\# \sigma \end{aligned}$$

□

**Lema 2.21.** *Sean $X, Y$ espacios topológicos y $f : X \to Y$ continua, entonces para todo $n \geq 0$ se cumple*

$$f_\#(Z_n(X)) \subset Z_n(Y) \text{ y } f_\#(B_n(X)) \subset B_n(Y)$$

*Demostración.* Sea $\sigma \in Z_n(X)$, entonces $\partial\sigma = 0$. Se sigue que $\partial f_\# \sigma = f_\# \partial\sigma = 0$ y por tanto $f_\#\sigma \in ker\ \partial_n = Z_n(Y)$.

Sea $\gamma \in B_n(X)$, entonces $\gamma = \partial\delta$ para algún $\delta \in S_{n+1}(X)$, y $f_\#\gamma = f_\#\partial\delta \in im\ \partial_{n+1} = B_n(Y)$. □

**Teorema 2.22.** *Para todo $n \geq 0$, $H_n$ : **Top**→**Ab** es un functor.*

*Demostración.* Sean $X, Y$ espacios topológicos ($X, Y \in obj$ **Top**) y $f : X \to Y$ continuo, definamos

$$H_n(f) : H_n(X) \to H_n(Y)$$

donde si $z_n \in Z_n(x)$, $z_n + B_n(X) \to f_\#(z_n) + B_n(Y)$, esto es

$$H_n(f) : cls\ z_n \to cls\ f_\#(z_n)$$

Es claro que, si $z_n \in Z_n(X)$, $f_\#(z_n) \in Z_n(Y)$ por el lema anterior. Además, esta definición es independiente de la elección del representante ya que

$$f_\#(B_n(X)) \subset B_n(Y)$$

por tanto si $b_n \in B_n(X)$, se sigue que

$$f_\#(z_n + b_n) + B_n(Y) = f_\#(z_n) + f_\#(b_n) + B_n(Y) = f_\#(z_n) + B_n(Y)$$

Y por definición de $H_n$, tenemos que, $H_n(gf) = H_n(g)H_n(f)$. Además, $H_n(1_X)$ es el homomorfismo identidad. □

**Corolario 2.23.** *Sean $X, Y$ espacios topológicos, si $X$ y $Y$ son homeomorfos, entonces $H_n(X) \cong H_n(Y)$ para todo $n \geq 0$.*

*Demostración.* Sea $f$ el homeomorfismo entre $X$ y $Y$. Veamos que

$$\begin{aligned} f^{-1}f &= 1_X \\ H_n(f^{-1}f) &= H_n(1_X) \\ H_n(f^{-1})H_n(f) &= 1_{H_n(X)} \end{aligned}$$

Similarmente, $H_n(f)H_n(f^{-1}) = 1_{H_n(Y)}$. Por tanto, $H_n(X) \cong H_n(Y)$ □

Así, tenemos que cada grupo de homología $H_n(X)$ es una ***invariante*** del espacio $X$, en particular, su rango es una invariante de $X$.

**Definición 2.24.** Para cada $n \geq 0$, definimos el ***n-ésimo número de Betti*** de $X$ como el rango de $H_n(X)$, y lo denotamos por $\beta_k$.

## 2.3. **Homología simplicial.**

Dada a lo complejo que es computar estos grupos de homología, usaremos la homología simplicial, la cual concuerda con la homología singular, cuando el espacio topológico es un complejo simplicial.

**Definición 2.25.** Un ***complejo simplicial abstracto*** es una pareja $(V, \Sigma)$ donde $V$ es un conjunto finito y $\Sigma$ es una familia de conjuntos no vacíos de $V$ tal que $\sigma \in \Sigma$ y $\tau \subseteq \sigma$ implica que $\tau \in \Sigma$. El espacio topológico asociado a un complejo simplicial, denotado por $|(V, \Sigma)|$ es definido usando una biyección $\varphi : V \to \{1, 2, \ldots, N\}$ como el subespacio de $\mathbb{R}^N$ dado por la unión $\bigcup_{\sigma\in\Sigma} c(\sigma)$, donde $c(\sigma) = conv(\{e_{\varphi(s)}\}_{s\in\sigma})$.

De manera intuitiva, un complejo simplicial en un espacio es una expresión del espacio como unión de puntos, segmentos, triángulos y sus análogos de dimensiones mayores. Los complejos simpliciales permiten describir ciertos espacios topológicos de una manera combinatórica bastante simple, además, la homología puede ser computada usando únicamente álgebra lineal de módulos sobre $\mathbb{Z}$.

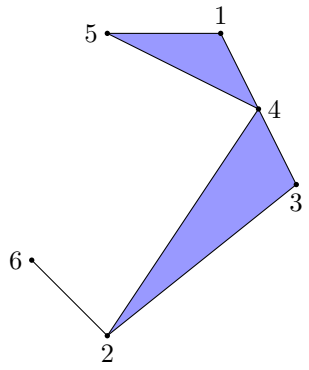



FIGURA 5. Ejemplo de un complejo simplicial en $\mathbb{R}^2$

**Definición 2.26.** Sea $X = (V, \Sigma)$ un complejo simplicial, denotamos como $\Sigma_k$ al conjunto de los $k-$***simplices***, dado por $\Sigma_k = \{\sigma \in \Sigma \mid |\sigma| = k+1\}$. Podemos ver un simplex $\sigma \in \Sigma_k$ como $[v_0, v_1 \ldots, v_k]$, de manera equivalente a la definición anteriormente dada.

**Definición 2.27.** Definimos el ***grupo de k-cadenas*** en $X$ como el grupo abeliano libre en $\Sigma_k$, y lo denotaremos por $C_k(X)$.

**Definición 2.28.** Al asignarle un orden total en el conjunto de vértices $V$, definimos los operadores $d_i : \Sigma_k \to \Sigma_{k-1}$, para $0 \leq i \leq k$, de forma que

$$d_i(\sigma) = \sigma - \{v_i\} = [v_0, v_1, \ldots, \hat{v_i}, \ldots, v_k]$$

donde $v_j \in V$ están ordenados por el orden total y eliminamos el $i$-ésimo vértice, $v_i$. Luego, podemos definir los operadores lineales $\partial_k : C_k(X) \to C_{k-1}(X)$ por

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i [v_0, v_1, \ldots, \hat{v_i}, \ldots, v_k] = \sum_{i=0}^{k} (-1)^i d_i$$

Como $C_k(X)$ están dados por las bases $\Sigma_k$, estos operadores pueden ser expresados como matrices $D(k)$ en la que las columnas están parametrizadas por $\Sigma_k$ y las filas por $\Sigma_{k-1}$, y para las que si $\sigma \in \Sigma_k$ y $\tau \in \Sigma_{k+1}$, $D(k)_{\tau\sigma}$ es 0 si $\tau \not\subset \sigma$, y $(-1)^i$ si $\tau \subseteq \sigma$ y $\tau = d_i(\sigma)$. Es sencillo ver que $\partial_k \circ \partial_{k+1} = 0$ de forma equivalente a la prueba anterior, por lo que $im\ \partial_{k+1} \subseteq ker\ \partial_k$.

**Definición 2.29.** Con esto podemos finalmente definir el $k$-ésimo ***grupo de homología simplicial*** con coeficientes en $\mathbb{Z}$, $H_k^{\triangle}(X, \mathbb{Z})$ por

$$H_k^{\triangle}(X, \mathbb{Z}) \cong \frac{ker\ \partial_k}{im\ \partial_{k+1}}$$

Esta definición puede ser reemplazada por el resultado de manipulación de la familia de matrices $\{D(k)\}_{k\geq 0}$, los resultados de estas manipulaciones son siempre la forma normal de Smith construida a partir de las $D(k)$. Además, es claro que $C_n \subseteq S_n$, esto nos induce el isomorfismo

$$H_k^{\triangle}(X, \mathbb{Z}) \cong H_k(|X|)$$

Así, es posible computar la homología de manera algorítmica, aunque el cómputo de la forma normal de Smith puede ser muy lento, ya que tiene complejidad de tiempo polinómica, aún para algoritmos eficientes como el algoritmo presentado por Saunders en [5], cuya complejidad de tiempo es, un caso favorable para su algoritmo de caja negra, $\mathcal{O}(d^2e^2n^2)$ para una matriz $n \times n$ sobre $F[z]/(f^e)$, y $f^e$ es una potencia de $f \in F[z]$ irreducible de grado $d$. Ésto, junto con el hecho de que $|\Sigma_k| \leq \binom{|V|}{k+1}$, hace que en muchas ocasiones el cómputo de la homología sea infinitamente lento.

**Teorema 2.30.** *Sean $X, Y$ espacios topológicos y $f : X \to Y$ una equivalencia homotópica, $f$ induce un isomorfismo $f_{\#} : H_n(X) \to H_n(Y)$ dado por el homomorfismo inducido en los complejos de cadenas. Decimos entonces que la homología presenta una invariancia homotópica (Véase: [7]).*

2.4. **Complejos simpliciales y coberturas.**

Debido a lo explicado anteriormente, es natural considerar la construcción de complejos simpliciales sobre nuestro espacio topológico para así poder calcular de manera algorítmica su homología. Esto no siempre es una tarea fácil, pero una manera en la que podemos garantizar que nuestro complejo simplicial calcule de manera precisa la homología de nuestro espacio, es construir una homotopía entre nuestro espacio y el complejo simplicial, o una homotopía entre nuestro espacio y un espacio homotópicamente equivalente a nuestro complejo simplicial.

**Definición 2.31.** Sea $X$ un espacio topológico y $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ una cobertura de $X$, el ***nervio*** de $\mathcal{U}$, que denotaremos por $N\mathcal{U}$ es el complejo simplicial abstracto con $A$ como su conjunto de vértices y donde $\{\alpha_0, \ldots, \alpha_k\}$ genera un $k$-simplex si y sólamente si $\bigcap_{i=0}^{k} U_{\alpha_i} \neq \emptyset$

**Definición 2.32.** Sea $X$ un espacio topológico, decimos que $X$ es ***contráctil o contractible*** si existe alguna aplicación constante $c : X :\to X$, esto es $c(x) = p$, para todo $x \in X$ y con $p \in X$ tal que $1_X \simeq c$ donde $1_X$ es la aplicación identidad sobre $X$.

**Definición 2.33.** Sea $\{U_i\}$ una cobertura abierta de un espacio topológico $X$. Decimos que es ***enumerable*** si existe una colección de funciones continuas

$$\varphi_i : X \to [0, 1] \subseteq \mathbb{R}$$

tales que:

- El soporte de la $i-$ésima función, $Supp(\varphi_i)$, está contenido en el $i$-ésimo subconjunto.
$$Supp(\varphi_i) \subset U_i$$
- Sea $x \in X$, $\varphi_i(x) \neq 0$ sólamente en un número finito de $\varphi_i$.
- $\forall x \in X$, $\sum_i \varphi_i(x) \equiv 1$.

**Teorema 2.34.** *Sea $X$ un espacio topológico y $\mathcal{U}$ una cobertura abierta y numerable de $X$ tal que para todo $\emptyset \neq S \subseteq A$, $\bigcap_{s \in S} U_s$ es o contractible, o vacía, entonces $N\mathcal{U} \simeq X$. Éste resultado se conoce como el Teorema del Nervio (Véase [1]).*

**Definición 2.35.** Sea $X$ un espacio métrico. Entonces tenemos la cubierta dada por $\mathcal{B}_\varepsilon = \{B_\varepsilon(x)\}_{x \in X}$, para algún $0 < \varepsilon \in \mathbb{R}$. Sea $V \subseteq X$ tal que $X = \bigcup_{v \in V} B_\varepsilon(v)$, es posible construir $NV$. A esta construcción la llamaremos ***complejo de Čech*** con vértices en $V$ y lo denotaremos por $\check{C}(V, \varepsilon)$.

**Definición 2.36.** Una pareja $(M, g)$ es una ***variedad de Riemann*** donde $M$ es una variedad suave y $g$ es la ***métrica de Riemann***. La métrica de Riemann es el campo tensorial $g : \mathcal{T}^2(M) \to \mathbb{R}$, esto es, si $X, Y \in T_p(M)$ se cumple que:

- $g(X,Y) = g(Y,X)$.(Simetría)
- $g(X,X) \geq 0$

**Teorema 2.37.** *Sea $M$ una variedad de Riemann, entonces existe $e > 0$ tal que $\check{C}(X,\varepsilon)$ tal que $\check{C}(X,\varepsilon) \simeq M$ para todo $\varepsilon \leq e$. Además, existe un subconjunto finito $V \subseteq M$ tal que $\check{C}(X,\varepsilon) \supseteq \check{C}(V,\varepsilon) \simeq M$.*

Uno de los problemas más grandes del complejo de Čech es que es muy ineficiente de computar (véase [10]). Dado ésto, resulta conveniente definir una variación del complejo de Čech que solo necesite información de los $1-$simplices para construirlo en su totalidad.

**Definición 2.38.** Sea $(X,d)$ un espacio métrico. Definimos el ***complejo de Vietoris-Rips*** en $X$ con parámetro $\varepsilon$ como el complejo simplicial con vértices en $X$ en el que $\{x_0, x_1, \dots x_k\} \subseteq X$ genera un $k$-simplex sí y sólamente si $d(x_i, x_j) \leq \varepsilon$ para todo $0 \leq i,j \leq k$, y lo denotaremos como $VR(X,\varepsilon)$. Es claro que ambos complejos pueden verse como subcomplejos del simplex completo en el conjunto $X$, además es claro que $\check{C}(X,\varepsilon) \subseteq VR(X,2\epsilon) \subseteq \check{C}(X,2\varepsilon)$. Comparemos los complejos en el conjunto de vértices

$$V = \left\{ \left(\frac{1}{2}, 2\right), (1,1), \left(-\frac{6}{5}, 0\right), (-1,2) \right\} \subseteq \mathbb{R}^2$$

.

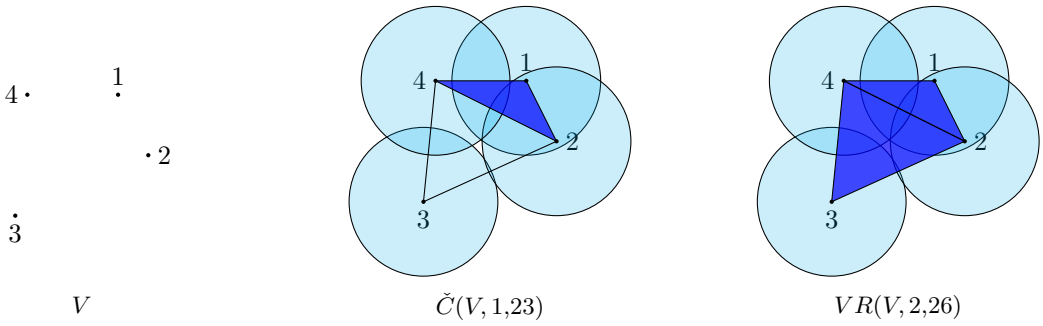



FIGURA 6. Comparación del complejo de Čech y de Vietoris Rips.

Para la implementación del algoritmo para generar el complejo de Vietoris Rips sobre una nube de puntos en $(\mathbb{R}^n, d)$ donde $d$ es la métrica euclídea se usó el lenguaje Python. Cabe destacar que esta implementación es increíblemente costosa de computar, ya que calculo y almaceno todas las combinaciones en memoria, además de verificar las distancias a parejas para todas las combinaciones, lo cual fue uno de los mayores obstáculos que tuve al estudiar la homología y su cómputo. Véase [18] para una implementación eficiente del complejo de Vietoris-Rips.

```
import numpy as np
from scipy.spatial import distance
from itertools import combinations
""" Funcion que genera el complejo de Vietoris-Rips de distancia epsilon con vertices en
    nodes. Regresa el complejo simplicial  como lista de simplices, donde un k-simplex es
    una lista de k+1 vertices en nodes.
"""

def rips(nodes,epsilon):
    simpcomplex=[]
    RipsV2(nodes,nodes.shape[1]+1, epsilon, simpcomplex)
```

```
    return np.asarray(simpcomplex)
""" Metodo recursivo para generar los (k-1)-simplices, buscando las combinaciones de k
    puntos que estan a lo sumo a distancia epsilon de los demas, y concatenandolos al
    complejo simplicial.
"""

def RipsV2(nodes, k, epsilon, simpcomplex):
    if 1<=k:
        temp = np.array([list(x) for x in combinations(nodes,k)])
        for combination in temp:
            forall = True;
            for point in combination:
                for point2 in combination:
                    if distance.euclidean(point,point2) > epsilon:
                        forall = False
            if forall:
                simpcomplex.append(np.array(combination))
        RipsV2(nodes, k - 1, epsilon, simpcomplex)
```

Un problema que tiene esta construcción para el estudio de la homología es que es enteramente dependiente de una elección apropiada de parámetro, lo cual es especialmente complicado en situaciones en la que no tenemos mucha información adicional además de las nubes de puntos. Debido a esto, resulta interesante estudiar la homología en todos los posibles valores del parámetro, lo que nos lleva a la homología persistente.

### 2.5. **Homología persistente.**

Es claro que cuando $\varepsilon \leq \varepsilon'$, tenemos una inclusión $VR(X,\varepsilon) \subseteq VR(X,\varepsilon')$, así, tenemos la siguiente definición.

**Definición 2.39.** Sea $K$ un complejo simplicial y $K' \subseteq K$. Decimos que $K'$ es un ***sub-complejo simplicial*** de $K$ si $K'$ es un complejo simplicial.

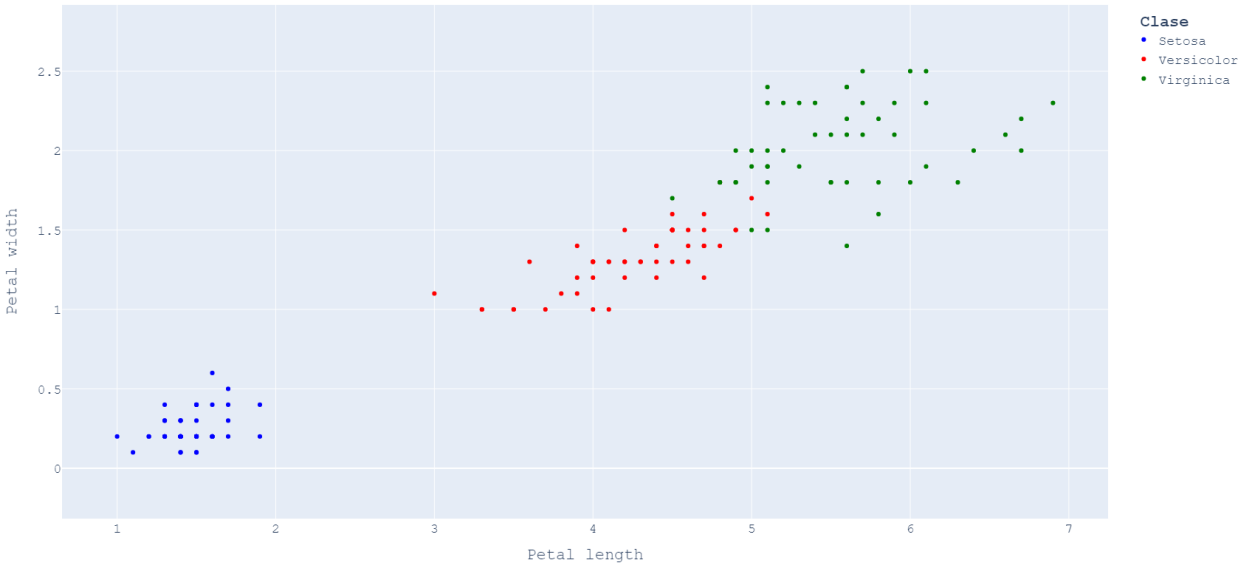
**Definición 2.40.** Sea $K$ un complejo simplicial, una ***filtración*** $F$ en $K$ es una sucesión creciente de sub-complejos de $K$. En otras palabras, una familia de sub-complejos $\{K_i\}_{i\in I}$ donde $\mathbb{N} \supseteq I = \{0,1,2,\ldots,n\}$ tal que

$$\emptyset \subseteq K_0 \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_n = K$$

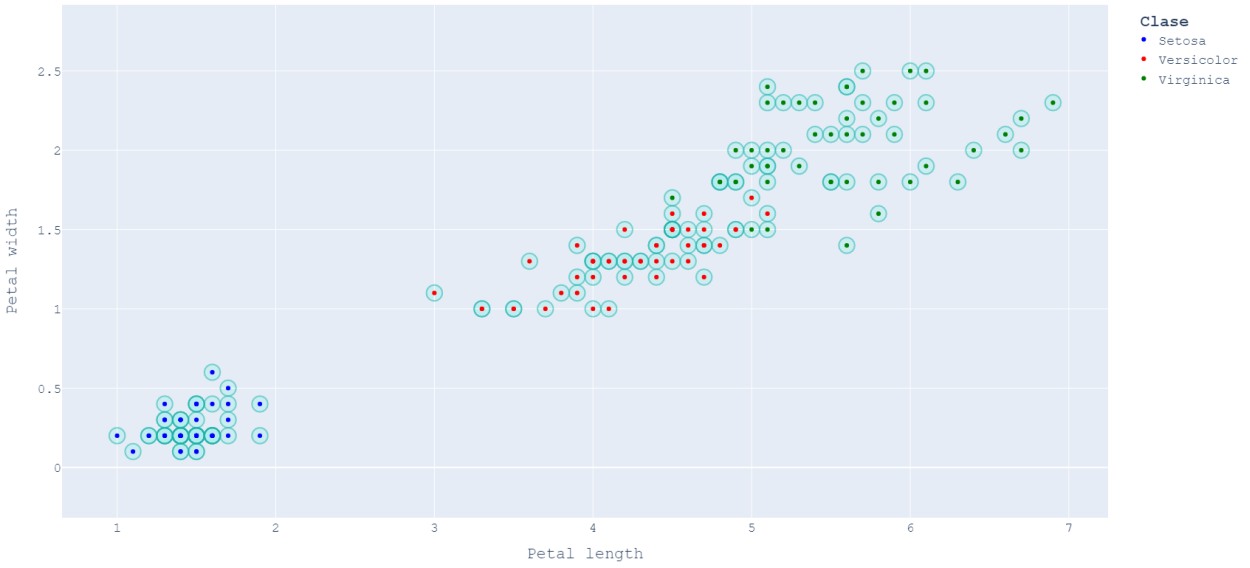
En la mayoría de los casos, la filtración está determinada por una función de pesos dada por la métrica. Un ejemplo de una función de pesos es la siguiente:

$$w(\sigma) := \begin{cases} 0 & \text{si } \sigma \text{ es un 0-simplex.} \\ d(v_0,v_1) & \text{si } \sigma = [v_0,v_1] \text{ (es un 1-simplex).} \\ \text{máx}_{\tau\subseteq\sigma}\, w(\tau) & \text{en cualquier otro caso.} \end{cases}$$

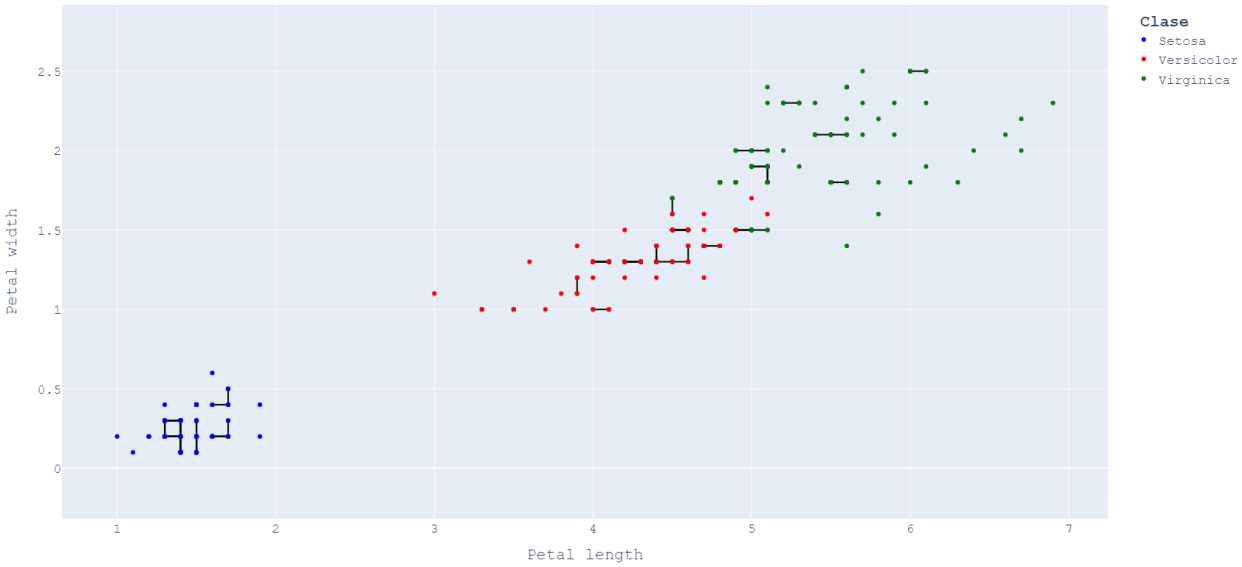
Es claro que, al variar $\varepsilon$ en un complejo de Vietoris-Rips, obtenemos una filtración $\{VR(X,\varepsilon)\}_{\varepsilon\geq 0}$. Veamos un ejemplo en los datos de pétalos de las flores en el dataset *Iris*.



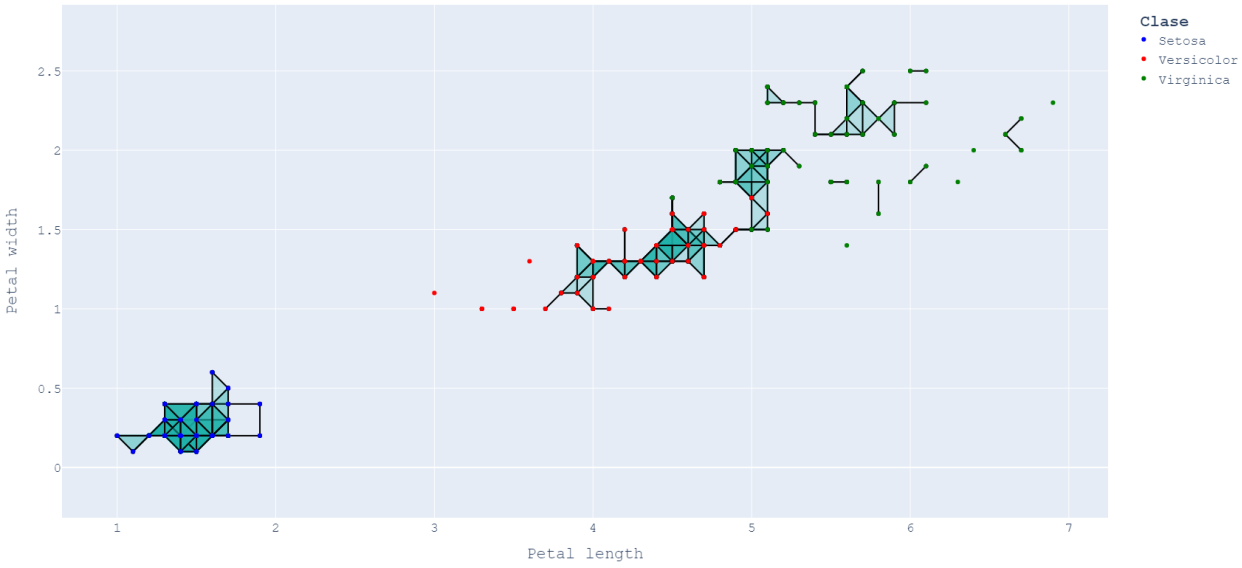

$V = VR(V, 0)$, $\beta_0 = 150$, $\beta_1 = 0$



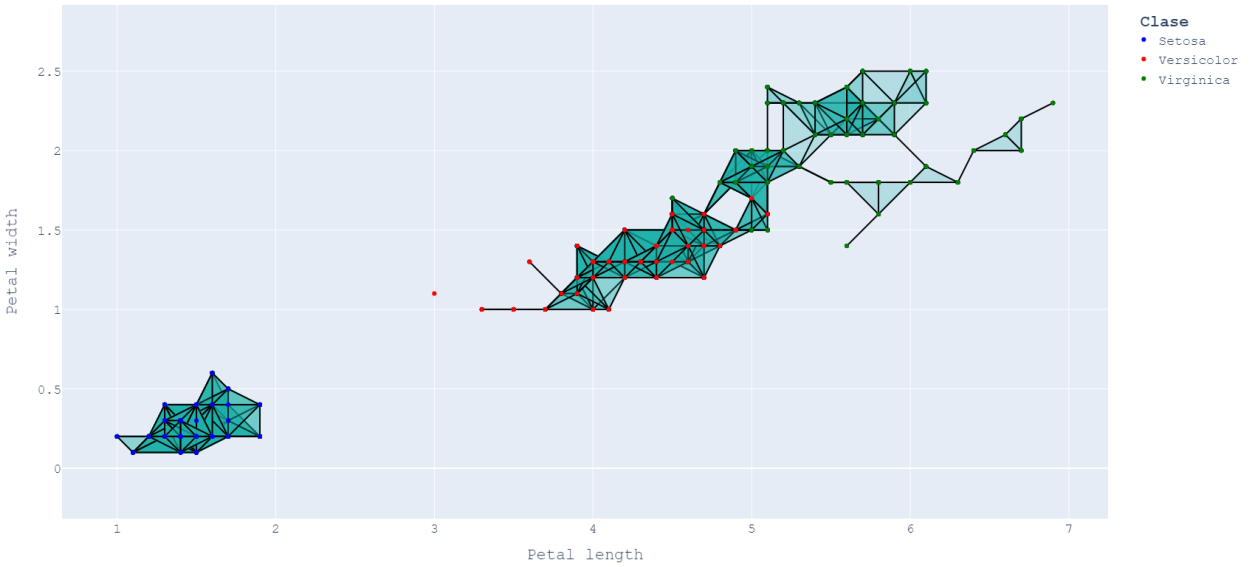

↓



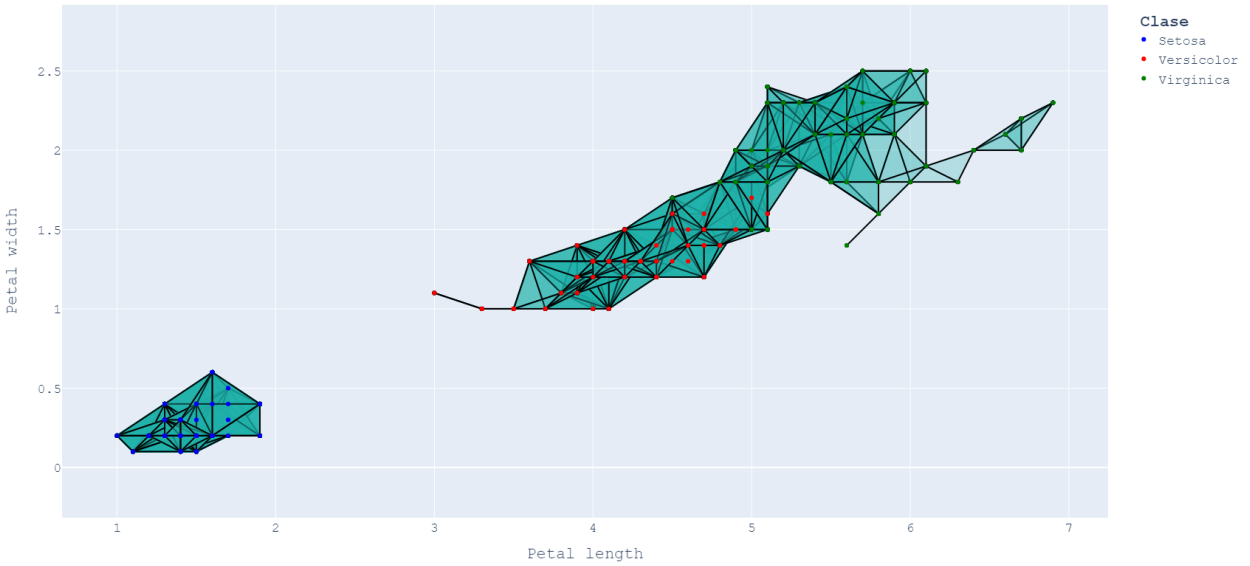

$VR(V,0,1)$, $\beta_0 = 70$, $\beta_1 = 1$



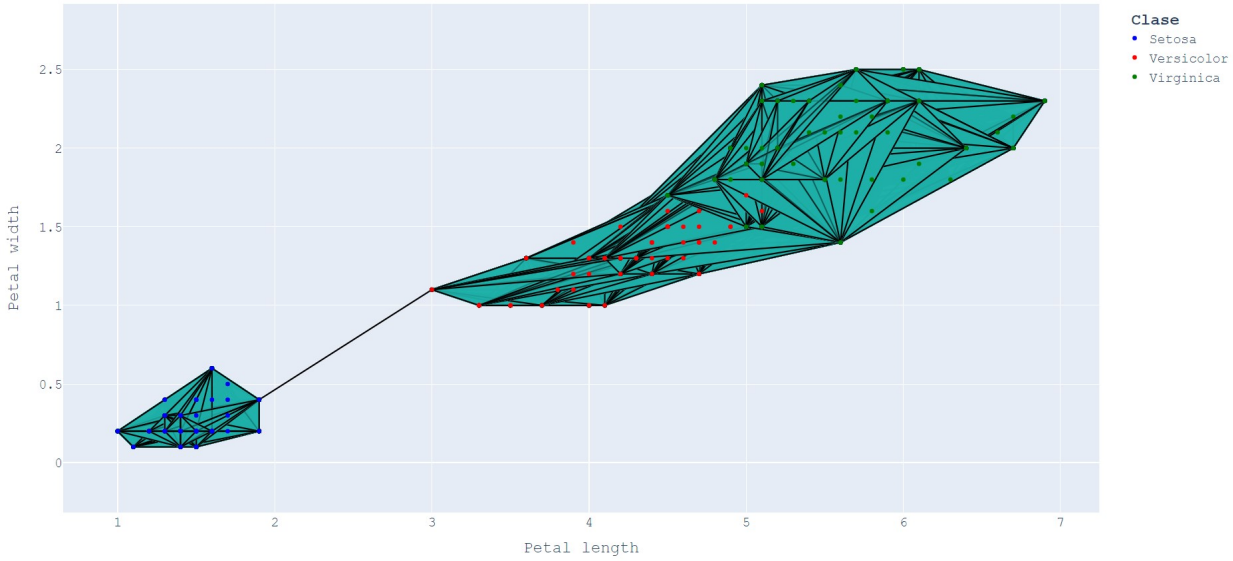

$VR(V,0,2)$, $\beta_0 = 16$, $\beta_1 = 1$




$VR(V,0{,}3)$, $\beta_0 = 3$, $\beta_1 = 3$




$VR(V,0{,}4)$, $\beta_0 = 2$, $\beta_1 = 0$




$VR(V, 1{,}31)$, $\beta_0 = 1$, $\beta_1 = 0$

FIGURA 10. Ejemplo de una filtración

Además, si $\varepsilon \leq \varepsilon'$ tenemos el homomorfismo $\iota : VR(X, \varepsilon) \hookrightarrow VR(X, \varepsilon')$ dado por $\iota(x) = x$; el cual dado que $H_k^{\triangle}(X, \mathbb{Z})$ es un functor, nos induce un homomorfismo $H_k^{\triangle}(VR(X, \varepsilon), \mathbb{Z}) \to H_k^{\triangle}(VR(X, \varepsilon'), \mathbb{Z})$ para cada $k \geq 0$. Así, en lugar de obtener un grupo de homología, obtenemos una familia de grupos de homología $\{H_k^{\triangle}(VR(X, \varepsilon), \mathbb{Z})\}_{\varepsilon \geq 0}$.

**Definición 2.41.** Sea $\underline{\mathcal{C}}$ una categoría y $\mathcal{P}$ un conjunto parcialmente ordenado. Llamaremos $\underline{\mathcal{P}}$ a la categoría con conjunto de objetos $\mathcal{P}$ y con un morfismo único de $x$ a $y$ siempre que $x \leq y$. Luego, definimos un ***$\mathcal{P}$-objeto de persistencia*** en $\underline{\mathcal{C}}$ como un functor $\phi : \underline{\mathcal{P}} \to \underline{\mathcal{C}}$. En otras palabras, una familia $\{c_x\}_{x \in \mathcal{P}}$ tal que $c_x \in \text{obj } \underline{\mathcal{C}}$ con morfismos $\phi_{xy} : c_x \to c_y$ siempre que $x \leq y$ tal que $\phi_{yz} \circ \phi_{xy} = \phi_{xz}$.

Es fácil ver que los $\mathcal{P}$-objetos de persistencia en $\underline{\mathcal{C}}$ son a su vez una categoría. Sean $\Phi, \Psi$ dos $\mathcal{P}$-objetos de persistencia sobre $\underline{\mathcal{C}}$. Entonces, el morfismo $\Phi \to \Psi$ es una transformación natural.

**Definición 2.41.1.** Sean $F, G : \mathcal{C} \to \mathcal{D}$, entonces una ***transformación natural*** $\eta$ de $F$ a $G$ es una familia $\{\eta_c : F(c) \to G(c)\}_{c \in \text{obj } \mathcal{C}}$ llamados componentes de $\eta$ en $c$ tal que para cada morfismo $f : x \to y$, con $x, y \in \text{obj } \mathcal{C}$ se tiene que

$$G(f) \circ \eta_x = \eta_y \circ F(f) : F(x) \to G(x)$$

En otras palabras, el siguiente diagrama conmuta en $\mathcal{D}$

$$\begin{array}{ccc} F(x) & \xrightarrow{F(f)} & F(y) \\ \downarrow{\scriptstyle \eta_x} & & \downarrow{\scriptstyle \eta_y} \\ G(x) & \xrightarrow{G(f)} & G(y) \end{array}$$

Denotaremos la categoría de $\mathcal{P}$-objetos de persistencia en $\underline{\mathcal{C}}$ por $\mathcal{P}_{pers}(\underline{\mathcal{C}})$, finalmente tenemos que si $f : \mathcal{P} \to \mathcal{Q}$ es un mapeo que preserva el orden parcial, tenemos un functor $f^* : Q_{pers}(\underline{\mathcal{C}}) \to \mathcal{P}_{pers}(\underline{\mathcal{C}})$ dado por $f^*(\Phi) = \Phi \circ \underline{f}$ donde $\underline{f}$ es $f$ como functor $\underline{\mathcal{P}} \to \underline{\mathcal{Q}}$.

En este sentido, es claro que $\{VR(X, \varepsilon)\}_{\varepsilon \geq 0}$ como definido antes es un complejo simplicial $\mathbb{R}$-persistente sobre $X$, y podemos calcular el complejo de cadenas $\mathbb{R}$-persistente y por tanto el grupo de homología $\mathbb{R}$-persistente $\{H_k^{\triangle}(VR(X, \varepsilon), \mathbb{Z})\}_{\varepsilon \geq 0}$.

Si bien no existe un teorema de clasificación de grupos abelianos $\mathbb{R}$-persistentes, se puede clasificar sobre una subcategoría de la categoría de $F$-espacios vectoriales $\mathbb{N}$-persistentes, donde $F$ es un campo. (Véase:[20])

Sea $\{A_n\}$ un grupo abeliano $\mathbb{N}$-persistente, definimos una álgebra graduada $\theta(\{A_n\})$ sobre el anillo $\mathbb{Z}[t]$ en el que $t$ es de grado 1 por

$$\theta(\{A_n\}) = \bigoplus_{s \geq 0} A_s$$

donde

$$t \cdot \{\alpha_n\} = \{\beta_n\}, \text{ con } \beta_n = \psi_{n-1,n}(\alpha_{n-1})$$

Tenemos así que $\theta$ es una equivalencia de categorías de $\mathbb{N}_{pers}(\underline{Ab})$ a la categoría de $\mathbb{Z}[t]$-módulos no-negativamente graduados, cuyo functor inverso está dado por $M_* \to \{M_n\}$ donde los morfismos $\psi_{mn}$ son dados por multiplicación por $t^{n-m}$.

**Teorema 2.42.** *Sea $M_*$ un $F[t]$-módulo no-negativamente graduado, entonces hay enteros $\{i_1, \dots i_m\}$,$\{j_1, \dots, j_n\}$, $\{l_1, \dots, l_n\}$ y un isomorfismo*

$$M_* \cong \bigoplus_{s=1}^{m} F[t](i_s) \oplus \bigoplus_{t=1}^{n} (F[t]/(t^{l_t}))(j_t)$$

*donde para cualquier $F[t]$-módulo $N_*$, $N_*(s)$ denota $N_*$ con un desplazamiento de $s$ dimensiones. En otras palabras,*

$$N_*(s)_l = N_{l-s}$$

*. La descomposición es única salvo permutación de factores. Véase [3] para el caso no graduado. El caso graduado se prueba de manera idéntica [2].*

**Definición 2.43.** Sea $\{V\}_n$ un $F$-espacio vectorial $\mathbb{N}$-persistente. Decimos que es ***tameado*** si $V_i$ es de dimensión finita para todo $i$ y si $\psi_{n,n+1} : V_n \to V_{n+1}$ es un isomorfismo para $n$ suficientemente grande.

Entonces, $\theta(\{V_n\}_n)$ es finitamente generado sí y sólamente si $\{V_n\}_n$ es tameado. Luego, para todo $0 \leq m \leq n$ podemos definir un $F$-espacio vectorial $\mathbb{N}$-persistente $U(m, n)$ dado por $U(m, n)_t = 0$ si $t < m$ o $n < t$, y $U(m, n)_t = F$ para $m \leq t \leq n$, y $\psi_{s,t} = 1_F$ para $m \leq s \leq t \leq n$. Esto puede extenderse para $n = +\infty$.

Luego, tenemos que si $\{V_n\}$ es tameado, este se puede descomponer como

$$\{V_n\} \cong \bigoplus_{i=0}^{N} U(m_i, n_i)$$

donde $m_i \in \mathbb{N}$ y $n_i \in \mathbb{N} \cup \{+\infty\}$. La cual es única salvo ordenamiento de los factores. Luego, un ***diagrama de barras*** es el conjunto finito de parejas $(m_i, n_i)$ con $m \in \mathbb{N}$ y $n \in \mathbb{N} \cup \{+\infty\}$. Así, tenemos que los espacios vectoriales $\mathbb{N}$-persistentes están clasificados por sus diagramas de barras salvo insomorfismos.

Ahora, para clasificar nuestros complejos simpliciales $\mathbb{R}-$persistentes, necesitamos una función que que preserve orden parcial $f : \mathbb{N} \to \mathbb{R}$, para así obtener un complejo simplicial $\mathbb{N}$-persitente. Una forma de hacerlo es $f_\varepsilon(n) = n\varepsilon$ para $\varepsilon \in \mathbb{R}$. Otra es, sabiendo que en la filtración solo tenemos un número finito de cambios en el complejo simplicial, entonces sea $\{\varepsilon_0, \varepsilon_1, \dots, \varepsilon_N\}$ el conjunto de parámetros en los que hay cambios en

el complejo ordenado de manera creciente, definimos $g(n) = \varepsilon_n$ para $n \leq N$. El primer caso es de manera intuitiva un muestreo en intervalos constantes sobre la filtración, la otra puede pensarse como un muestreo más específico, hecho acorde a las propiedades de nuestro complejo simplicial.

Finalmente, tenemos un método para estudiar la homología persistente de un complejo simplicial, y una forma de visualizarla en los diagramas de barras o diagramas de persistencia como podemos ver en (Figura 12 y Figura 11, respectivamente) con la filtración presentada como ejemplo anteriormente.(Véase: Figura 10 para una vista detallada a la filtración). En estos diagramas, cada intervalo $(birth, death)$ (o punto en el caso de el diagrama de persistencia) inicia en el valor de filtración en el que aparece una característica topológica y termina en el valor de filtración en el que desaparece, en este sentido, el intervalo es el "tiempo de vida" de la característica topológica.

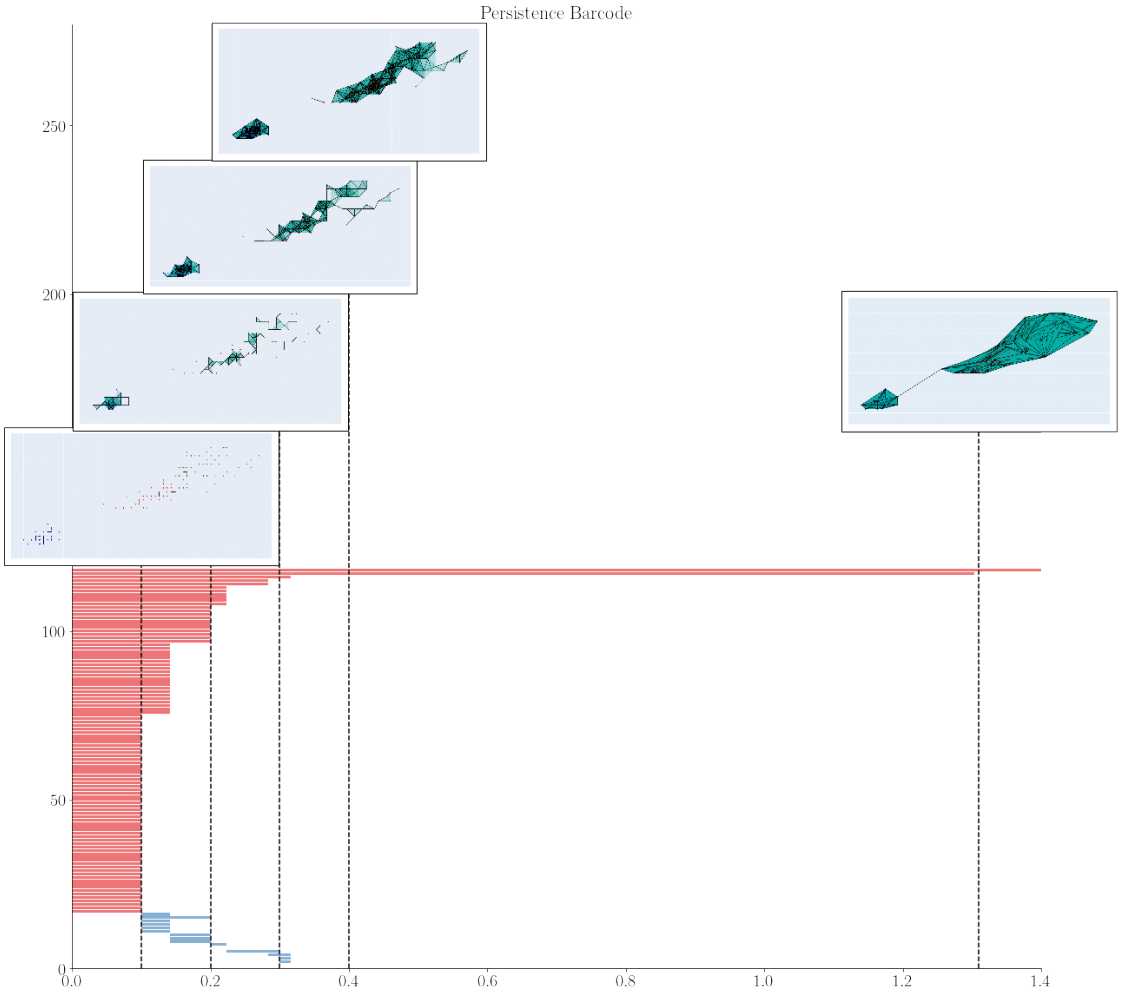



FIGURA 11. Diagrama de barras sobre la filtración.

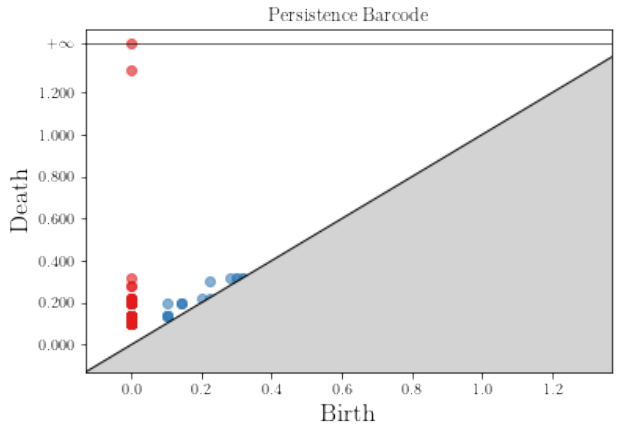



FIGURA 12. Diagrama de persistencia sobre la filtración.

En estas figuras, las barras (o puntos) rojos representan los elementos de $H_0$ (componentes conexas por caminos) y los azules los elementos de $H_1$ (huecos).

Luego, Silva et al.(2011) proponen tres algoritmos para el cálculo de la homología persistente sobre una filtración. Dos de estos son mediante reducción de una matriz de fronteras sobre la filtración, la cual es una matriz $D$ triangular superior en la que $D[i,j]$ está dado por los coeficientes $D_{i,j}$ del operador frontera $\partial$, en otras palabras la $j$-ésima columna $D[:,j]$ representa los coeficientes de $\partial\sigma_j$. El órden de los simplices está dado por la filtración, de manera que las caras siempre preceden a las co-caras si su valor de filtración es el mismo. Luego, al implementar en Python los métodos *pHcol* y *pHrow* se observó que *pHrow* es una mejora sustancial en comparación a *pHcol*, debido a que no debe procesar todos los simplices sino que puede "ver al futuro" en el sentido de que puede ver qué columnas tienen el valor no nulo con indice más alto; al ver las filas en lugar de las columnas, toma en cierto sentido un poco de información de la cohomología persistente para no tener que procesar múltiples veces la misma fila. Véase [15] para el pseudocódigo de los métodos *pHcol* y *pHrow*.

Pero, evidentemente estos algoritmos tienen un grave problema, y es que ésta matriz $D[i,j]$ es cuadrada con tamaño igual al número de simplices en el complejo simplicial filtrado, el cual crece de manera combinatórica, complicando ampliamente el cómputo de la homología persistente. Debido a esto, y a que la cohomología persistente y la homología persistente tienen el mismo diagrama de barras, usaré las librerías *GUDHI* [17] dado a la facilidad para trabajar con complejos simpliciales y *Giotto-TDA* [16] ya que ésta es una librería altamente eficiente para acelerar (y simplificar) gran parte del algoritmo presentado por [9], además que ambas librerías están basadas en cohomología persistente, la cual es muchísimo más eficiente computacionalmente (Véase:[11, 15]).

## 3. MÉTODO TDABC

### 3.1. **Descripción del algoritmo.**

De manera intuitiva, el método consiste en construir un complejo simplicial sobre el conjunto total de puntos, para posteriormente elegir un subcomplejo simplicial apropiado guiado por la homología persistente. Esto se hace seleccionando un intervalo de persistencia adecuado, haciendo uso de las tres funciones de elección que se definirán posteriormente. Así podemos obtener un subcomplejo simplicial cuyo valor de filtración máximo es igual a el valor de nacimiento del intervalo seleccionado, de manera que se pueda asegurar que las relaciones entre los datos sobre el subcomplejo elegido son relevantes. Esto nos ayuda a

maximizar el número de simplices útiles (que contienen más elementos del conjunto de entrenamiento que del conjunto de prueba) y así poder asegurar una elección correcta de etiqueta para el punto a clasificar.

Las siguientes definiciones y el algoritmo fueron basados en [9].

**Definición 3.1.** Sea $\sigma$ un $k-$simplex y $\tau$ un $p$-simplex con $p \leq k$, decimos que $\tau$ es una ***p-cara*** de $\sigma$ si $\mathcal{V}(\tau) \subseteq \mathcal{V}(\sigma)$. Donde $\mathcal{V}(\sigma)$ denota el ***conjunto de vértices*** de $\sigma$, y lo denotamos por $\tau \leq \sigma$. Además, decimos que $\sigma$ es una ***co-cara*** de $\tau$.

**Definición 3.2.** Sea $\mathcal{K}$ un ***complejo simplicial*** y $\sigma \in \mathcal{K}$ un $k-$simplex. Definimos la estrella de $\sigma$ como

$$St_{\mathcal{K}}(\sigma) = \{\tau \in \mathcal{K} \mid \sigma \leq \tau\}$$

Sea $K \subset \mathcal{K}$. Definimos la ***cerradura*** de $K$ como el complejo simplicial más pequeño que contiene a $K$.

$$Cl_{\mathcal{K}}(K) = \{\mu \in \mathcal{K} \mid \mu \leq \sigma \text{ para algún } \sigma \in K\}$$

Es claro que $St_{\mathcal{K}}(\sigma)$ no es un complejo simplicial, por esto, el complejo simplicial más pequeño que contiene a $St_{\mathcal{K}}(\sigma)$ es la estrella cerrada

$$\overline{St_{\mathcal{K}}}(\sigma) = Cl_{\mathcal{K}}(St_{\mathcal{K}}(\sigma))$$

Finalmente, el ***link*** de $\sigma$ es el conjunto de simplices en la estrella cerrada que no comparten ninguna cara con $\sigma$, es decir

$$Lk_{\mathcal{K}}(\sigma) = \{\tau \in \overline{St_{\mathcal{K}}}(\sigma) \mid \tau \cap \sigma = \emptyset\}$$

**Lema 3.3.** *Sea $\mathcal{K}$ un complejo simplicial y $\sigma \in \mathcal{K}$, entonces se cumple*

$$Lk_{\mathcal{K}}(\sigma) = \overline{St_{\mathcal{K}}}(\sigma) \backslash (St_{\mathcal{K}}(\sigma) \cup Cl_{\mathcal{K}}(\sigma))$$

$$Lk_{\mathcal{K}}(\sigma) = \bigcup_{\mu \in St_{\mathcal{K}}(\sigma)} \{[\mathcal{V}(\mu) \backslash \mathcal{V}(\sigma)]\}$$

**Definición 3.4.** Sea $\mathcal{K}$ un complejo simplicial filtrado, $F$ una filtración en $\mathcal{K}$ y $\mathcal{E}_{\mathcal{K}} = \{\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_n\} \subseteq \mathbb{R}$ un conjunto de números no negativos tal que $\varepsilon_j < \varepsilon_i$ siempre que $j < i$ y donde $\varepsilon_i$ es un valor de parámetro en la construcción de el sub-complejo $\mathcal{K}_i \subseteq \mathcal{K}$ en $F$. Entonces, decimos que $\mathcal{E}_{\mathcal{K}}$ es la ***colección de valores de filtración*** asociada a $F$.

**Definición 3.5.** Sea $\mathcal{K}$ un complejo simplicial finito y $F$ una filtración en $\mathcal{K}$. Definimos la función de nivel de filtración como

$$\psi_F : \mathcal{E}_{\mathcal{K}} \to F$$

dada por

$$\psi_F(\varepsilon_i) = \mathcal{K}_i$$

**Definición 3.6.** Sea $\mathcal{K}$ un complejo simplicial filtrado y $\mathcal{E}_{\mathcal{K}}$ su colección de valores de filtración. Sea $\sigma \in \mathcal{K}$ un $k-$simplex, definimos su ***valor de filtración*** como $\xi_{\mathcal{K}}(\sigma) = \varepsilon_j$ si $\sigma \in \mathcal{K}_j$ y $\sigma \notin \mathcal{K}_i, \forall i < j$. Es claro que si $\tau \leq \sigma$ tenemos que $\xi_{\mathcal{K}}(\tau) \leq \xi_{\mathcal{K}}(\sigma)$ por lo que, como mencioné antes, las caras preceden a las co-caras.

Sea $P \subseteq R^n$ nuestro conjunto de puntos en el que para cada coordenada $p_i$ de un punto $p = (p_1, p_2, \ldots, p_n)$, tenemos que $p_i$ representa una característica de el dataset, además, $P = P_{test} \cup P_{train}$ donde $P_{train}$ es el conjunto de entrenamiento y $P_{test}$ es el conjunto de prueba. Sea $L = \{l_1, l_2, \ldots, l_N\}$ el conjunto de etiquetas o clases. Entonces, podemos construir un espacio de asociación $T$ dado por $P \times L \supseteq T = T_{train} \cup T_{test}$ en el que se le asocie una etiqueta en $L$ a cada punto en $P$, donde $T_{test}$ son las etiquetas reales del cada punto en el conjunto de prueba, luego, el propósito es predecir un conjunto de etiquetas $T_{pred} := \{(p, l) \mid p \in P_{test}, l \in L\}$ basado en $T_{train}$, para posteriormente poder evaluar la efectividad del algoritmo de clasificación al compararlo con las etiquetas reales $A_{test}$, dado que ambos representan asociaciones para cada punto en $P_{train}$ con su respectiva etiqueta en $L$.

Sea $A$ el $\mathbb{R}-$módulo con generadores $\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_N$ donde $\hat{l}_j$ corresponde a la etiqueta $l_j$. Dado que $\mathbb{R}$ es un campo, ésto será visto como un campo vectorial sobre $\mathbb{R}$, donde para cada $A \ni a = \sum_{j=1}^{N} a_j \hat{l}_j$, tenemos un vector $(a_1, a_2, \ldots, a_N) \in \mathbb{R}^N$ con el propósito de facilitar el cómputo.

**Definición 3.7.** Con el propósito de darle mayor valor a las relaciones entre los puntos que a los puntos (ya que éstos tienen valor de filtración cero), y a su vez, darle mayor relevancia a aquellas relaciones con los vértices cercanos al punto a clasificar, se define la siguiente función de relevancia.

$$\Xi_{\mathcal{K}}(\sigma) := \begin{cases} 1 & \text{, si } \sigma \text{ es un 0-simplex.} \\ \frac{1}{\xi_{\mathcal{K}}(\sigma)} & \text{, de cualquier otro modo.} \end{cases}$$

**Definición 3.8.** Definimos la función de asociación como $\Phi_i : \mathcal{K}_i \to A$

$$\Phi_i(\sigma) := \begin{cases} \hat{l}_j & \text{, si } \sigma \text{ es un 0-simplex.y } (\sigma, l_j) \in T_{train} \\ 0 & \text{, si } \sigma \text{ es un 0-simplex y } \forall j, \nexists (\sigma, l_j) \in T_{train} \\ \sum_{v \in \mathcal{V}(\sigma)} \Phi_i(v) & \text{, de cualquier otro modo.} \end{cases}$$

**Definición 3.9.** Definimos la función de extensión $\Psi_i : P_{test} \to A$ como

$$\Psi_i(p) = \sum_{\sigma \in Lk_{\mathcal{K}_i}([p])} \Xi_{\mathcal{K}}(\sigma) \cdot \Phi_i(\sigma)$$

**Definición 3.10.** Sea $v$ un punto en $T_{test}$ tal que $A \ni \Psi_i = \sum_{j=1}^{N} a_j \cdot \hat{l}_j$, y

$$\mathbb{R} \ni \overline{a} = \text{máx}\{a_j\}_{j=1}^{N}$$

, definimos la función de asignación de etiqueta como

$$\Lambda_i(v) = l_k$$

donde $k$ es seleccionado aleatoriamente de $\{j \mid a_j = \overline{a}\}$ en caso de empate; en la mayoría de los casos este conjunto contiene sólamente un elemento gracias a $\Xi_{\mathcal{K}}$.

Finalmente, el algoritmo para clasificación de datos fue implementado en python haciendo uso de las librerías *GUDHI* y *Giotto-TDA* [16, 17] y consiste en, dados $P_{train}, P_{test}, T_{train}$ no vacíos y una dimensión maximal $2 \leq q$:

- Obtener un conjunto total de puntos $P = P_{train} \cup P_{test}$
- Construir un complejo de Vietoris-Rips maximal $\mathcal{K}$ con vértices en $P$ usando *Giotto-TDA* y el método de colapso de lados de *GUDHI* [13].
- Computar la homología persistente hasta el $q$-ésimo grupo de homología.
- Obtener el conjunto de intervalos de persistencia $D = \{D^0, D^1, \ldots, D^q\}$ donde cada $D^i$ es el conjunto de intervalos de persistencia en el $i$-ésimo grupo de homología.
- Obtener el conjunto de intervalos de persitencia $D^k$ tal que $0 \leq k \leq q$ es la mayor dimensión para la que $D^k \neq \emptyset$.
- Se le asigna un valor de muerte igual a máx$(\mathcal{E}_{\mathcal{K}})$ a todo intervalo en $d \in D^k$, tal que $d[death] = +\infty$.
- Obtener un intervalo de persistencia deseado $d \in D^k$ por una de las siguientes funciones de selección, donde $life(d) = d[death] - d[birth]$
  - El intervalo de máxima persistencia:
  $$d_m = \arg\max_{d \in D^k}(life(d))$$
  - Un intervalo aleatorio:
  $$d_r = random(D^k)$$

– El intervalo más cercano a la persistencia promedio de los intervalos en $D^k$:

$$d_a = \arg\min_{d \in D^k} |life(d) - \mu(D^k)|$$

donde $\mu(D) = \frac{\sum_{d \in D} life(d)}{|D|}$.

- Obtener un subcomplejo simplicial $\mathcal{K}'$ dado por $d[birth]$,

$$\mathcal{K}' = \psi_F(d[birth]) = VR(P, d[birth])$$

- Generar un conjunto de etiquetas predecidas en $P_{test}$, $T_{pred}$ dado por

$$T_{pred} = \Lambda_i(P_{test})$$

- **Regresar** $T_{pred}$.

La implementación en Python del método TDABC puede encontrarse en mi repositorio: https://github.com/CamiloZamora21/TDABC. Todas las pruebas e implementaciones realizadas en este estudio fueron ejecutadas en un computador con procesador Ryzen 5 2400g @ 4ghz, 16GB de RAM y usando la versión 21H1 de Windows 10.

## 4. Resultados

Dado que la elección de un subcomplejo apropiado es crucial para el algoritmo, se evaluará la efectividad del método para cada una de las funciones de elección.

- TDABC-R, si se escoge un intervalo aleatoriamente.
- TDABC-A, si se escoge el intervalo con tiempo de vida más cercano al promedio.
- TDABC-M, si se escoge el intervalo con tiempo de vida máximo.

Tomaré como punto de referencia el método de $k$ vecinos más cercanos (k-NN) y la variante con distancia ponderada (wk-NN) de scikit-learn [12].

### 4.1. **Datasets.**

Para evaluar la efectividad del método se seleccionaron las siguientes datasets:

- Dataset “Pima Indians Diabetes”, tomado de [8].
  Cada muestra en este dataset consiste de 8 atributos numéricos de los pacientes, como el número de embarazos, la edad, su concentración plasmática de glucosa, etc. Y un atributo numérico de clase; 1 si la paciente es diabética, de lo contrario, 0. el dataset cuenta con 268 muestras positivas entre un total de 768 muestras, y tiene con datos faltantes.
- Dataset “Iris”, tomado de [4].
  Cada muestra consiste de 4 atributos numéricos que corresponden a las medidas de sépalo y pétalo de tres especies distintas de flor Iris; que constituyen las tres clases, “Setosa”, “Virginica ” y “Versicolor”. el dataset cuenta con 50 muestras por clase para un total de 150 muestras.
- Dataset “Sonar”, tomada de [4]. Cada muestra consta de 60 atributos numéricos que corresponden a la energía en una banda de frecuencias específica en un periodo de tiempo; y un atributo de clase “R” ó “M”, donde “R” es una roca y “M” es una “mina”; un cilindro metálico. el dataset cuenta con 111 muestras de “minas” y 97 muestras de rocas.
- Dataset “seeds”, tomado de [4].
  Cada muestra consta de 7 atributos numéricos que corresponden a mediciones de las semillas de diferentes variedades de trigo, y un atributo de clase, con tres clases distintas que corresponden a las variedades “Kama”, “Rosa” y “Canadian” de trigo, con 70 muestras cada una.

- Dataset "banknote authentication", tomado de [4].
  Cada muestra cuenta con 4 atributos numéricos que corresponden a datos tomados de imágenes de elementos similares a billetes, y un atributo de clase, donde 0 es auténtico y 1 es inauténtico.
- Dataset "Ionosphere", tomado de [4].
  Cada muestra consta de 34 atributos numéricos que corresponden a datos tomados por un radar de la ionósfera, y un atributo de clase con valores "g" y "b" donde "g" representa una buena muestra y "b" una mala muestra.

### 4.2. **Comparación con otros clasificadores.**

Todos los clasificadores serán evaluados usando el método de validación cruzada Repeated Stratified $K$-Fold de scikit-learn con $K = 10$ y 5 repeticiones. Éste método de validación cruzada consiste básicamente en dividir el dataset en $K$ subconjuntos de la manera más uniforme posible, y manteniendo la proporción de clases en todos los subconjuntos lo más cercana posible a la proporción de clases en el dataset, iterar el clasificador por cada subconjunto de manera que los $K - 1$ conjuntos restantes se usen como conjunto de entrenamiento, y repetirlo un número deseado de veces. No se hará preprocesado de datos de ningún tipo. El método TDABC será evaluado para cada una de sus variantes, y será ejecutado con $q \in [3, 9] \subseteq \mathbb{N}$, se mostrarán resultados para el valor de $q$ que muestre mejores resultados en cada dataset. Los métodos de $k$ vecinos más cercanos serán ejecutados para $k \in [3, 39] \subseteq \mathbb{N}$ y se mostrarán los resultados para el valor de $k$ que muestre mejores resultados en cada dataset.

En el dataset *Iris* se escogió una dimensión $q = 4$. Para el dataset *banknote authentication* se escogió una dimensión de $q = 3$. Para el dataset *Seeds* se escogió una dimensión de $q = 5$. Para el dataset *Sonar* se escogió una dimensión de $q = 4$. Para el dataset *Pima Indians Diabetes* se escogió una dimensión de $q = 6$. Para el dataset *Ionosphere* se escogió una dimensión de $q = 4$.

### 4.3. **Métricas.**

Las métricas fueron tomadas de las matrices de confusión, usando los valores de verdaderos positivos, $TP$, verdaderos negativos, $TN$, falsos negativos, $FN$, y falsos positivos, $FP$.

Por ejemplo, veamos la siguiente matriz de confusión en Figura 13. Para esta matriz de confusión, tenemos que un 70 % de los negativos fueron correctamente clasificados como negativos, $TN = 0{,}7 =$; un 30 % de los negativos fueron identificados como positivos, por lo que $FP = 0{,}3$; un 60 % de los positivos fueron identificados como negativos, por lo que $FN = 0{,}6$ y un 40 % de los positivos fueron correctamente identificados como positivos, por lo que $TP = 0{,}4$.

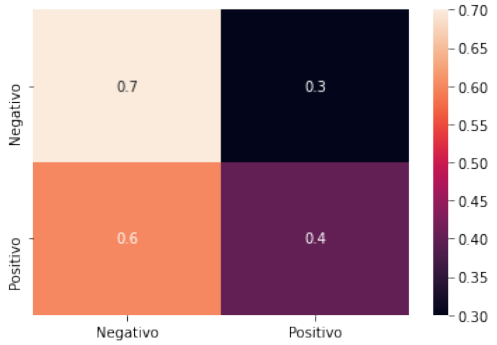



FIGURA 13. Ejemplo de matriz de confusión

Para los problemas de multiclase, éstas métricas fueron tomadas para cada etiqueta, donde para cada etiqueta, los verdaderos positivos se encuentran en la celda de la diagonal que corresponde a esa etiqueta, los falsos negativos son la suma de las celdas de la fila que corresponde a la etiqueta que no están en la diagonal, los falsos positivos son la suma de las celdas que se encuentran en la columna correspondiente a la etiqueta y no en la diagonal, y los verdaderos negativos son la suma de las celdas que no se encuentran ni en la fila ni en la columna correspondientes a la etiqueta.

Las métricas fueron promediadas usando la media aritmética (macro). Sea $l \in L$

- Exactitud. Representa la proporción de predicciones correctas en cada etiqueta.

$$ACC_l = \frac{TP_l + TN_l}{TP_l + TN_l + FP_l + FN_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 1 Peor valor: 0.
- Precisión o valor predictivo positivo. Representa la tasa de predicciones positivas correctas en el total de predicciones positivas para cada etiqueta.

$$PPV_l = \frac{TP_l}{TP_l + FP_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 1 Peor valor: 0.
- Valor predictivo negativo. Representa la tasa de predicciones negativas correctas en el total de predicciones negativas para cada etiqueta.

$$NPV_l = \frac{TN_l}{TN_l + FN_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 1 Peor valor: 0.
- Sensibilidad. Representa la tasa de predicciones positivas correctas en el total de verdaderos positivos para cada etiqueta.

$$TPR_l = \frac{TP_l}{TP_l + FN_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor :1 Peor valor: 0.
- Especificidad. Representa la tasa de predicciones negativas correctas en el total de predicciones negativas para cada etiqueta.

$$TNR_l = \frac{TN_l}{TN_l + FP_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor :1 Peor valor: 0.
- Tasa de falsos descubrimientos. Representa la tasa de falsos positivos en el total de predicciones positivas para cada etiqueta.

$$FDR_l = \frac{FP_l}{FP_l + TP_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 0 Peor valor: 1.
- Tasa de falsos positivos. Representa la tasa de predicciones positivas incorrectas en el total de negativos reales para cada etiqueta.

$$FPR_l = \frac{FP_l}{FP_l + TN_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 0 Peor valor: 1.

- Tasa de falsos negativos. Representa el número de predicciones negativas incorrectas en el total de positivos reales para cada etiqueta.

$$FNR_l = \frac{FN_l}{TP_l + FN_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 0 Peor valor: 1.
- Puntaje F1. Representa la media armónica entre la precisión y la sensibilidad.

$$F1_l = 2\frac{PPV_l * TPR_l}{PPV_l + TPR_l}$$

  Rango de valores: $[0, 1] \in \mathbb{R}$ Mejor valor : 1 Peor valor: 0.

Veamos las comparaciones de las métricas y las matrices de confusión en los distintos datasets.

### 4.4. **Matrices de confusión.**

Las matrices de confusión fueron normalizadas para facilitar el análisis de las mismas. Es decir, cada celda representa el porcentaje de elementos cuya etiqueta real corresponde a la fila de la matriz que se les asignó la etiqueta correspondiente a la columna.

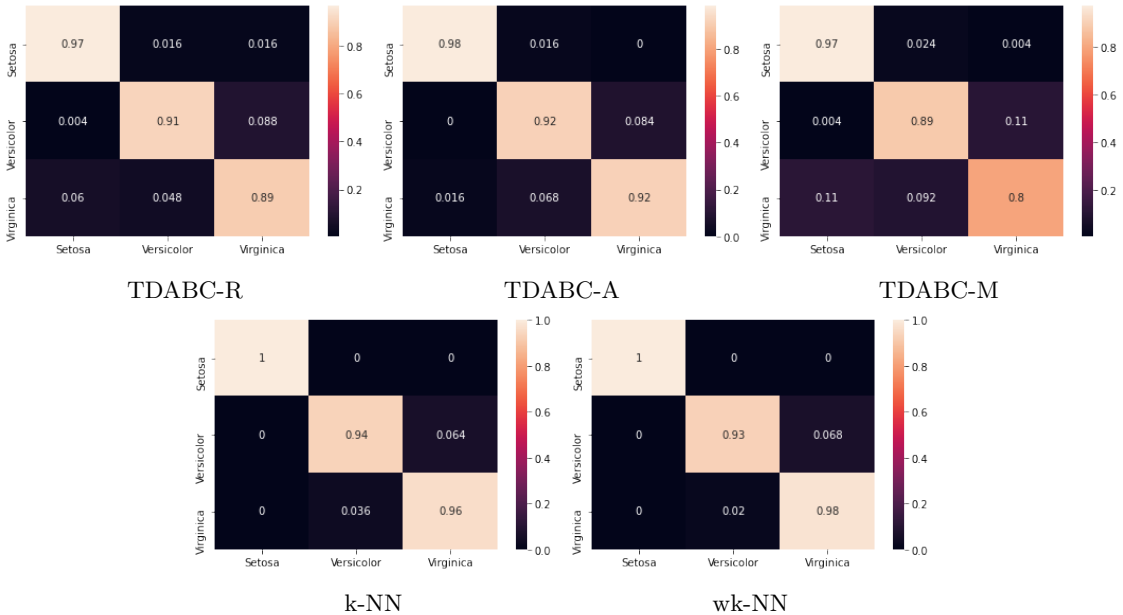



FIGURA 14. Matrices de confusión en el dataset *Iris* con $q = 4$

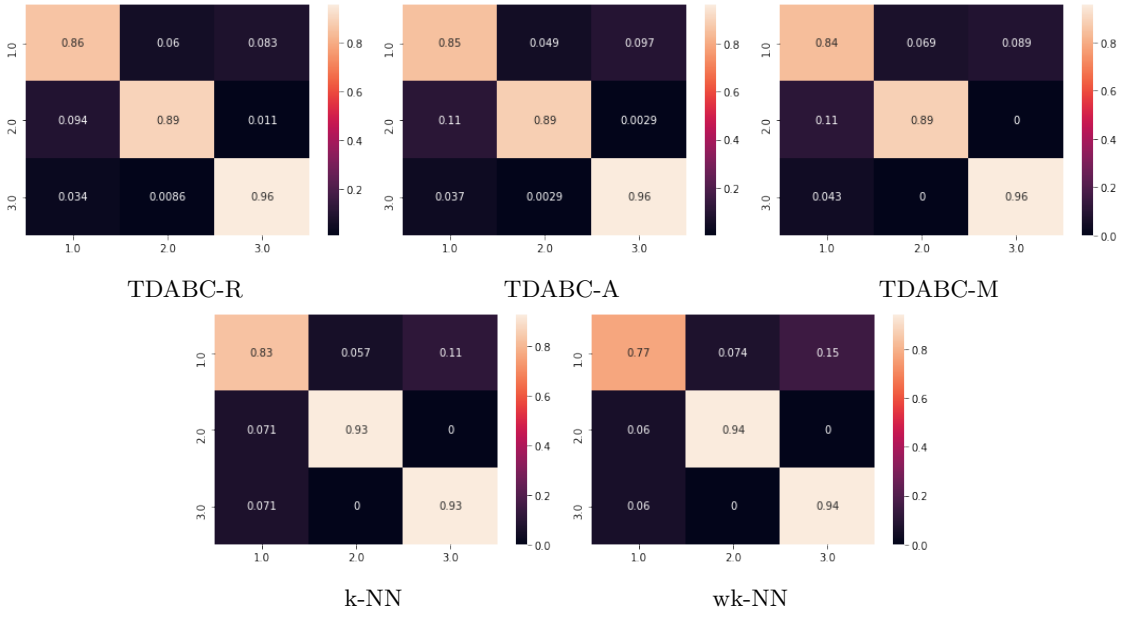



FIGURA 15. Matrices de confusión en el dataset *Seeds* con $q = 5$

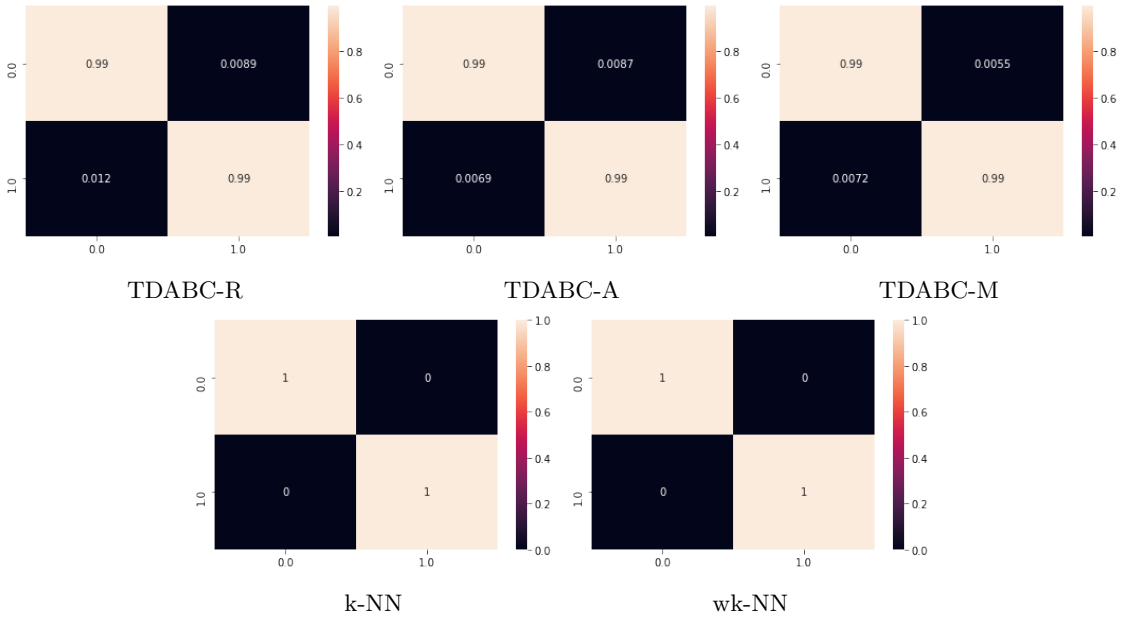



FIGURA 16. Matrices de confusión en el dataset *banknote authentication* con $q = 3$

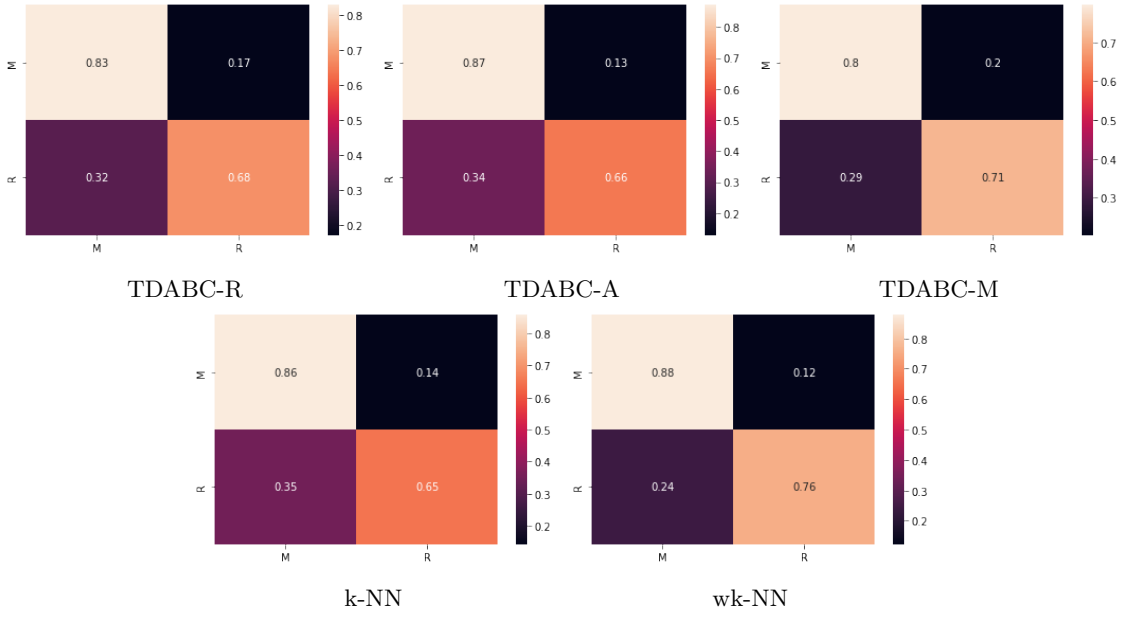



FIGURA 17. Matrices de confusión en el dataset *Sonar* con $q = 4$

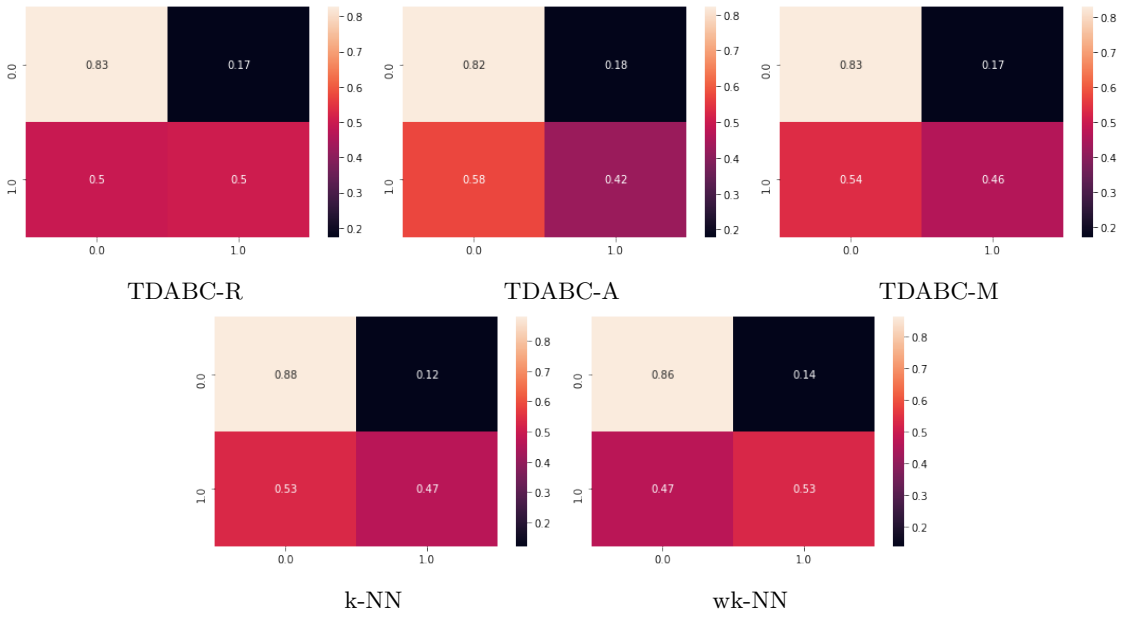



FIGURA 18. Matrices de confusión en el dataset *Pima Indians Diabetes* con $q = 6$

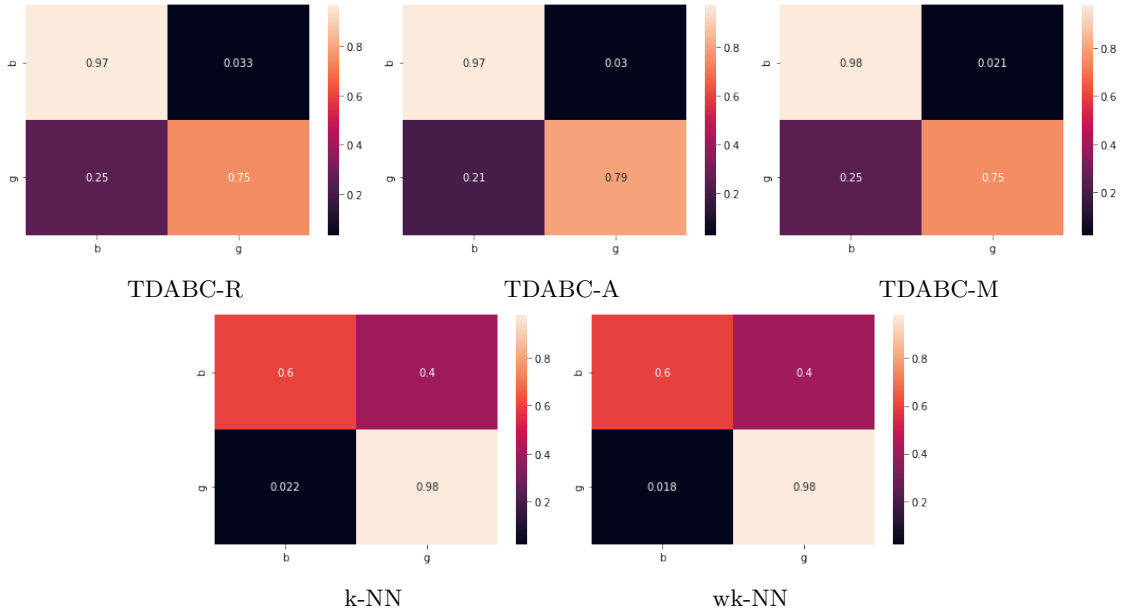



FIGURA 19. Matrices de confusión en el dataset *Ionosphere* con $q = 4$

## 4.5. **Comparación de métricas.**

| ACC | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.992128 | 0.683073 | **0.857550** | 0.959111 | 0.933968 | 0.774038 |
| TDABC-M | 0.993732 | 0.701302 | 0.833048 | 0.923556 | 0.930794 | 0.757692 |
| TDABC-R | 0.989650 | **0.714583** | 0.825641 | 0.948444 | **0.935238** | 0.758654 |
| k-NN | **1.000000** | 0.672436 | 0.789683 | 0.977778 | 0.930159 | 0.753571 |
| wk-NN | **1.000000** | 0.695152 | 0.788730 | **0.980444** | 0.923175 | **0.818120** |

| PPV | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.991847 | 0.643835 | **0.852421** | 0.938667 | 0.901745 | 0.782896 |
| TDABC-M | 0.993670 | 0.667192 | 0.836348 | 0.884832 | 0.896310 | 0.757285 |
| TDABC-R | 0.989569 | 0.683531 | 0.828409 | 0.922574 | **0.902892** | 0.761348 |
| k-NN | **1.000000** | 0.708006 | 0.837436 | 0.966902 | 0.895127 | 0.765058 |
| wk-NN | **1.000000** | **0.719343** | 0.839585 | **0.971369** | 0.884611 | **0.822668** |

| NPV | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.991847 | 0.643835 | **0.852421** | 0.969333 | 0.950811 | 0.782896 |
| TDABC-M | 0.993670 | 0.667192 | 0.836348 | 0.943910 | 0.948355 | 0.757285 |
| TDABC-R | 0.989569 | 0.683531 | 0.828409 | 0.961494 | **0.951730** | 0.761348 |
| k-NN | **1.000000** | 0.708006 | 0.837436 | 0.983395 | 0.947845 | 0.765058 |
| wk-NN | **1.000000** | **0.719343** | 0.839585 | **0.985517** | 0.943719 | **0.822668** |

| TPR | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.992227 | 0.623113 | **0.882254** | 0.938667 | 0.900952 | 0.766964 |
| TDABC-M | 0.993638 | 0.645770 | 0.865238 | 0.885333 | 0.896190 | 0.754899 |
| TDABC-R | 0.989472 | 0.665839 | 0.856667 | 0.922667 | **0.902857** | 0.753590 |
| k-NN | **1.000000** | 0.672436 | 0.789683 | 0.966667 | 0.895238 | 0.753571 |
| wk-NN | **1.000000** | **0.695152** | 0.788730 | **0.970667** | 0.884762 | **0.818120** |

| TNR | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.992227 | 0.623113 | **0.882254** | 0.969333 | 0.950476 | 0.766964 |
| TDABC-M | 0.993638 | 0.645770 | 0.865238 | 0.942667 | 0.948095 | 0.754899 |
| TDABC-R | 0.989472 | 0.665839 | 0.856667 | 0.961333 | **0.951429** | 0.753590 |
| k-NN | **1.000000** | 0.672436 | 0.789683 | 0.983333 | 0.947619 | 0.753571 |
| wk-NN | **1.000000** | **0.695152** | 0.788730 | **0.985333** | 0.942381 | **0.818120** |

| FDR | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.008153 | 0.356165 | **0.147579** | 0.061333 | 0.098255 | 0.217104 |
| TDABC-M | 0.006330 | 0.332808 | 0.163652 | 0.115168 | 0.103690 | 0.242715 |
| TDABC-R | 0.010431 | 0.316469 | 0.171591 | 0.077426 | **0.097108** | 0.238652 |
| k-NN | **0.000000** | 0.291994 | 0.162564 | 0.033098 | 0.104873 | 0.234942 |
| wk-NN | **0.000000** | **0.280657** | 0.160415 | **0.028631** | 0.115389 | **0.177332** |

| FPR | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.007773 | 0.376887 | **0.117746** | 0.030667 | 0.049524 | 0.233036 |
| TDABC-M | 0.006362 | 0.354230 | 0.134762 | 0.057333 | 0.051905 | 0.245101 |
| TDABC-R | 0.010528 | 0.334161 | 0.143333 | 0.038667 | **0.048571** | 0.246410 |
| k-NN | **0.000000** | 0.327564 | 0.210317 | 0.016667 | 0.052381 | 0.246429 |
| wk-NN | **0.000000** | **0.304848** | 0.211270 | **0.014667** | 0.057619 | **0.181880** |

| FNR | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.007773 | 0.376887 | **0.117746** | 0.061333 | 0.099048 | 0.233036 |
| TDABC-M | 0.006362 | 0.354230 | 0.134762 | 0.114667 | 0.103810 | 0.245101 |
| TDABC-R | 0.010528 | 0.334161 | 0.143333 | 0.077333 | **0.097143** | 0.246410 |
| k-NN | **0.000000** | 0.327564 | 0.210317 | 0.033333 | 0.104762 | 0.246429 |
| wk-NN | **0.000000** | **0.304848** | 0.211270 | **0.029333** | 0.115238 | **0.181880** |

| F1 | | | | | | |
|---|---|---|---|---|---|---|
| Dataset | Banknote | Diabetes | Ionosphere | Iris | Seeds | Sonar |
| TDABC-A | 0.992034 | 0.627341 | **0.853747** | 0.938667 | 0.900793 | 0.768348 |
| TDABC-M | 0.993654 | 0.651229 | 0.830182 | 0.883881 | 0.895949 | 0.755610 |
| TDABC-R | 0.989521 | 0.671404 | 0.822567 | 0.922479 | **0.902555** | 0.754763 |
| k-NN | **1.000000** | 0.681360 | 0.810803 | 0.966660 | 0.894974 | 0.754698 |
| wk-NN | **1.000000** | **0.703366** | 0.810522 | **0.970650** | 0.883246 | **0.819756** |

### 4.6. **Conclusión.**

Acorde a lo esperado, el método funciona de manera efectiva en distintos tipos de datasets, tengan o no desbalance de datos. Sin embargo, es evidente que aún tiene problemas para predecir en datasets con datos faltantes como vimos anteriormente en el dataset *Pima Indians Diabetes* con lo que podemos concluir que el método funciona de manera efectiva para un amplio rango de problemas de clasificación independientemente del balance de datos, siempre y cuándo no haya una cantidad significativa de datos faltantes. Al compararlo con los métodos de $k$ vecinos más cercanos se observó que el método TDABC es competitivo en todos los casos y superior en los casos de los datasets *Ionosphere* y *Seeds*. No obstante, tanto la construcción de los complejos simpliciales como el cómputo de la homología persistente tienen una naturaleza combinatórica, mientras que para los métodos de $k$-vecinos más cercanos, las implementaciones en *scikit-learn* son increíblemente eficientes. Por lo que generalmente, al igual que para cualquier otro método de machine learning no es recomendable aplicar el método ciegamente, sino compararlo con otros métodos de clasificación y decidir si vale o no la pena el tiempo extra que toma aplicar el método TDABC en cada dataset como lo fue para los datasets *Ionosphere* y *Seeds* en mis pruebas. Sin duda, es un método prometedor, además de ser una aplicación novedosa de el análisis topológico de datos.

## Referencias


[1] A. Björner. "Topological Methods". En: *Handbook of Combinatorics (Vol. 2)*. Cambridge, MA, USA: MIT Press, 1996, págs. 1819-1872. ISBN: 0262071711 (vid. pág. 10).

[2] Gunnar Carlsson. "Topology and Data". En: *Bulletin of The American Mathematical Society - BULL AMER MATH SOC* 46 (abr. de 2009), págs. 255-308. DOI: 10.1090/S0273-0979-09-01249-X (vid. págs. 2, 17).

[3] Richard M. Foote David S. Dummit. *Abstract Algebra - 3rd Edition*. 3.ª ed. John Wiley y Sons, Inc., 2004. ISBN: 9780471433347,0471433349 (vid. pág. 17).

[4] Dheeru Dua y Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (vid. págs. 22, 23).

[5] Mustafa Elsheikh, Mark Giesbrecht, Andy Novocin y B. David Saunders. "Fast Computation of Smith Forms of Sparse Matrices Over Local Rings". En: *CoRR* abs/1201.5365 (2012). arXiv: 1201.5365. URL: http://arxiv.org/abs/1201.5365 (vid. pág. 10).

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke y Travis E. Oliphant. "Array programming with NumPy". En: *Nature* 585 (2020), págs. 357-362. DOI: 10.1038/s41586-020-2649-2.

[7] Allen Hatcher. *Algebraic topology*. 1.ª ed. Cambridge University Press, 2001. ISBN: 9780521795401,0521795400 (vid. pág. 10).

[8] *Kaggle*. https://www.kaggle.com (vid. pág. 22).

[9] Rolando Kindelan, José Frías, Mauricio Cerda y Nancy Hitschfeld. "Classification based on Topological Data Analysis". En: *CoRR* abs/2102.03709 (2021). arXiv: 2102.03709. URL: https://arxiv.org/abs/2102.03709 (vid. págs. 1, 19, 20).

[10] Ngoc-Khuyen Le, Philippe MARTINS, Laurent Decreusefond y Ana¨s Vergne. "Construction of the generalized Cech complex". En: *Vehicular Technology Conference*. VTC 2015. Glasgow, United Kingdom, mayo de 2015. URL: https://hal.archives-ouvertes.fr/hal-01069775 (vid. pág. 11).

[11] Clément Maria. "Persistent Cohomology". En: *GUDHI User and Reference Manual*. 3.4.1. GUDHI Editorial Board, 2021. URL: https://gudhi.inria.fr/doc/3.4.1/group__persistent__cohomology.html (vid. pág. 19).

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot y E. Duchesnay. "Scikit-learn: Machine Learning in Python". En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830 (vid. pág. 22).

[13] Siddharth Pritam. "Edge collapse". En: *GUDHI User and Reference Manual*. 3.4.1. GUDHI Editorial Board, 2021. URL: https://gudhi.inria.fr/doc/3.4.1/group__edge__collapse.html (vid. pág. 21).

[14] Joseph J. Rotman. *An Introduction to Algebraic Topology*. Graduate Texts in Mathematics. Springer, New York, NY, 1998. DOI: https://doi.org/10.1007/978-1-4612-4576-6 (vid. pág. 2).

[15] Vin de Silva, Dmitriy Morozov y Mikael Vejdemo-Johansson. "Dualities in persistent (co)homology". En: *Inverse Problems* 27.12 (2011), pág. 124003. ISSN: 1361-6420. DOI: 10.1088/0266-5611/27/12/124003. URL: http://dx.doi.org/10.1088/0266-5611/27/12/124003 (vid. pág. 19).

[16] Guillaume Tauzin, Umberto Lupo, Lewis Tunstall, Julian Burella Pérez, Matteo Caorsi, Wojciech Reise, Anibal Medina-Mardones, Alberto Dassatti y Kathryn Hess. *giotto-tda: A Topological Data Analysis*

*Toolkit for Machine Learning and Data Exploration*. 2021. arXiv: 2004.02551 [cs.LG] (vid. págs. 19, 21).

[17] The GUDHI Project. *GUDHI User and Reference Manual*. 3.4.1. GUDHI Editorial Board, 2021. URL: https://gudhi.inria.fr/doc/3.4.1/ (vid. págs. 19, 21).

[18] Afra Zomorodian. "Fast construction of the Vietoris-Rips complex". En: *Computers & Graphics* 34.3 (2010). Shape Modelling International (SMI) Conference 2010, págs. 263-271. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2010.03.007. URL: https://www.sciencedirect.com/science/article/pii/S0097849310000464 (vid. pág. 11).

[19] Afra Zomorodian. "Topological Data Analysis -I". En: 70 (2013). DOI: 10.1090/psapm/070/587 (vid. pág. 1).

[20] Afra Zomorodian y Gunnar Carlsson. "Computing Persistent Homology". En: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. Brooklyn, New York, USA: Association for Computing Machinery, 2004, págs. 347-356. ISBN: 1581138857. DOI: 10.1145/997817.997870. URL: https://doi.org/10.1145/997817.997870 (vid. págs. 2, 17).

*Email address*: crzamora@uninorte.edu.co