

**UNIVERSIDAD COMPLUTENSE DE MADRID**  
**FACULTAD DE CIENCIAS MATEMÁTICAS**  
**DPTO. DE ESTADÍSTICA E INVESTIGACIÓN OPERATIVA**



## Trabajo de Fin de Máster

**UN ESTUDIO COMPARATIVO DE PCA Y T-SNE  
EN TEXTOS PROCESADOS CON NLP**

Tutor: Daniel Vélez Serrano

**Carlos Rodrigo Pascual**

Master de Ingeniería Matemática

Curso 2023/2024



## Abstract

This project presents a comparison of different dimensionality reduction models applied to a dataset from a textual corpus. The aim is to contribute to the data science literature, especially in the area of textual data, which historically has not played a relevant role in the comparison of dimensionality reduction techniques.

The report is structured in several chapters. The first chapter presents a description of the study and the motivations behind it, together with an overview of the steps taken to achieve the objectives. The second chapter develops the theoretical framework, explaining in detail the Machine Learning algorithms used, their schemes, parameters and calibration techniques, highlighting their advantages and limitations, as well as the results evaluation methods and the integrative methodology used for the experiment. The third chapter details the steps of the experiment, based on the methodology and processes explained in the theoretical framework. Finally, the fourth chapter compiles the results, concluding how they enrich the existing literature in the field of dimensionality reduction and suggesting different alternatives for further study. A final chapter provides access to the code used.

## Objetivos

Este proyecto presenta una comparativa de diferentes modelos de reducción de dimensionalidad aplicados a un conjunto de datos provenientes de un corpus textual. El objetivo es contribuir a la literatura de la ciencia de datos, especialmente en el ámbito de los datos textuales, que históricamente no han desempeñado un papel relevante en la comparación de técnicas de reducción de dimensionalidad.

El informe está estructurado en varios capítulos. El primer capítulo presenta una descripción del estudio y las motivaciones que lo impulsaron, junto con una visión general de los pasos realizados para alcanzar los objetivos. En el segundo capítulo se desarrolla el marco teórico, explicando en detalle los algoritmos de *Machine Learning* utilizados, sus esquemas, parámetros y técnicas de calibración, resaltando sus ventajas y limitaciones, así como los métodos de evaluación de resultados y la metodología integradora empleada para el experimento. El tercer capítulo detalla los pasos del experimento, basándose en la metodología y los procesos explicados en el marco teórico. Finalmente, el cuarto capítulo recopila los resultados, concluyendo cómo estos enriquecen la literatura existente en el ámbito de la reducción de dimensionalidad y sugiriendo diferentes alternativas para profundizar en el estudio. Se incluye un último capítulo que proporciona acceso al código utilizado.



# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
<b>2</b>	<b>Marco teórico</b>	<b>2</b>
2.1	Técnicas de Machine Learning . . . . .	2
2.1.1	Minería de textos . . . . .	2
2.1.2	Reducción de dimensionalidad . . . . .	5
2.1.2.1	Análisis de Componentes Principales, PCA . . . . .	6
2.1.2.2	t-Distributed Stochastic Neighbor Embedding, t-SNE . . . . .	8
2.1.3	Clustering . . . . .	11
2.1.3.1	Algoritmos de clustering . . . . .	12
2.1.3.2	Evaluación de clasificaciones de datos . . . . .	17
2.1.4	Evaluación de modelos de clasificación . . . . .	19
2.2	Metodología para la clasificación de textos por temáticas . . . . .	23
<b>3</b>	<b>Caso práctico</b>	<b>25</b>
3.1	Construcción del corpus . . . . .	25
3.2	Procesado y vectorización . . . . .	27
3.3	Reducción de dimensionalidad . . . . .	28
3.4	Clasificación y evaluación . . . . .	29
3.5	Comparación . . . . .	38
<b>4</b>	<b>Conclusiones</b>	<b>39</b>
<b>5</b>	<b>Repositorio</b>	<b>40</b>



# 1. Introducción

El presente informe es fruto de un Trabajo de Fin de Máster del Máster de Ingeniería Matemática impartido en la Universidad Complutense de Madrid. El estudio que aquí se presenta tiene como objetivo principal realizar una comparativa exhaustiva entre los modelos de reducción de dimensionalidad PCA (*Principal Component Analysis*) y t-SNE (*t-Distributed Stochastic Neighbor Embedding*) aplicados a un conjunto de datos derivados de textos procesados mediante técnicas de NLP (*Natural Language Processing*). Tradicionalmente, la comparación entre PCA y t-SNE se ha realizado utilizando conjuntos de datos procedentes de imágenes o de extracciones médicas, biológicas y genéticas. El enfoque que se presenta aquí trata de contribuir a la literatura existente sobre reducción de dimensionalidad de datos textuales midiendo la eficacia de dos técnicas que ya han sido enfrentadas en otros contextos.

La motivación principal que ha llevado a la realización de este trabajo radica en la necesidad de explorar y comprender cómo se comportan los modelos de reducción de dimensionalidad, en este caso PCA y t-SNE, cuando se aplican a datos textuales, en contraste con otros tipos de datos que han sido el foco predominante de investigaciones anteriores ([1], [2], [3], etc.). A lo largo de los últimos años en la ciencia de datos, la comparación entre PCA y t-SNE se ha centrado en conjuntos de datos visuales, especialmente el conjunto de datos MNIST, debido a su simplicidad y a la facilidad con la que se pueden interpretar los resultados visuales. Sin embargo, el procesamiento y análisis de datos textuales presenta desafíos únicos y diferentes en comparación con los datos visuales. Los textos tienen una estructura y características propias que pueden influir significativamente en los resultados de la reducción de dimensionalidad. Por tanto, surge la necesidad de investigar cómo estos modelos se comportan en este contexto, con el objetivo de proporcionar una comprensión más profunda y mejorar las prácticas de análisis de datos textuales en la ciencia de datos. Además, con el creciente volumen de datos textuales generados diariamente a través de diversas plataformas y medios, es crucial desarrollar y evaluar herramientas eficaces para el análisis de estos datos. La reducción de dimensionalidad es una técnica fundamental en este campo, y esta investigación pretende contribuir a la identificación de los métodos más adecuados para aplicaciones de NLP.

Para llevar a cabo esta comparativa, primero se aplicarán técnicas de NLP a un conjunto de datos textuales para transformar los textos en representaciones vectoriales. Posteriormente, se utilizarán los modelos PCA y t-SNE para reducir la dimensionalidad de estos vectores, permitiendo visualizar y analizar las diferencias en la representación de los datos. Los resultados serán evaluados y comparados mediante diversas metodologías de *Machine Learning*, obteniendo métricas y visualizaciones para destacar las ventajas y desventajas de cada método en el contexto de datos textuales.

## 2. Marco teórico

En este capítulo se desarrollan los fundamentos teóricos de las técnicas, algoritmos y metodologías que son usados a lo largo del estudio.

El capítulo está dividido en dos partes: una primera parte en la que se explican diferentes ramas de *Machine Learning* y se detallan las técnicas englobadas en dichas ramas que servirán para alcanzar los objetivos del trabajo; una segunda parte en la que se explica la integración de dichas técnicas en una metodología para clasificar un conjunto de textos por temáticas.

### 2.1. Técnicas de Machine Learning

El *Machine Learning*, o aprendizaje automático, es una rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender y tomar decisiones basadas en datos, sin ser programadas explícitamente para cada tarea específica. En lugar de seguir instrucciones paso a paso, el aprendizaje automático permite a las máquinas aprender mediante ejemplos y experiencias, y mejorar su rendimiento a medida que se les proporciona más información. Esto se logra mediante la construcción y entrenamiento de modelos, utilizando algoritmos que pueden reconocer patrones, hacer predicciones y/o tomar decisiones basadas en esos patrones.

A continuación se presentan diferentes ramas del *Machine Learning* y se desarrollan algunas técnicas o algoritmos englobados en dichas ramas. Se realiza un recorrido por el procesamiento y la estructuración de conjuntos de textos, se continua explicando diferentes técnicas de reducción de dimensionalidad, que constituyen el objeto de estudio del proyecto, luego se presentan varios algoritmos de *clustering* y finalmente se presentarán algunas métricas para evaluar modelos de *Machine Learning*.

#### 2.1.1. Minería de textos

La minería de textos engloba todos los procesos mediante los cuales se convierte un corpus de textos en un conjunto de datos con estructura. Esta estructuración de los textos es imprescindible a la hora de analizar un corpus, pues los algoritmos de analítica avanzada, por sí solos, no son capaces de leer textos, salvo que estos estén estructurados como datos. Tanto si se quiere segmentar el corpus por temáticas o por entidades relevantes, como si se quiere entrenar modelos generativos como traductores o *chatbots*, los textos han de pasar por un primer proceso de estructuración.

En este proyecto, el proceso de estructuración de textos que se va a utilizar consta de dos grandes fases: una primera etapa de procesamiento de los textos en la que se filtrarán y normalizarán las palabras de los textos; y una segunda en la que se construirá una matriz de documentos - términos en la que se representará de forma fiable el contenido de los textos.

El objetivo del procesar un corpus de textos es múltiple: seleccionar la información relevante, homogeneizar el formato de dicha información, aglomerar expresiones con significados parecidos en un mismo representante, corregir erratas, etc.. Estos objetivos están muy ligados al entendimiento humano del lenguaje, lo que hace que no sea trivial abordarlos. A continuación se explican algunas de las fases más comunes en los procesados de corpus de textos que tratan de darles solución:



- **Detección del idioma:** Es el proceso por el cuál se le asigna (manualmente o de forma automatizada) el idioma a cada texto. El objetivo es decidir si filtrar los textos por un idioma específico o si dar un tratamiento específico a los textos de cada idioma.
- **Corrección ortográfica:** Consiste en procesar los textos por correctores ortográficos para eliminar faltas de ortografía.
- **Normalización:** Consiste en transformar el texto a una forma estándar o normalizada. Esto puede incluir la eliminación de caracteres especiales, conversión a minúsculas, eliminación de puntuación y otros pasos para que el texto sea más uniforme y fácil de procesar.
- **Tokenización:** Es el proceso de dividir el texto en unidades más pequeñas llamadas *tokens*. Los *tokens* pueden ser palabras individuales o *n*-gramas (grupos ordenados de *n* palabras). Tras dividir el texto en palabras, si se quieren considerar *n*-gramas, se procesan los textos divididos en palabras por un modelo que evalúe la frecuencia de todas las tuplas de palabras que aparecen, extrayendo como *n*-gramas aquellas *n*-uplas cuya frecuencia de aparición es significativamente mayor que la media. Este proceso es útil cuando en el texto aparecen expresiones de jerga, nombres y apellidos, nombres de marcas y entidades, etc..
- **Eliminación de *stopwords*:** Las denominadas *stopwords* son palabras comunes y no informativas, como preposiciones, conjunciones, determinantes, etc.. Se eliminan del texto ya que no aportan mucho significado, consiguiendo así reducir el ruido y el tamaño del vocabulario.
- **Reconocimiento de entidades relevantes:** Consiste en identificar y clasificar entidades nombradas, como nombres de personas, organizaciones, ubicaciones, fechas, etc., en un texto.
- **Etiquetado POS (*Parts of speech*):** El etiquetado gramatical o de partes del discurso es el proceso de asignar una etiqueta gramatical (sustantivo, verbo, adjetivo, etc.) a cada palabra en un texto. Ayuda a comprender la estructura y el significado de las oraciones.
- **Lematización:** La lematización reduce las palabras a su forma base, teniendo en cuenta el contexto y la morfología del idioma. El resultado es un lema, que es la forma canónica o el diccionario de una palabra. Se nutre del etiquetado POS para tratar de forma diferente los diferentes tipos de palabras (simplificar las conjugaciones verbales a sus infinitivos, normalizar al masculino - singular los sustantivos comunes y los adjetivos, etc.).
- ***Stemming*:** Es el proceso de reducir las palabras a su raíz o base, eliminando los sufijos. El objetivo es reducir las palabras a una forma común para tratar las variantes léxicas de una palabra como la misma entidad.

Los pasos anteriores son los más habituales en lo procesados de textos, sin embargo, cada problema requiere un estudio de cómo abordarlo y una calibración de los procesos, eligiéndose con cuidado qué procesos se realizan, en qué orden y con qué modelos.

A modo de ejemplo, se muestran en la tabla 2.1 varios textos y sus versiones al atravesar varias fases del procesado. En la primera columna se muestran los textos originales. En la segunda, su resultado tras un preprocesado que incluye normalización, *tokenización* (sin considerar *n*-gramas) y eliminación de *stopwords*. En la tercera, el resultado de hacer una lematización a los textos preprocesados de la segunda columna. Y en la cuarta, el resultado de aplicar *stemming* a las listas de lemas de la tercera columna.

Texto original	Preprocesado	Lematización	Stemming
'La casa está encantada. ¿Quién la desencantará?'	'casa', 'encantada', 'desencantará'	'caso', 'encantado', 'desencantar'	'cas', 'encant', 'desencant'
'El casero me ha arreglado la lavadora. ¡Qué encanto de señor!'	'casero', 'arreglado', 'lavadora', 'encanto', 'señor'	'casero', 'arreglar', 'lavadora', 'encanto', 'señor'	'cas', 'arregl', 'lavador', 'encant', 'señ'
'A Rubén le encanta lavar el coche.'	'ruben', 'encanta', 'lavar', 'coche'	'ruben', 'encanta', 'lavar', 'coche'	'rub', 'encant', 'lav', 'coch'

Tabla 2.1: Ejemplo de procesamiento de textos.

De los ejemplos anteriores se pueden destacar varias cosas. En primer lugar, hay que tener en cuenta que los modelos de lematización y *stemming* no son perfectos y pueden cometer errores. Es el caso de la palabra 'casa' cuyo resultado por el lematizador ha sido 'caso' que no se corresponde con su forma masculina, pues, de hecho, en este caso, no existe tal forma. El *stemming* por su parte ha cortado todas las palabras, incluso cuando estas no incluían sufijos. A pesar de estos errores, los modelos se han combinado para intentar homogeneizar los resultados de palabras que, en esencia, esconden el mismo significado. Es el caso de 'casa' y 'casero', cuyo resultado común ha sido 'cas', o de 'encantada', 'encanto' y 'encanta' que han terminado en 'encant'. No se ha conseguido lo mismo en el caso de 'lavadora' y 'lavar' debido a que el *stemming* no ha cortado lo suficiente la palabra 'lavadora'. En segundo lugar se observa que la lematización y el *stemming* han sido posibles gracias al preprocesado que ha dividido en *tokens* los textos, eliminando signos de puntuación y caracteres especiales y normalizando los términos. En tercer lugar, se destaca el hecho de que las palabras 'la', 'está', 'quién', 'el', 'me', 'ha', 'qué', 'de', 'a' y 'le' no han sido consideradas, pues están categorizadas como *stop-words*.

Una vez se ha homogeneizado el formato de los textos, se aplica un vectorizador para darles estructura, y así, poder procesarlos por modelos de analítica avanzada. Existen varias alternativas para hacer esto, pero aquí se desarrolla la construcción de una matriz de documentos - términos cuyos valores sean frecuencias de tipo tf-idf. Para calcular esta matriz han de calcularse los siguientes valores:

$$TF_{td} = \frac{\text{Número de veces que aparece } d \text{ en } t}{\text{Número total de términos de } d} \quad (2.1)$$

$$IDF_t = \log \left( \frac{\text{Número total de documentos}}{\text{Número de documentos en los que aparece } t} \right) \quad (2.2)$$

El coeficiente 2.1 representa la frecuencia de aparición de un término en un documento (*term frequency*, TF). El coeficiente 2.2, en cambio, representa una frecuencia inversa de un término en el corpus de documentos (*inverse document frequency*, IDF), es decir, su valor es mayor cuando el término de estudio aparece en pocos documentos. Con estas dos medidas, dado un documento  $d$  y un término  $t$ , se construye la frecuencia tf-idf como:

$$TFIDF_{td} = TF_{td} \cdot IDF_t \quad (2.3)$$

Con esta combinación de medidas se consigue que el coeficiente 2.3 sea directamente proporcional al número de apariciones de una término en un documento pero inversamente proporcional al número de apariciones del término en el corpus. Así, un término es más representativo de un documento cuanto mayor sea su frecuencia en él, pero menor si representa a muchos documentos. Así se quita

importancia a los términos que, sin ser *stopwords*, aparecen en muchos textos, y que por tanto no son representativos de ningún documento en particular.

Si  $D$  es el número de documentos de un corpus y  $T$  es el número de términos obtenidos en el procesamiento del corpus, la vectorización tf-idf consiste en construir una matriz  $DT \in \mathcal{M}_{D \times T}$  cuyos valores sean  $DT_{ij} = TFIDF_{ji}$  para cada  $i = 1, \dots, D$  y  $j = 1, \dots, T$ . Se obtiene así un conjunto de datos que representa el corpus original.

A continuación, para ilustrar el cálculo de los coeficientes TFIDF, se aplica la vectorización al corpus de la tabla 2.1. La matriz de documentos - términos obtenida se muestra en la tabla 2.2.

	arregl	cas	coch	desencant	encant	lav	lavador	rub	señ
'cas', 'encant', 'desencant'	0	$\frac{1}{3} \log(\frac{3}{2})$	0	$\frac{1}{3} \log(\frac{3}{1})$	$\frac{1}{3} \log(\frac{3}{3})$	0	0	0	0
'cas', 'arregl', 'lavador', 'encant', 'señ'	$\frac{1}{5} \log(\frac{3}{1})$	$\frac{1}{5} \log(\frac{3}{2})$	0	0	$\frac{1}{5} \log(\frac{3}{3})$	0	$\frac{1}{5} \log(\frac{3}{1})$	0	$\frac{1}{5} \log(\frac{3}{1})$
'rub', 'encant', 'lav', 'coch'	0	0	$\frac{1}{4} \log(\frac{3}{1})$	0	$\frac{1}{4} \log(\frac{3}{3})$	$\frac{1}{4} \log(\frac{3}{1})$	0	$\frac{1}{4} \log(\frac{3}{1})$	0

Tabla 2.2: Ejemplo de estructuración de textos.

Hay que destacar que este vectorizado no considera la ordenación de las palabras en los textos, tratándolos como un conjunto (*bag of words*). Esto hace que textos con los mismos conjuntos de palabras tengan una misma representación como dato. Cuando interese evitar esta casuística o cuando sea conveniente considerar la ordenación de las palabras, habría que recurrir a otro tipo de vectorizado.

### 2.1.2. Reducción de dimensionalidad

La reducción de dimensionalidad es la tarea mediante la cuál se proyecta un conjunto de datos  $x_1, \dots, x_n$  de un espacio de dimensión  $p$ , en un conjunto de puntos  $y_1, \dots, y_n$  en un espacio de dimensión menor,  $q < p$ , de forma que las proyecciones sean similares a los puntos originales según algún criterio de distancias, varianzas, geometrías, etc.. El objetivo principal de la reducción de dimensionalidad es el de simplificación: obtener un conjunto de datos sencillo que refleje fielmente la información del conjunto original. Este cometido ha adquirido mucha importancia en los últimos años dentro del *Machine Learning*, sobretodo desde que es habitual entrenar modelos de clasificación sobre imágenes o textos, que en formato estructurado pueden llegar a tener miles de variables (tantas como píxeles tiene una imagen o como *tokens* tiene un texto). Además, la reducción de dimensionalidad ha irrumpido en la ciencia de datos con un doble objetivo:

- Reducir el tamaño del *input* de los modelos, abaratando así el proceso de entrenamiento, en términos de tiempo y memoria.
- Facilitar la comprensión de la estructura (global y local) de los datos o, incluso, hacer posible su visualización (reduciéndolos a espacios de dos o tres dimensiones).

Para abordar estos objetivos se han desarrollado diferentes técnicas. El algoritmo clásico es el PCA (*Principal Component Analysis*) con el que se proyectan los datos en sucesivas direcciones independientes intentando maximizar la varianza de estos en cada dimensión (componente principal). De este algoritmo se han desarrollado diferentes versiones para mejorar su eficacia o reducir costes (*Singular Values Decomposition* o *Independent Component Analysis*). Estos métodos capturan relaciones lineales entre las variables, y lo que consiguen es mantener separados en el espacio reducido aquellos datos que originalmente eran muy diferentes. Sin embargo, esto no asegura que se mantengan las estructuras de entornos de los datos. De hecho, esto es una tarea bastante complicada, y, para intentar darle solución, se han desarrollado en los últimos años diferentes técnicas de reducción de dimensionalidad no lineales. Algunos de estos algoritmos son LLE (*Locally Linear Embedding*), SOM (*Self-Organizing Map*), Isomap (*Isometric Mapping*), UMAP (*Uniform Manifold Approximation and Projection*) y SNE (*Stochastic Neighbor Embedding*) y sus variantes como SNE simetrizado o t-SNE (*t-distributed*). Estas técnicas abordan la tarea de reducción de dimensionalidad desde diferentes enfoques, algunas con el objetivo de preservar la geometría de los datos, otras centrándose en la topología, pero, en cualquier caso, todas buscando superar la eficacia de las técnicas clásicas. Es cierto que estos nuevos desarrollos han obtenido mejoras a la hora de representar correctamente los entornos de los datos, pero no todos ellos parecen tener éxito en mantener la estructura global a la par que la local.

A continuación se desarrollan los detalles de los algoritmos PCA y t-SNE, pues son los algoritmos que serán objeto de estudio y comparación a lo largo del caso práctico. Estos modelos han sido los seleccionados debido a que son representativos de entre las técnicas clásicas lineales y los nuevos desarrollos no lineales, respectivamente. Mientras que PCA es el algoritmo por excelencia que se explica en cualquier curso introductorio de minería de datos, t-SNE ha adquirido mucha popularidad en los últimos años y parece que es uno de los pocos que es capaz de reflejar la estructura global de los datos además de la local.

### 2.1.2.1. Análisis de Componentes Principales, PCA

El PCA es una técnica estadística que busca reducir la dimensionalidad de conjuntos de datos complejos, manteniendo la mayor cantidad posible de información relevante. Este método busca una representación reducida de los datos mediante variables denominadas componentes principales, que son combinaciones lineales de las variables originales, que están incorreladas y que tratan de describir la estructura de correlación de las mismas. Estas componentes principales se calculan y se presentan de forma ordenada, cada una explicando más variabilidad de los datos que la siguiente. Con esto se consigue que cada conjunto de las  $k$  primeras componentes sea el conjunto más reducido de  $k$  variables que explica más información sobre los datos.

El algoritmo utiliza nociones de álgebra lineal para calcular las componentes principales. Consiste en calcular los autovalores y autovectores de la matriz de varianzas-covarianzas, que es la que explica la estructura de correlación de los datos. Tomando como componentes principales los autovectores (combinaciones lineales de las variables originales) y ordenándolas de forma decreciente por su autovvalor asociado se obtiene el resultado. Esto siempre se puede hacer, ya que la matriz de *varianzas-covarianzas* es una matriz simétrica y real y se puede demostrar que sus autovalores representan la varianza explicada por el autovector asociado.

Los primeros desarrollos sobre esta idea de transformación se pueden leer en [4]. A continuación se presenta el esquema de cálculo de las componentes principales paso por paso:

### Algoritmo 2.1.2.1.1 (PCA)

1. Escribir los datos  $x_1, \dots, x_n$  en forma de  $p$  componentes  $V_1, \dots, V_p$ .
2. Calcular la matriz de varianzas-covarianzas de  $V_1, \dots, V_p$ :

$$\Sigma = \begin{pmatrix} \text{var}(V_1) & \text{cov}(V_1, V_2) & \cdots & \text{cov}(V_1, V_p) \\ \text{cov}(V_2, V_1) & \text{var}(V_2) & \cdots & \text{cov}(V_2, V_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(V_p, V_1) & \text{cov}(V_p, V_2) & \cdots & \text{var}(V_p) \end{pmatrix}$$

3. Calcular los autovalores y autovectores de  $\Sigma$ ,  $(\lambda_1, v_1), \dots, (\lambda_p, v_p)$ .
4. Tomar como componentes principales  $PCA_i = v_{\sigma(i)}$  para cada  $i = 1, \dots, p$ , donde  $\sigma$  es una permutación para la que se verifica que  $\lambda_{\sigma(1)} \geq \dots \geq \lambda_{\sigma(p)}$ .
5. Escoger el número de componentes principales  $k$ .
6. Obtener los datos  $y_1, \dots, y_n$  proyectando  $x_1, \dots, x_n$  en las variables  $PCA_1, \dots, PCA_k$ .

Observando el esquema queda claro que un aspecto relevante al aplicar el PCA es la elección del número de componentes principales. Para tomar esta decisión es importante tener en mente criterios de calidad, pues, en caso contrario, podría ocurrir que en la transformación del conjunto de datos se reduzca demasiado la información como para que los análisis posteriores perdieran sentido. Es por ello que el PCA no se usa para reducir un conjunto de datos a un número predefinido de variables, sino que se aplica para simplificar, en la medida de lo posible, un conjunto de datos de muchas características.

A continuación se listan los criterios de calidad más utilizados para decidir el número de componentes principales a seleccionar:

- Método de la proporción de varianza explicada: se calcula la proporción de varianza explicada de cada componente respecto del total y se eligen las componentes principales necesarias para que la proporción de varianza acumulada supere un ratio prefijado.
- Método de codo/rodilla: se calcula la proporción de varianza explicada de cada componente respecto del total, se grafican estos valores respecto de las varianzas explicadas absolutas y se busca el punto de inflexión.
- Método de Kaiser: se seleccionan las componentes principales cuya varianza explicada es superior a 1 y solo se aplica cuando las variables originales han sido estandarizadas. Una variable estandarizada tiene varianza  $\sigma = 1$ , luego tomar las componentes con varianza explicada superior a 1 es tomar las componentes que explican más varianza que las variables originales.

En función de las necesidades del análisis posterior, se escoge un método u otro y se determina un número determinado de componentes. En ocasiones, esta elección se incluye en la calibración de los modelos posteriores, y se escoge el número de componentes mediante dinámicas de prueba y error.

Aunque el PCA es una técnica sencilla que permite sacar y simplificar la información subyacente de un conjunto de datos, tiene varias limitaciones. En primer lugar, puede darle más peso a variables

que tienen una varianza mayor, pudiendo hacer que los análisis posteriores estén sesgados. Esto se puede solucionar aplicando el algoritmo sobre la matriz de correlaciones en vez de la de varianzas-covarianzas o, equivalentemente, estandarizando las variables antes de aplicar el algoritmo. En segundo lugar, PCA es esencialmente una transformación lineal de los datos, lo que hace que capture alguna de las relaciones lineales de las variables originales, pero sea incapaz de capturar relaciones más complejas.

A pesar de sus limitaciones, PCA es una técnica sencilla, fácil de entender y no muy costosa computacionalmente, que permite simplificar grandes conjuntos de datos. Es por ello que es muy usada, sola o en combinación de técnicas más complejas.

### 2.1.2.2. t-Distributed Stochastic Neighbor Embedding, t-SNE

El t-SNE es una técnica de reducción de dimensionalidad no lineal que está diseñada para preservar las estructuras de entornos de los datos. Gracias a este enfoque, t-SNE es eficaz cuando se busca analizar agrupamientos o relaciones intrínsecas complejas (no lineales) en conjuntos de datos.

La esencia del t-SNE se presentó en [5], en donde se publicó por primera vez el algoritmo SNE. Este algoritmo no lineal se presentaba como una mejora a LLE o SOM, que aunque, como el SNE, estaban enfocados en mantener la estructura local de los datos, parecían tener problemas al juntar puntos que originalmente estaban muy separados.

El SNE funciona de la siguiente manera. Dados  $x_1, \dots, x_n$  puntos en un espacio de dimensión  $p$ , hay que encontrar  $y_1, \dots, y_n$  puntos en un espacio de dimensión  $q < p$  cuya geometría sea similar a la de los puntos originales. Para ello, para cada  $i = 1, \dots, n$  se modelizan como distribuciones gaussianas dos variables aleatorias:

- La asociada a la probabilidad de que un punto  $x_j$  esté en el entorno de  $x_i$ .
- La asociada a la probabilidad de que un punto  $y_j$  esté en el entorno de  $y_i$ .

Concretamente, estas distribuciones se definen como:

$$P_i = \left\{ p_{ij} = \frac{e^{-\frac{\|x_j - x_i\|^2}{2\sigma_i^2}}}{\sum_{k=1}^n e^{-\frac{\|x_k - x_i\|^2}{2\sigma_i^2}}} \mid j = 1, \dots, n, j \neq i \right\} \cup \{ p_{ii} = 0 \} \quad (2.4)$$

$$Q_i = \left\{ q_{ij} = \frac{e^{-\|y_j - y_i\|^2}}{\sum_{k=1}^n e^{-\|y_k - y_i\|^2}} \mid j = 1, \dots, n, j \neq i \right\} \cup \{ q_{ii} = 0 \} \quad (2.5)$$

Dado un valor del parámetro *perplexity*,  $k$ , para cada  $i = 1, \dots, n$  se calibra el valor de  $\sigma_i$  imponiendo la condición  $H(P_i) = \log_2(k)$  donde  $H(P_i) = -\sum_{j=1}^n p_{ij} \log_2(p_{ij})$  representa la entropía de *Shannon* ([6]) de  $P_i$ . Con esto se consigue que en cada  $P_i$  sean  $k$  los puntos en los que se acumula la mayoría de la probabilidad, que son los  $k$  puntos más cercanos a  $x_i$ . Así, con el parámetro *perplexity*, se consigue controlar el tamaño de los entornos cuya geometría se quiere mantener en la reducción de dimensionalidad.

Con esta modelización, el SNE aborda la tarea de reducir dimensionalidad calculando los puntos  $y_1, \dots, y_n$  de forma que las distribuciones  $P_i, Q_i$  sean lo más parecidas posible dos a dos. Para ello se utiliza la divergencia de Kullback-Leibler ([7]) que mide la diferencia entre dos distribuciones de probabilidad. La función objetivo que se plantea a minimizar es  $F_{obj} = \sum_{i=1}^n D_{KL}(P_i || Q_i) = \sum_{i=1}^n \sum_{j=1}^n p_{ij} \cdot \log\left(\frac{p_{ij}}{q_{ij}}\right)$ . Se puede demostrar que una expresión del grandiente de  $F_{obj}$  es:

$$\nabla F_{obj} = \left( \frac{\partial F_{obj}}{\partial y_i} \right)_{i=1}^n = \left( 2 \sum_{j=1}^n (p_{ij} - q_{ij} + p_{ji} - q_{ji})(y_i - y_j) \right)_{i=1}^n \quad (2.6)$$

Para resolver el problema de optimización se utiliza el método de descenso de gradiente ([8]) con momento ([9], [10]). Dado un valor inicial de los puntos  $y_1, \dots, y_n$ , un *learning\_rate*, una función *momentum* y unos criterios de convergencia, se actualizan iterativamente los puntos en base a su valor en el gradiente de  $F_{obj}$ , al *learning\_rate* y a *momentum*, hasta alcanzar convergencia.

El SNE demostró alcanzar un mayor rendimiento que el resto de algoritmos en mantener la geometría (tanto local como global) de los datos al reducir dimensionalidad. Sin embargo, se descubrieron varios puntos débiles que limitaban el uso del algoritmo: la complejidad de la expresión 2.6 implica un gran coste computacional; la asimetría de las distribuciones  $P_i$  hace que SNE sea muy sensible a *outliers*; el *crowding problem* (problema de "apiñamiento") que describe la imposibilidad de preservar simultáneamente la distancia entre todos los puntos vecinos. Para dar solución a estos problemas sin renunciar al enfoque del SNE, *Maaten* y *Hinton* presentan en [1] el t-SNE.

Para reducir el coste computacional y mejorar la representación de los *outliers*, se comienza presentando una versión simétrica del SNE. En esta versión se consideran las siguientes distribuciones:

$$P'_i = \left\{ p'_{ij} = \frac{p_{ij} + p_{ji}}{2n} \mid j = 1, \dots, n \right\} \quad (2.7)$$

$$Q'_i = \left\{ q'_{ij} = \frac{q_{ij} + q_{ji}}{2n} \mid j = 1, \dots, n \right\} \quad (2.8)$$

Tomando como función objetivo  $F_{obj} = \sum_{i=1}^n D_{KL}(P'_i || Q'_i)$  se obtiene una expresión más sencilla del grandiente:

$$\nabla F_{obj} = \left( \frac{\partial F_{obj}}{\partial y_i} \right)_{i=1}^n = \left( 4 \sum_{j=1}^n (p'_{ij} - q'_{ij})(y_i - y_j) \right)_{i=1}^n \quad (2.9)$$

Esta expresión simplificada del gradiente hace que el coste computacional de la optimización se reduzca significativamente. Además, esta simetrización hace que para cada  $i = 1, \dots, n$  se cumpla que  $\sum_{j=1}^n p'_{ij} > \frac{1}{2n}$  lo que hace que todos los puntos, sean o no *outliers*, tengan un peso mínimo en el proceso de optimización. Esto supone una mejora en la eficacia de la proyección de los *outliers* en el espacio de dimensión  $q$ .

Tras proponer esta simetrización, *Maaten* y *Hinton* presentan el problema conocido como *crowding problem* que aparece en los resultados del SNE y también de su versión simétrica y en otras técnicas de reducción de dimensionalidad. Este fenómeno consiste en que entornos con muchos puntos se proyectan en regiones muy pequeñas, siendo las distancias entre los puntos proyectados tan pequeñas y similares que no reflejan correctamente la estructura del entorno original. Esto ocurre debido a que al reducir la dimensión del espacio, se pierde volumen en los entornos y por tanto es más difícil mantener los esquemas de distancias cuando estos contienen muchos puntos.



La mejora que proponen *Maaten* y *Hinton* para evitar los problemas de "apiñamiento" es el t-SNE. Este algoritmo propone continuar modelizando las similitudes del espacio original con una distribución gaussiana, pero cambiar en el espacio proyectado esta distribución por una *t-Student* de un grado de libertad. La elección de esta distribución se debe a la siguientes razones:

- La *t-Student* es una distribución de cola pesada lo que facilita que las proyecciones se expandan, aliviando los problemas de "apiñamiento".
- La *t-Student* está muy relacionada con la distribución gaussiana.
- La *t-Student* es más fácil de evaluar que una gaussina, computacionalmente hablando.

Con este cambio de distribución, las similaridades en el espacio original se siguen modelando como en 2.4, sin embargo, las similaridades 2.5 del espacio proyectado se sustituyen por:

$$Q_i = \left\{ q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k=1}^n (1 + \|y_k - y_i\|^2)^{-1}} \mid j = 1, \dots, n, j \neq i \right\} \cup \{ q_{ii} = 0 \} \quad (2.10)$$

Realizando las simetrización como en 2.7 y 2.8 y tomando, de nuevo,  $F_{obj} = \sum_{i=1}^n D_{KL}(P'_i || Q'_i)$ , se puede demostrar que la nueva expresión del gradiente es:

$$\nabla F_{obj} = \left( \frac{\partial F_{obj}}{\partial y_i} \right)_{i=1}^n = \left( 4 \sum_{j=1}^n (p'_{ij} - q'_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j) \right)_{i=1}^n \quad (2.11)$$

Aunque este cambio no asegura que no se puedan dar casos de "apiñamineto", en términos generales, sí que alivia este problema, a la par que disminuye el coste computacional de la optimización (pues el cálculo de los  $q_{ij}$  ya no involucra una exponencial).

A continuación se detalla el esquema de computación del t-SNE:

#### Algoritmo 2.1.2.2.1 (t-SNE)

1. Escoger un valor  $k$  de perplexity, un valor  $\alpha > 0$  de *learning\_rate* y una función  $\beta(t)$  de momentum.
2. Calcular las distribuciones  $P_i$  como en la fórmula 2.4 calibrando con  $k$  los valores de  $\sigma_i$ .
3. Calcular las distribuciones  $P'_i$  como en la fórmula 2.7.
4. Fijar  $t = 0$  y elegir las proyecciones iniciales:

$$\begin{aligned} \mathcal{Y}^{(-1)} &= \{0, \dots, 0\} \\ \mathcal{Y}^{(0)} &= \{y_1^0, \dots, y_n^0\} \end{aligned}$$

5. Calcular las distribuciones  $Q_i(\mathcal{Y}^{(t)})$  como en la fórmula 2.10.
6. Calcular las distribuciones  $Q'_i(\mathcal{Y}^{(t)})$  como en la fórmula 2.8.



7. Actualizar  $t = t + 1$  y con la fórmula 2.11 calcular la nueva proyección como:

$$\{y_1^t, \dots, y_n^t\} = \mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} - \alpha \nabla F_{obj}(\mathcal{Y}^{(t-1)}) + \beta(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$$

8. Evaluar los criterios de convergencia. Si no se han satisfecho, volver la paso 5.

El algoritmo t-SNE ofrece una metodología robusta para reducir la dimensionalidad de conjuntos de datos complejos manteniendo la geometría global y local de los datos. Sin embargo, esta técnica no está libre de limitaciones. La mayor limitación de t-SNE es su complejidad computacional. Como ya se ha comentado, la simetrización y la modelización de las similitudes del espacio proyectado mediante distribuciones *t-Student* aumentaron la eficiencia del t-SNE respecto del SNE, pero aun con estas mejoras, t-SNE sigue siendo una técnica de reducción de dimensionalidad limitada por su coste computacional. El diseño de t-SNE hace que su coste aumente significativamente según el número de dimensiones del espacio proyectado. Es por esto que, actualmente, t-SNE se utiliza como método de visualización de conjuntos complejos de datos, pues solo se implementa para reducir datos a espacios de dimension  $q = 2, 3$ , permitiendo graficar los resultados obtenidos. También por su diseño, el coste de t-SNE aumenta cuadráticamente con el tamaño del conjunto de datos a proyectar, lo que, a día de hoy, hace que solo pueda aplicarse a conjuntos con, a lo sumo, decenas de miles de datos. Aun siendo grandes las limitaciones del algoritmo t-SNE, este se ha convertido en una técnica muy popular en la ciencia de datos que ayuda a comprender estructuras y relaciones en los datos que se mantenían escondidas al aplicar las metodologías clásicas. Es por ello que surgen desarrollos para aumentar la eficiencia del t-SNE. En [11] se plantean las causas del coste computacional del t-SNE y se presentan variantes que aumentan la velocidad en la convergencia del t-SNE, con el objetivo de aumentar la escalabilidad del algoritmo a mayores conjuntos de datos.

Otra limitación del algoritmo es que los resultados proporcionados son muy sensibles a modificaciones en los parámetros  $k, \alpha$ , la función  $\beta$  y la proyección inicial de los datos. Esto hace que los procesos de calibración puedan llegar a ser complejos. Otra carencia del t-SNE es que no ofrece una fórmula de proyección si no que calcula, mediante optimización, una proyección de los datos de partida. Esto supone varios inconvenientes: dificulta la interpretabilidad de los datos; impide extrapolar la proyección a conjuntos de *test*.

A pesar de las limitaciones, los resultados del t-SNE han demostrado suponer una avance en captar y mantener tanto las estructuras de entornos como la geometría global al reducir de dimensión conjuntos complejos de datos. Por ello, la popularidad de t-SNE ha crecido mucho en los últimos años, siendo una opción muy atractiva en los procesados iniciales de conjuntos de datos, para obtener información de los mismos y decidir con criterio como utilizar los datos y con qué objetivos.

### 2.1.3. Clustering

Se conoce como *clustering* a las técnicas de *Machine Learning* mediante las cuales se intenta segmentar un conjunto de datos en diferentes categorías llamadas *clusters*. Obtener segmentaciones en los datos es útil para entender la estructura subyacente en los mismos, lo que puede servir para segmentar carteras de clientes y adaptar las medidas reactivas a los tipos de clientes o para detectar comunidades de usuarios en redes sociales en *marketing*, para agrupar transacciones parecidas y detectar fraudes en finanzas, para analizar los diferentes usos de redes y gestionar las cargas de tráfico en telecomunicaciones, etc..

Estas técnicas tratan de agrupar los datos en base a algún criterio de similitud, ya sea de proximidad, de pertenencia a un mismo grupo de alta densidad de puntos o de maximizar la probabilidad de estar generados por una misma distribución. Esta variedad de criterios ha dado lugar a diferentes tipos de algoritmos, que tienen esquemas totalmente diferentes, y producen segmentaciones muy dispares. También existen diferentes métricas que tratan de medir la calidad de una segmentación de un conjunto de datos, que atienden a diferentes criterios de similitud. Durante esta sección se explican, en primer lugar, los tipos de algoritmos de *clustering* más populares, y en segundo lugar, las métricas más usadas que evalúan la calidad de unos *clusters* construidos a partir de un conjunto de datos.

Es importante conocer diferentes tipos de algoritmos y diferentes métricas de evaluación de *clusters*, pues no todos los algoritmos ni todas las métricas son adecuados para todos los objetivos. Teniendo claro el objetivo para el que se segmentan unos datos y conociendo diversas técnicas, se puede evaluar con criterio qué algoritmo aplicar y cómo evaluar el resultado.

### 2.1.3.1. Algoritmos de clustering

Como ya se ha comentado, existen diferentes tipos de algoritmos de *clustering*, cada uno de ellos abordando la tarea de agrupar datos desde un enfoque distinto. A continuación se explican cuatro tipos de algoritmos: los particionales, los jerárquicos, los basados en conceptos de densidad (*density-based*) y los basados en estimación de densidades de probabilidad. De cada uno de estos tipos se expone el esquema de un algoritmo que será usado en el caso práctico: *k-Means*, jerárquico aglomerativo, DBSCAN (*Density Based Spatial Clustering of Applications with Noise*) y GMM (*Gaussian Mixutre Models*).

#### Algoritmos particionales:

Los algoritmos particionales agrupan los datos con un criterio de proximidad definido por una distancia. Para hacer esto, siguen un esquema iterativo en el que en cada etapa se determinan unos centroides que definen unos *clusters*. La forma de determinar estos centroides es lo que distingue a los distintos algoritmos particionales: en *k-Means* se toman como centroides las medias aritméticas de los *clusters* de la iteración anterior, en *k-Medoids* se escogen los puntos de cada *cluster* de la iteración anterior que minimizan la suma de distancias al resto de puntos del *cluster*, etc.. Aunque la forma de calcular los centroides pueda variar, en general, los *clusters* siempre se construyen agrupando los datos cuyo centroide más cercano es el mismo.

En cada iteración se modifican los centroides, sin embargo, el número de centroides no varía, pues no se añaden ni eliminan centroides. Esto hace que el número de *clusters* sea constante y que haya que prefijarlo al inicializar el algoritmo. Así, el resultado del algoritmo depende fuertemente de la elección del número de *clusters*. Además, para poder inicializar el algoritmo es necesario elegir unos centroides iniciales. Esta elección también afecta a los resultados. Otra elección necesaria para implementar un algoritmo particional, y que puede influir en los resultados obtenidos, es la de los criterios de convergencia. En general, un algoritmo particional se detiene si en una iteración se obtiene la misma clasificación que en la iteración anterior, pero se pueden añadir criterios de convergencia adicionales que permitan parar el algoritmo antes.

A continuación se detalla la estructura del *k-Means* ([12]):

#### Algoritmo 2.1.3.1.1 (*k-Means*)

1. Escoger el número esperado de clusters,  $k$ .
2. Escoger  $k$  centroides iniciales.
3. Asociar a cada punto su centroide más cercano. Construir los clusters de la nueva segmentación como los puntos asociados a un mismo centroide.
4. Calcular los nuevos centroides como la media aritmética de los puntos de cada cluster.
5. Evaluar los criterios de convergencia. Si no se han satisfecho y los nuevos centroides son distintos a los anteriores, volver al paso 3.

Con esta estructura queda patente que el resultado depende de la elección de  $k$ , de los centroides iniciales y de los criterios de convergencia. Sin embargo, también depende de la distancia escogida (en el paso 3 se construye la nueva segmentación en base a criterios de distancia). Se puede demostrar que *k-Means* siempre alcanza la convergencia con independencia de la elección de distancia, sin embargo, esta sí que puede afectar a la velocidad de convergencia o a si esta resulta en un mínimo local.

Aunque el *k-Means* es el algoritmo particional más conocido, existen otros algoritmos particionales que, según el caso, podrían considerarse más adecuados. Existe el *k-Medoids* o PAM (*Partitional Around Medoids*, [13]), que es muy parecido al *k-Means*, pero en este esquema se fuerza a que los centroides sean puntos del propio cluster. Se ha probado que esta ligera modificación reduce la sensibilidad del algoritmo a *outliers*, lo que puede resultar muy útil cuando existen datos atípicos en nuestro conjunto de datos. El inconveniente es que se aumenta el coste computacional. Con el objetivo de mejorar la eficiencia computacional de PAM se han desarrollado algoritmos más complejos que escapan del esquema clásico de los algoritmos particionales: CLARA (*Clustering Large Applications*, [13]) que en cada iteración aplica PAM sobre una muestra de datos para obtener unos centroides provisionales que se evalúan y se modifican para aplicarlos al conjunto total, y, más adelante, CLARANS (*Clustering Large Applications based on Randomized Search*, [14]) que aplica una estrategia de búsqueda aleatoria para hacer modificaciones en los centroides y evitar quedarse atascado en mínimos locales. Con un enfoque completamente diferente, existe también el *Fuzzy k-Means* ([15]). Este algoritmo está basado en el *k-Means*, pero escapa del esquema particional, pues en cada iteración no se asocia cada punto a un cluster sino que se calcula su grado de pertenencia a todos los clusters. Esto puede mejorar los resultados en los casos en los que se intuye que puede haber clusters solapados.

#### Algoritmos jerárquicos:

Los algoritmos jerárquicos, como los particionales, agrupan datos en base a criterios de proximidad que vienen determinados por una distancia y un método de actualización de distancias. Estos algoritmos siguen un esquema iterativo sobre el número de clusters, variando este desde 1 hasta el número total de puntos, o al revés. En cada iteración se construye un conjunto de clusters a partir del conjunto de la iteración anterior. Esto se hace agrupando dos clusters de la iteración anterior o dividiendo uno en dos, dependiendo del tipo de algoritmo. El hecho de que estas clasificaciones sean encajadas, permite que se puedan representar todas a la vez en una especie de grafo o diagrama, denominado *dendrograma*, con el que se puede estudiar el proceso y tomar la decisión final de escoger una clasificación.

Existen dos tipos de algoritmos jerárquicos: aglomerativos y divisivos. Los algoritmos aglomerativos se inician con tantos *clusters* como puntos y en cada iteración se buscan los dos *clusters* más cercanos y se unen. El proceso se itera hasta llegar a tener un único *cluster* con todos los puntos. Por el contrario, los algoritmos divisivos se inician con un único *cluster* que contiene a todos los puntos y en cada iteración se selecciona un *cluster* para dividir en dos atendiendo a criterios de distancia dentro del *cluster*. En este caso, el proceso se itera hasta tener tantos *clusters* unitarios como puntos.

A continuación se detalla la estructura de un algoritmo jerárquico aglomerativo ([16]):

**Algoritmo 2.1.3.1.2 (Clustering jerárquico aglomerativo)**

1. Inicializar con una segmentación de  $n$  clusters unitarios. Construir la matriz de distancias entre clusters asociada (matriz de distancias entre puntos).
2. Seleccionar la mínima entrada no diagonal de la matriz de distancias.
3. Unir los clusters correspondientes a la fila y a la columna de la entrada seleccionada:
  - 4.1. Calcular la distancia del nuevo cluster al resto de cluster en base a las distancias de los clusters originales.
  - 4.2. Actualizar la matriz de distancias.
4. Si quedan más de un cluster, volver al paso 3.
5. Construir el dendrograma y escoger una segmentación.

Se puede observar que este método depende esencialmente de cómo actualizar las distancias entre *clusters*. En 1967 ([16]), se propuso un fórmula paramétrica que aglomeraba todos los métodos utilizados hasta el momento para actualizar las distancias entre *clusters*:

$$d(A \cup B, C) = \alpha_A d(A, C) + \alpha_B d(B, C) + \gamma d(A, B) + \delta |d(A, C) - d(B, C)|$$

En función de la elección de los parámetros  $\alpha_A, \alpha_B, \gamma, \delta \in \mathbb{R}$  se obtiene lo que se conoce como método de enlace (fórmula de actualización de las distancias entre *clusters*). Algunos ejemplos de los métodos clásicos de enlace son los siguientes:

- Enlace sencillo: la distancia del nuevo *cluster* a otro *cluster* se actualiza como el mínimo de las distancias de los *clusters* originales al tercero. Se obtiene tomando  $\alpha_A = 0,5, \alpha_B = 0,5, \gamma = 0, \delta = -0,5$ :

$$d(A \cup B, C) = \min\{d(A, C), d(B, C)\}$$

- Enlace diámetro: la distancia del nuevo *cluster* a otro *cluster* se actualiza como el máximo de las distancias de los *clusters* originales al tercero. Se obtiene tomando  $\alpha_A = 0,5, \alpha_B = 0,5, \gamma = 0, \delta = 0,5$ :

$$d(A \cup B, C) = \max\{d(A, C), d(B, C)\}$$

- Enlace medio: la distancia del nuevo *cluster* a otro *cluster* se actualiza como la media de las distancias de los *clusters* originales al tercero ponderada por su número de elementos. Se obtiene tomando  $\alpha_A = \frac{|A|}{|A \cup B|}, \alpha_B = \frac{|B|}{|A \cup B|}, \gamma = \delta = 0$ :

$$d(A \cup B, C) = \frac{1}{|A \cup B|} (|A| \cdot d(A, C) + |B| \cdot d(B, C))$$

Además de elegir de forma coherente el método de enlace, es importante tener una metodología robusta para elegir el número de *clusters*, y, por tanto, la clasificación final. Existen varias formas de tomar esta decisión (el método codo/rodilla, el método Calinski y Harabasz, etc.) que están basadas en los criterios de calidad de segmentaciones de datos. En la sección 2.1.3.2 se explican estos criterios y se dan ejemplos de las métricas más utilizadas para medirlos.

### Algoritmos basados en conceptos de densidad:

Los algoritmos basados en conceptos de densidad o *density-based* analizan los entornos de los datos a agrupar y conectan cada uno con los datos de su entorno cuando se cumple cierto criterio de densidad predefinido. Tras este análisis, se construyen los *clusters* como las componentes conexas del grafo construido. Este tipo de dinámica permite detectar *clusters* de formas y tamaños diversos, y suele ser bastante robusta a *outliers*.

En los algoritmos *density-based* no se prefija el número de *clusters* como en los algoritmos particionales, aunque tampoco se escoge directamente, como en los jerárquicos. Estos algoritmos dependen de dos tipos de parámetros, los que definen cómo son los entornos de cada punto que hay que analizar y los que dictan el criterio de densidad que tienen que cumplir dichos entornos para conectar sus puntos.

El algoritmo basado en densidad más utilizado es el DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*, [17]). Este algoritmo es biparamétrico. Dados  $\epsilon$ ,  $min\_samples$ , DBSCAN analiza el entorno de radio  $\epsilon$  de cada punto y si en dicho entorno hay más de  $min\_samples$  puntos, se considera que el punto cumple la condición de densidad mínima. Haciendo este análisis, se consideran tres tipos de puntos:

- CORE: punto en cuyo entorno de radio  $\epsilon$  hay al menos  $min\_samples$  puntos.
- BORDER: punto que sin ser central está en el entorno de radio  $\epsilon$  de un punto central.
- NOISE: punto que no es ni central ni fronterizo.

Esta categorización es exhaustiva y sus clases son mutuamente excluyentes. Esto hace que el proceso de análisis de los puntos sea determinista y no dependa del orden en el que se analizan los puntos. Sin embargo, esto no quiere decir que el algoritmo en sí sea determinista.

A continuación se concreta el esquema del DBSCAN:

#### Algoritmo 2.1.3.1.3 (DBSCAN)

1. Elegir los parámetros  $\epsilon$ ,  $min\_samples$ .
2. Elegir aleatoriamente un punto que no haya sido clasificado y analizar su pertenencia a los clusters existentes y su entorno de radio  $\epsilon$ .
  - 2.a. Si el punto escogido no pertenece a ningún cluster y su entorno de radio  $\epsilon$  contiene al menos  $min\_samples$  puntos, se etiqueta como CORE y se crea un cluster con los puntos del entorno (si alguno de estos punto está etiquetado como NOISE, hay que cambiar su etiqueta a BORDER).
  - 2.b. Si el punto escogido no pertenece a ningún cluster y su entorno de radio  $\epsilon$  contiene menos de  $min\_samples$  puntos, se etiqueta como NOISE.

- 2.c. Si el punto escogido pertenece a algún cluster y su entorno de radio  $\epsilon$  contiene al menos  $\min\_samples$  puntos, se etiqueta como *CORE* y se añaden los puntos del entorno al cluster al que pertenece el punto escogido (si alguno de estos punto está etiquetado como *NOISE*, hay que cambiar su etiqueta a *BORDER*).
- 2.d. Si el punto escogido pertenece a algún cluster y su entorno de radio  $\epsilon$  contiene menos de  $\min\_samples$  puntos, se etiqueta como *BORDER*.
3. Si quedan puntos sin etiquetar, volver al paso 2.

Con este esquema queda patente la importancia de la elección de los parámetros  $\epsilon$ ,  $\min\_samples$  y de la distancia que se utiliza en el análisis de cada punto para construir su entorno de radio  $\epsilon$ . Aunque existen varias técnicas diseñadas para realizar una elección adecuada de estos parámetros, estas han sido diseñadas en base a criterios de optimalidad de segmentación de puntos. En la sección 2.1.3.2 se explican estos criterios.

Hay que mencionar que el algoritmo no es completamente determinístico, pues el orden en el que se analizan los puntos puede influir en el resultado. Este caso se da cuando un punto es *BORDER* pero está en el entorno de varios puntos *CORE* de *clusters* diferentes, pues, en este caso el punto *BORDER* será incluido en el *cluster* del primer punto *CORE* analizado. Así, los posibles resultados del algoritmo son idénticos salvo para estos puntos, que no dejan de ser casos muy singulares.

El algoritmo DBSCAN presenta claras ventajas, como la capacidad para detectar *clusters* de diferentes formas o la robustez frente a *outliers*. Sin embargo, puede ser difícil obtener buenos resultados cuando existen grupos de datos con diferentes niveles de densidad. En estos casos puede ser difícil calibrar los parámetros e incluso puede ocurrir que ninguna elección de estos proporcione los resultados esperados. Es por esta limitación que se han desarrollado extensiones escalables del algoritmo enfocadas en la detección de *clusters* con diferentes densidades. Algunos ejemplos son el HDBSCAN (*Hierarchical Density-Based Spatial Clustering of Applications with Noise*, [18]), OPTICS (*Ordered Points To Identify the Clustering Structure*, [19]) o GDBSCAN (*Generic Density-Based Spatial Clustering of Applications with Noise*, [20]).

### Algoritmos basados en estimación de densidades de probabilidad:

Otras metodologías de *clustering* son las basadas en inferir distribuciones de probabilidad sobre los datos. Estas técnicas parten de la consideración de que los datos provienen de una variable aleatoria cuya distribución tiene varias componentes y tratan de inferir su distribución de probabilidad. Dada la muestra de datos y la distribución con varias componentes, se evalúa cómo de verosímil es observar cada punto en cada componente, y a cada punto se le asocia el componente de mayor verosimilitud. Finalmente, se construyen los *clusters* agrupando los puntos que están asociados a cada componente.

En general, se asume que los datos provienen de una combinación lineal de variables aleatorias con mismo tipo de distribución y se aplica alguna técnica de inferencia paramétrica para calcular los parámetros de cada componente y los pesos de la combinación. El algoritmo más famoso es el GMM (*Gaussian Mixture Models*, [21]) que asume que los datos provienen de una combinación lineal (convexa) de distribuciones normales:

$$F(x) = \sum_{i=1}^k \pi_i N_{\mu_i, \sigma_i}(x) \quad (2.12)$$



Se calculan los parámetros  $\mu = (\mu_1, \dots, \mu_k), \sigma = (\sigma_1, \dots, \sigma_k), \pi = (\pi_1, \dots, \pi_k)$  para que se maximice la probabilidad de que los datos provengan de la distribución  $F$ . Para ello, se aplica el método EM (*Expectation Maximization*) sobre función de verosimilitud de  $F$  respecto de una muestra de datos. Conocidos  $\mu, \sigma$ , para cada  $i = 1, \dots, k$  y para cada punto  $x_j$  se calcula la verosimilitud  $v_j^i$  de  $N(\mu_i, \sigma_i)$  en  $\{x_j\}$ . Con esto se construyen los *clusters* como  $C_i = \{x_j \mid v_j^i = \max_k \{v_j^k\}\}$  para cada  $i = 1, \dots, k$ .

A continuación se detallan los pasos del GMM:

#### Algoritmo 2.1.3.1.4 (GMM)

1. *Escoger el número esperado de clusters,  $k$ .*
2. *Definir la distribución teórica  $F$  como en la fórmula 2.12.*
3. *Aplicar el algoritmo EM sobre la función de verosimilitud de  $F$  en la muestra de datos.*
4. *Calcular la verosimilitud de cada componente respecto de cada muestra unitaria.*
5. *Asociar a cada punto la componente que maximice su verosimilitud.*
6. *Construir los clusters agrupando los puntos que estén asociados a una misma componente.*

Como ocurría con el *k-Means*, este algoritmo requiere que se prefije el número esperado de *clusters*. Sin embargo, el GMM no depende de la elección de la distancia, pues su enfoque es probabilístico. Esto hace que el GMM pueda captar *clusters* con diferentes formas, a diferencia del *k-Means* que, al estar basado en la distancia, tiende a identificar solamente los *clusters* esféricos.

Algunos autores son reticentes a considerar estas metodologías como algoritmos de *clustering* y consideran más adecuado decir que se aplican técnicas de inferencia estadística para segmentar un conjunto de datos. En cualquier caso, este tipo de procesos está muy extendido y es conveniente conocerlos para poder aplicarlos como una herramienta más para segmentar conjuntos de datos.

#### 2.1.3.2. Evaluación de clasificaciones de datos

A medida que ha aumentado el interés en el *clustering* han ido surgiendo formas de medir la calidad de las segmentaciones de datos. Esto permite comprobar si una clasificación es coherente respecto de ciertos criterios de cohesión. Con estas métricas es posible evaluar y comparar segmentaciones de datos, pero también medir la eficacia de los algoritmos expuestos en la sección anterior. Indirectamente, se pueden utilizar para calibrar dichos algoritmos, es decir, escoger el número de *clusters*, los parámetros, las segmentaciones iniciales y los criterios de convergencia de una forma adecuada para obtener un resultado de calidad.

Aunque existen distintas métricas para medir la calidad de una segmentación de datos, estas siempre tratan de cuantificar alguno de los siguientes conceptos (o una combinación de ambos):

- Homogeneidad intragrupo: similitud entre los datos pertenecientes a un mismo *cluster*.
- Heterogeneidad intergrupo: diferencia entre los datos que pertenecen a diferentes *clusters*.

Los ejemplos más sencillos de métricas de segmentación de datos son las sumas de cuadrados *intra-cluster* e *intercluster*, cada una de las cuales trata de cuantificar la homogeneidad intragrupo y la heterogeneidad intergrupo, respectivamente. Dados  $k$  clusters,  $C_1, \dots, C_k$ , tal que para cada  $j = 1, \dots, k$ , el cluster  $C_j$  tiene  $n_j$  datos,  $x_1^j, \dots, x_{n_j}^j$  y un centroide  $c_j$  (media, moda, mediana etc..), se definen las sumas de cuadrados como:

$$SC_{intra} = \sum_{j=1}^k \sum_{i=1}^{n_j} (x_i^j - c_j)^2 \quad (2.13)$$

$$SC_{inter} = \sum_{j_1=1}^k \sum_{j_2=1}^k \sum_{i=1}^{n_{j_1}} (x_i^{j_1} - c_{j_2})^2 \quad (2.14)$$

Se puede observar como la suma de cuadrados *intracluster* suma el cuadrado de la distancia de cada punto al centroide de su *cluster*, cuantificando así la homogeneidad intragrupo. Así, respecto de este criterio, una segmentación es de mayor calidad cuanto menor es su suma de cuadrados *intracluster*. Por el contrario, la suma de cuadrados *intercluster* suma el cuadrado de la distancia de cada punto a cada uno de los centroides de los *clusters* a los que no pertenece, cuantificando la heterogeneidad intergrupo. Por ello, en este caso, la calidad de una segmentación es mayor cuanto mayor sea su suma de cuadrados *intercluster*.

Hay que tener en cuenta que estos criterios valen para evaluar y comparar segmentaciones de un mismo conjunto de datos. En caso de que se quieran comparar segmentaciones de distintos conjuntos de datos, habría que tener cuidado al interpretar las sumas de cuadrados, pues estas pueden aumentar significativamente si el rango de los datos es muy grande o si un conjunto tiene muchos más datos que el otro, sin ello implicar que su segmentación sea mejor o peor. Incluso si estamos comparando dos segmentaciones del mismo conjunto de datos, el hecho de que estas tengan diferente número de *clusters* puede hacer que las sumas de cuadrados no sean comparables.

Una medida un poco más compleja que las anteriores es el índice de Duhn. Esta medida establece un ratio entre la menor distancia *intracluster* y la mayor distancia *intercluster*:

$$IDuhn = \frac{\min_j \{ \sum_{i=1}^{n_j} (x_i^j - c_j)^2 \}}{\max_{j_1, j_2} \{ \sum_{i=1}^{n_{j_1}} (x_i^{j_1} - c_{j_2})^2 \}} \quad (2.15)$$

Esta medida es más completa que las anteriores pues combina las dos nociones de calidad de una segmentación de datos. Además, esta medida no es sensible al número de *clusters* de la segmentación, luego permite comparar dos clasificaciones de datos aun teniendo diferente número de grupos. Sin embargo, sigue siendo sensible al número de datos y a su rango.

A continuación se presenta una medida de la calidad de las segmentaciones de datos más elaborada y que es menos sensible al número de datos y al rango de los mismos. Esta medida se conoce como coeficiente de silueta y será usada durante el caso práctico. Dado un conjunto de  $n$  datos,  $x_1, \dots, x_n$  segmentados en  $k < n$  clusters,  $C_1, \dots, C_k$ , sea  $i = 1, \dots, n$  y  $I = 1, \dots, k$  tal que  $x_i \in C_I$ , se definen los siguientes coeficientes:

$$a(i) = \frac{1}{|C_I| - 1} \sum_{x_j \in C_I \setminus \{x_i\}} d(x_i, x_j) \quad (2.16)$$

$$b(i) = \min \left\{ \frac{1}{|C_J|} \sum_{x_j \in C_J} d(x_i, x_j) \mid J \neq I \right\} \quad (2.17)$$



Con estos coeficientes, se construye el coeficiente de silueta de  $x_i$  como:

$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, & |C_I| > 1 \\ 0, & |C_I| = 1 \end{cases}$$

Es fácil darse cuenta de que  $-1 \leq s(i) \leq 1$ . También es fácil observar que si el punto  $x_i$  está correctamente asignado, entonces  $s(i) > 0$ , pues en este caso se esperaría que  $a(i) < b(i)$ . De hecho, cuanto mejor construido esté el *cluster*  $C_I$ , mayor será la diferencia entre  $b(i)$  y  $a(i)$ , y  $s(i)$  se acercará más a 1. En cambio, si  $s(i) < 0$  significa que  $a(i) > b(i)$  lo que indica que el punto  $x_i$  está mal asignado pues está más cerca de un *cluster* distinto a  $C_I$ . Valores cercanos a 0 de  $s(i)$  indican que la distancia del punto  $x_i$  a su *cluster* es similar a la de otro *cluster*, lo que puede ser síntoma de que se estén solapando varios *clusters*.

Para evaluar la clasificación de forma global se construye el coeficiente de silueta  $s$  como la media de los valores de  $s(i)$ . En este caso se tiene que  $-1 \leq s \leq 1$  independientemente del número de datos, de su rango o del número de *clusters* de la segmentación. Un valor negativo de  $s$  significa que la mayoría de los puntos están mal asignados. Un valor de  $s > 0$  es un indicador de calidad de la segmentación, a nivel local y global, que puede dar mucha información. Aunque este coeficiente de silueta es valioso, estudiar la distribución de los valores  $\{s(i)\}_i$  aumenta la información sobre la calidad de las asignaciones individuales y puede ayudar a analizar y solventar los motivos por los que algunos puntos no se asignan correctamente.

#### 2.1.4. Evaluación de modelos de clasificación

En esta sección se explican algunas herramientas utilizadas en el ámbito de *Machine Learning* para evaluar la eficacia de un modelo de clasificación de datos cuando es conocida la clasificación real de los mismos. Estas medidas se obtienen de comparar esta clasificación original con la clasificación obtenida por el modelo. Es por esto que estas medidas se suelen utilizar para evaluar modelos de clasificación supervisada, tales como árboles de clasificación, regresiones logísticas o redes neuronales, pues en estos casos es obligatorio conocer la clasificación real de los datos. Sin embargo, esto no significa que no se puedan utilizar estas métricas para evaluar modelos de *clustering* (clasificación no supervisada), pero para poder hacerlo, se requiere tener una clasificación de referencia de los datos.

La primera herramienta que se presenta es la matriz de confusión, que permite visualizar en forma de matriz la distribución de los datos en función de las dos clasificaciones, la original y la predicha. Derivadas de esta matriz, se presentan las medidas de *accuracy* y *precision* (existen otras métricas como *recall* o *sensitivity*, *F-scores*, etc., pero en este estudio no serán necesarias).

Dados el conjunto de etiquetas de la clasificación original es  $A = \{A_1, A_2, \dots, A_n\}$  y el conjunto de etiquetas de la clasificación predicha es  $B = \{B_1, B_2, \dots, B_m\}$ , como no se ha impuesto ninguna restricción sobre estas clasificaciones, cualquier relación es posible entre  $A$  y  $B$ :  $A = B$ ,  $A \subset B$ ,  $B \subset A$ ,  $A \cap B \neq \emptyset$  con  $A \not\subset B$  y  $B \not\subset A$ , y  $A \cap B = \emptyset$ . Ahora bien, dado que las etiquetas de  $B$  provienen de un modelo que trata de replicar las etiquetas de  $A$ , tiene sentido renombrar  $B_1, \dots, B_m$  por  $A_{\sigma(1)}, \dots, A_{\sigma(m)}$  donde  $\sigma$  es una variación de  $m$  elementos del conjunto  $\{1, \dots, \max\{n, m\}\}$  que se puede escoger de forma inteligente para que las dos clasificaciones se parezcan lo máximo posible. Así, los conjuntos de etiquetas son  $A = \{A_1, \dots, A_n\}$  y  $B = \{A_{\sigma(1)}, \dots, A_{\sigma(m)}\}$ , luego solo hay tres casos posibles: si  $n = m$ , entonces  $A = B$ ; si  $n > m$ , entonces  $A \supset B$ ; si  $n < m$ , entonces  $A \subset B$ .

En este contexto, se define la matriz de confusión como  $CM \in \mathcal{M}_{n \times m}$  tal que  $CM_{ij}$  representa el número de datos cuyas etiquetas son  $A_i, A_{\sigma(j)}$ . En esta matriz, se dice que un elemento  $CM_{ij}$  es diagonal si  $i = \sigma(j)$ . Estos elementos representan los datos de cada categoría predicha que están correctamente clasificados. En la figura 2.1 se representan ejemplos de matrices de confusión para  $n = 3$  en los casos  $m = 3, 2, 4$ . En estas representaciones aparecen marcados en rojo los elementos diagonales de las matrices.

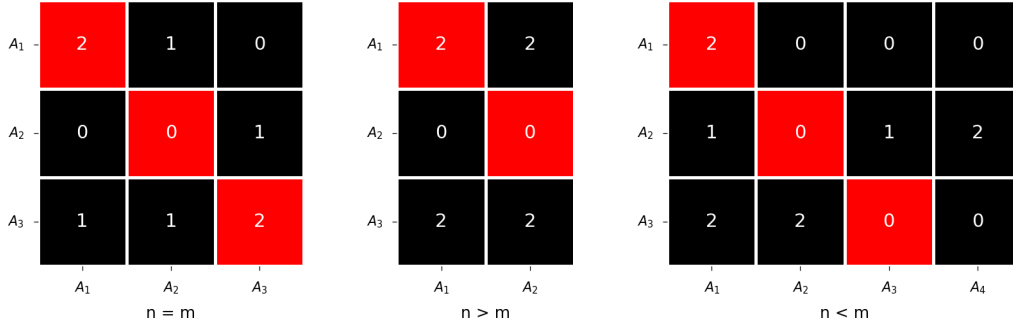


Figura 2.1: Ejemplos de matrices de confusión en función de  $n, m$ .

Una vez introducido el concepto y la notación de matrices de confusión y sus elementos diagonales, podemos introducir las medidas de *accuracy* y *precision* como medidas que evalúan la eficacia de la clasificación predicha respecto de la clasificación original. La medida de *accuracy* mide la eficacia de una clasificación en términos globales. Esto lo hace midiendo la proporción de datos correctamente clasificados de entre todos los datos. Sin embargo, la medida de *precision* es una medida local, pues mide, para cada etiqueta predicha, cuál es la proporción de datos correctamente clasificados.

Si para cada  $i = 1, \dots, n$  y  $j = 1, \dots, m$  definimos  $\delta_{ij} = 1$  si  $i = \sigma(j)$  y  $\delta_{ij} = 0$  en otro caso, se tiene un indicador de los elementos diagonales de la matriz de confusión, y así, se puede calcular las medidas de *accuracy* y *precision* como:

$$Accuracy = \frac{\sum_{i=1}^n \sum_{j=1}^m CM_{ij} \cdot \delta_{ij}}{\sum_{i=1}^n \sum_{j=1}^m CM_{ij}} \quad (2.18)$$

$$Precision(j) = \frac{\sum_{i=1}^n CM_{ij} \cdot \delta_{ij}}{\sum_{i=1}^n CM_{ij}} \quad (2.19)$$

Como ejemplo, aparecen en la tabla 2.3 los valores de *accuracy* y *precision* de las matrices de confusión de la figura 2.1.

	Accuracy	Precision			
		$A_1$	$A_2$	$A_3$	$A_4$
$n = m$	4/8	2/3	0/2	2/3	-
$n > m$	2/8	2/4	0/4	-	-
$n < m$	2/10	2/5	0/2	0/1	0/2

Tabla 2.3: Ejemplos de *accuracy* y *precision*.

Finalmente, se expone un procedimiento para escoger la variación  $\sigma$  del conjunto  $\{1, \dots, \max\{n, m\}\}$ , de forma que las clasificaciones original y predicha sean similares. Para ello, se propone un problema de programación binaria para maximizar el valor de *accuracy*. Este problema trata de escoger  $\max\{n, m\}$  elementos en una matriz de confusión ficticia, de forma que no haya dos en una misma fila ni en una misma columna, para que se maximice su suma. Estos elementos representarán los datos bien clasificados. La variación  $\sigma$  se obtiene de reordenar las columnas de la matriz para que los elementos obtenidos estén en la diagonal.

Este problema se plantea sobre una matriz  $M = (M_{ij})_{i,j} \in \mathcal{M}_{k \times k}$  con  $k = \max\{n, m\}$  que se construye como una matriz de confusión ( $M_{ij}$  representa el número de datos con etiquetas  $A_i, A_j$ ). Esta matriz es una extensión de la matriz de confusión pues se construye de la misma forma y, según el caso, puede tener filas o columnas ficticias (con todos sus valores nulos):

- Si  $n = m$ , entonces  $A = B$  y la matriz no tiene ni filas ni columnas ficticias.
- Si  $n > m$ , entonces  $A \supset B$  y la matriz tiene columnas ficticias (las de las etiquetas de  $A \setminus B$ ).
- Si  $n < m$ , entonces  $A \subset B$  y la matriz tiene filas ficticias (las de las etiquetas de  $B \setminus A$ ).

A continuación se detalla el proceso paso por paso:

#### Algoritmo 2.1.4.0.1 (Optimización de la matriz de confusión)

1. Renombrar las etiquetas de  $B$  sustituyendo  $B_i$  por  $A_i$  para cada  $i = 1, \dots, m$ .
2. Considerar el conjunto  $C = A \cup B = \{A_1, \dots, A_k\}$  con  $k = \max\{n, m\}$ .
3. Construir una matriz  $M \in \mathcal{M}_{k \times k}$  con valores  $M_{ij}$  que representan el número de datos cuyas etiquetas son  $A_i$  y  $A_j$ .
4. Se plantea el siguiente problema de programación binaria:

$$\left\{ \begin{array}{ll} \text{Max} & \sum_{i=1}^k \sum_{j=1}^k M_{ij} \cdot x_{ij} \\ \text{s.a.} & \sum_{j=1}^k x_{ij} = 1, \quad \forall i = 1, \dots, k \\ & \sum_{i=1}^k x_{ij} = 1, \quad \forall j = 1, \dots, k \\ & x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \dots, k \end{array} \right. \quad (2.20)$$

5. Calcular la solución óptima  $x^*$  del problema 2.20.
6. Construir  $\sigma : \{1, \dots, k\} \rightarrow \{1, \dots, k\}$  como  $\sigma(j) = \sum_{i=1}^k x_{ij}^* \cdot i$ , para cada  $j = 1, \dots, k$ .
7. Aplicar  $\sigma|_{\{1, \dots, m\}}$  para renombrar las etiquetas: para cada  $j = 1, \dots, m$ , la etiqueta  $A_j$ , que proviene de  $B_j$ , se sustituye por  $A_{\sigma(j)}$ .

Para poner este procedimiento en práctica, se retoma el ejemplo de la figura 2.1. En este caso, las etiquetas de  $B$  ya están en la forma  $A_j$  con  $j = 1, \dots, m$ . En la figura 2.2 se representan las matrices auxiliares de los tres casos. Se puede observar cómo en el caso en el que  $n > m$  aparece una columna ficticia (que se corresponde con la etiqueta  $A_3$ ) y cómo cuando  $n < m$  aparece una fila ficticia (que se corresponde con la etiqueta  $A_4$ ).

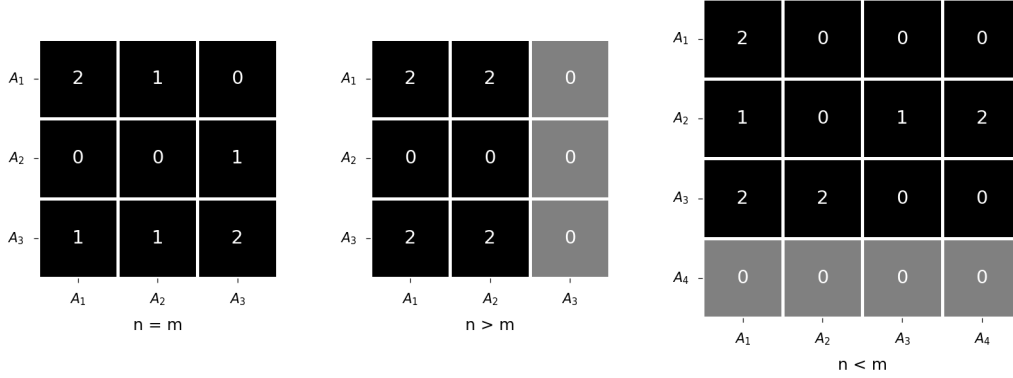


Figura 2.2: Ejemplos de matrices auxiliares en función de  $n, m$ .

Con estas matrices se plantean los problemas de programación binaria, obteniéndose las siguientes soluciones:

- En el caso  $n = m$  se obtiene  $x_{11}^* = x_{22}^* = x_{33}^* = 1$ .
- En el caso  $n > m$  se obtiene  $x_{11}^* = x_{32}^* = x_{23}^* = 1$ .
- En el caso  $n < m$  se obtiene  $x_{11}^* = x_{32}^* = x_{43}^* = x_{24}^* = 1$ .

En la figura 2.3 se representan estas soluciones.

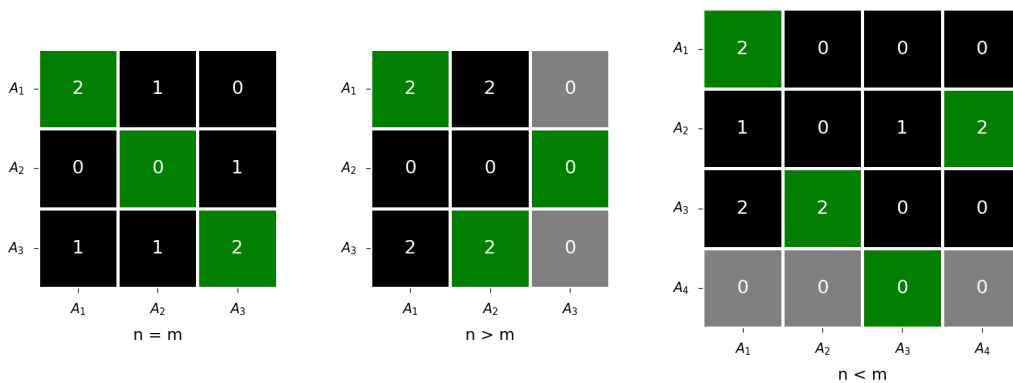


Figura 2.3: Representación de los elementos diagonales óptimos en función de  $n, m$ .

Tras renombrar las etiquetas, eliminar las filas y columnas ficticias, y reordenar las etiquetas, se obtienen las matrices de confusión optimizadas. Estas matrices se representan en la figura 2.4. Se puede observar que en estas matrices optimizadas los elementos diagonales, los que representan el número de elementos clasificados correctamente de cada etiqueta, son los valores más altos de las matrices.

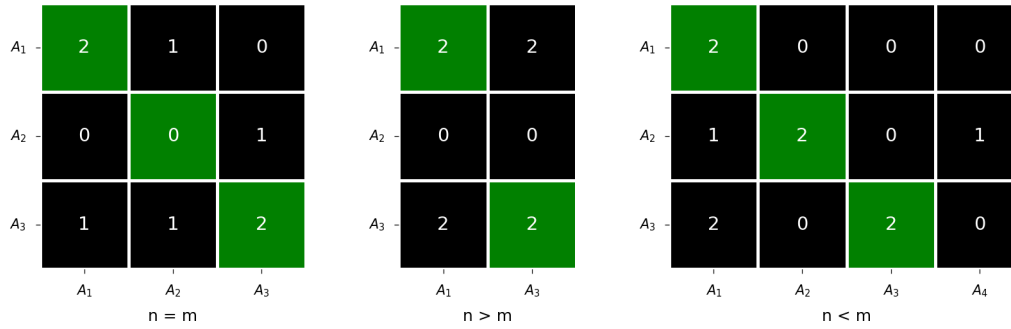


Figura 2.4: Ejemplos de matrices de confusión optimizadas en función de  $n, m$ .

En la tabla 2.4 se presentan los nuevos valores de *accuracy* y *precision*. Se observa que los valores de *accuracy* son mayores o iguales que los de la tabla 2.3. A su vez, algunos de los valores de *precision* han aumentado.

	Accuracy	Precision			
		$A_1$	$A_2$	$A_3$	$A_4$
$n = m$	4/8	2/3	0/2	2/3	-
$n > m$	4/8	2/4	-	2/4	-
$n < m$	6/10	2/5	2/2	2/2	0/1

Tabla 2.4: Ejemplos de *accuracy* y *precision* optimizados.

## 2.2. Metodología para la clasificación de textos por temáticas

Los algoritmos de *Machine Learning* explicados en la sección anterior servirán de componentes para construir una metodología robusta para alcanzar el objetivo del trabajo: comparar la capacidad de PCA y t-SNE para mantener las estructuras geométricas al reducir dimensionalidad en conjunto de datos procedentes de un corpus de textos.

En la figura 2.5 se representa en forma de diagrama una metodología para agrupar textos tras procesarlos, estructurarlos y reducirlos. Esta dinámica comienza con la obtención del corpus de textos, continua con las dos fases de la minería de textos presentadas en 2.1.1 para estructurar dicho corpus para después aplicar un modelo de reducción de dimensionalidad (2.1.2) y finalmente un algoritmo de *clustering* (2.1.3). Esta metodología constituye una herramienta para realizar *Topic Modeling*.

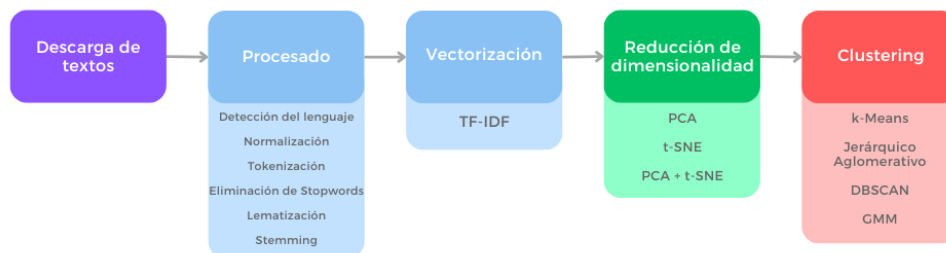


Figura 2.5: Metodología para un ejercicio de *Topic Modeling*.

Cuando se quiere aplicar varios modelos de reducción de dimensionalidad y comparar su desempeño para mantener la estructura de los datos en temáticas, se propone una metodología un poco más sofisticada, que está representada en la figura 2.6. Este proceso consiste en aplicar diferentes modelos de reducción de dimensionalidad al resultado de estructurar un corpus, para después clasificar los resultados con un mismo proceso de *clustering*, evaluar las clasificaciones con un mismo sistema de calidad y comparar los indicadores obtenidos para los diferentes modelos.

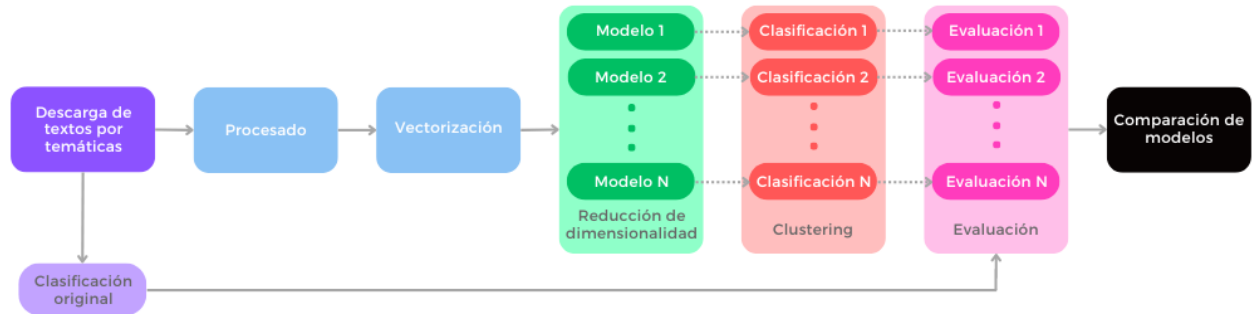


Figura 2.6: Metodología para la comparación de técnicas de reducción de dimensionalidad.

En el siguiente capítulo se utilizará esta metodología para comparar la eficacia de los modelos presentados en 2.1.2 para captar las estructuras escondidas en un corpus de textos estructurado.

### 3. Caso práctico

En este capítulo se presenta un experimento que tiene como objetivo comparar la eficacia de varias técnicas de reducción de dimensionalidad a la hora de mantener la información y las relaciones geométricas de conjuntos de datos procedentes de un corpus de textos. Para ello se va a aplicar la metodología resumida en la figura 2.6.

Los textos que se procesan en este ejercicio son *tweets* (textos cortos de la red social X, antes conocida como *Twitter*). Como la metodología requiere que el corpus tenga una clasificación de referencia, los *tweets* han sido descargados por temáticas que configuran los *clusters* originales.

Las fases que constituyen este ejercicio son las siguientes:

- 1) Descarga del corpus de *tweets* por temáticas.
- 2) Aplicación de procesamiento de lenguaje natural para convertir la información textual no estructurada en una matriz de documentos - términos cuyos valores son tf-idfs.
- 3) Aplicación de los métodos de reducción de dimensionalidad sobre dicha matriz para reducir los datos a dos componentes. Los métodos empleados son PCA, t-SNE y un proceso bietápico consistente en realizar primero un PCA y, sobre el resultado, un t-SNE.
- 4) Evaluación de los modelos de reducción de dimensionalidad bajo diferentes enfoques. Para cada enfoque, la evaluación se realiza en dos pasos:
  - 4.1) Aplicación de métodos de clasificación no supervisada sobre los datos proyectados.
  - 4.2) Extracción de métricas mediante la comparación de las clasificaciones obtenidas con la clasificación original.
- 5) Comparación de los modelos de reducción de dimensionalidad.

A continuación se detallará cada una de estas fases:

#### 3.1. Construcción del corpus

El objetivo de este primer paso es obtener diez archivos de *tweets*, de forma que los *tweets* de cada archivo hablen de una misma temática. Además, se requiere que cada uno de estos archivos contenga al menos 2000 *tweets*, pues así se tiene margen para aplicar filtros en el segundo paso y poder obtener, finalmente, un conjunto de 10000 datos (1000 de cada temática).

Para realizar las descargas se ha hecho uso de la librería *rtweet* de R, en concreto de la función *search\_tweets2* con la que se pueden realizar descargas personalizadas de *tweets* publicados durante la última semana. Con esta función se han realizado diez descargas independientes, de forma que los *tweets* de cada una de estas descargas hablan de una temática de actualidad (a día 22/11/22). Gracias al uso de parámetros como *q*, *n*, *lang*, *result\_type*, *include\_rts* se han personalizado las descargas para realizarlas por *queries* múltiples, para controlar el número de *tweets* a descargar, su idioma, etc.. En la tabla 3.1 se muestran las *queries* utilizadas para descargar los *tweets* de cada una de las categorías.

CLUSTER_REAL	PALABRAS CLAVE DE LA QUERY
1	'Ucrania guerra', 'Ucrania Rusia', 'Putin guerra', 'Putin nuclear', 'Ucrania invasión', 'sanciones Rusia', 'OTAN Rusia', 'Zelenski', 'Ucrania fosas', 'Rusia crímenes guerra', 'bomba nuclear', 'Trump guerra Rusia', 'Rusia movilización', 'tropas Rusia', 'movilización Rusia', 'avance Ucrania', 'Nord Stream'
2	'Qatar', 'mundial Qatar', 'mundial 2022', 'fútbol Qatar', 'homosexualidad Qatar', 'construcción estadio mundial', 'brazalete selección mundial', 'LGTBI Qatar', 'FIFA corrupción', 'polémica Qatar muertos', 'gais Qatar', 'gays Qatar', 'esposas Qatar', 'cárceles Qatar', 'Derechos Humanos mundial', 'Derechos humanos Qatar', 'DDHH Qatar', 'homosexual daño mental Qatar', 'falsos aficionados', '#Qatar2022', 'Qatar cerveza'
3	'coronavirus', 'COVID', 'vacuna', 'Pfizer vacuna', 'AstraZeneca', 'COVID gripe', 'pandemia', 'variantes covid', 'confinamiento', 'mascarillas', 'restricciones covid', 'vacunación', 'primera ola', 'segunda ola', 'tercera ola', 'cuarta ola', 'quinta ola', 'sexta ola', 'residencias COVID'
4	'crisis energética', 'crisis energía', 'corbata Sánchez', 'inflación', 'recesión', 'precio gas', 'IPC', 'precio energía', 'independencia energética', 'dependencia gas', 'energía nuclear crisis', 'UE gas', 'Alemania gas', 'encarecimiento energía', 'encarecimiento gas', 'invierno Europa', 'precio calefacción'
5	'inmigración España', 'menas', 'Melilla BBC', 'valla Melilla', 'inmigrantes ilegales España', 'inmigración Europa', 'pateras', 'Marlaska Melilla', 'inmigrantes Melilla', 'tragedia Melilla', 'Melilla BBC', 'Ceuta Marruecos', 'Melilla Marruecos', 'delitos extranjeros', 'España Marruecos valla', 'ministerio interior Melilla', 'migrantes muertos', 'mafias inmigración', 'Ceuta Melilla', 'inmigración Barcelona', 'inmigrantes España', 'inmigrantes Barcelona', 'inmigrantes Valencia', 'menas Batán', 'inmigrantes Europa', 'inmigración marroquí', 'imágenes tragedia Melilla', 'videos Melilla'
6	'mujeres iraníes', 'mujeres Irán', 'machismo Irán', 'revolución hijab', 'protestas Irán', 'Mahsa Amini', 'política Irán', 'sanciones Irán', 'represión Irán', 'DDHH Irán', '#IranProtests2022', '#IranRevolution2022', 'clérigos Irán', 'condena muerte Irán', 'Derechos Humanos Irán', 'manifestante Irán', 'dictadura Irán', 'velo Irán', 'pena de muerte Irán', 'Islám Irán', 'feministas Irán', 'activistas iraníes', 'mujeres velo', 'gobierno Irán', '#IranRevolution', 'solidaridad Irán', 'turbante Irán', 'líderes religiosos Irán'
7	'LGTBI SEPE', 'Ley trans', 'ley sí es sí', 'justicia machista', 'ministerio igualdad', 'niños trans', 'médicos trans', 'trans deporte femenino', 'hormonación trans', 'registro trans', 'Irene Montero', 'lenguaje inclusivo Irene Montero', 'fascistas con toga', 'cambio de sexo', 'reducción condena ley', 'ideología de género', 'hazte oír', 'rebajas pena sí es sí'
8	'correos comunista', 'sello PCE', 'sello comunista', 'abogados cristianos sello', 'abogados cristianos correos', 'sello partido comunista', 'juez sello PCE', 'centenario PCE', 'centenario comunista', 'correos neutralidad PCE'
9	'sanidad Madrid', 'sanidad pública Ayuso', 'sanidad profesionales huelga', '#SanidadPublica', '#MadridSeLevantaEl13', 'sanitarios Madrid', 'sanitarios Ayuso', 'huelga sanidad', 'ambulatorios Madrid', 'atención primaria Madrid', 'marea blanca Madrid', 'Ayuso sanidad', 'recortes sanidad PP', 'sanidad madrileña'
10	'delito sedición', 'independentismo', 'líderes independentistas', 'reforma sedición', 'castellano aulas Cataluña', 'Cataluña castellano', 'Cataluña referéndum', 'sedición 1-O', 'Cataluña 2017', 'Puigdemont', 'Junqueras', 'delito malversación', 'reforma malversación'

Tabla 3.1: *Querys* utilizadas para las descarga de *tweets*.



## 3.2. Procesado y vectorización

Una vez descargado el conjunto de *tweets* en bruto, el objetivo es obtener un conjunto de datos estructurado al que se le puedan aplicar los métodos de reducción de dimensionalidad. Para hacer esto se intercalan los procesos usuales de NLP con diferentes filtros que limpian el conjunto de datos de registros duplicados y de variables no deseadas. Los pasos de este proceso son los siguientes:

- 1) Eliminación de *tweets* duplicados en cada categoría y entre categorías diferentes.
- 2) Preprocesado de *tweets*: tokenización y normalización de *tweets*, eliminación de *urls*, *hashtags* y menciones de usuarios (*tokens* que comienzan por *https://*, *http://*, *www.*, *#* o *@*), eliminación de caracteres especiales, eliminación de *tokens* con menos de tres caracteres y eliminación de *stopwords*.
- 3) Eliminación de datos cuyo resultado del preprocesado está duplicado.
- 4) Eliminación de *tweets* que no están en castellano (se utiliza un modelo de detección del lenguaje sobre el resultado del preprocesado de los *tweets*).
- 5) Etiquetado y lematización: se realiza un etiquetado morfológico de los *tokens* y se aplica un lematizador a aquellos *tokens* clasificados como adjetivo, sustantivo, verbo o proposición nominal (el resto son descartados). El resultado se normaliza eliminando los acentos gráficos.
- 6) Eliminación de datos cuyo resultado de la lematización está duplicado.
- 7) *Stemming*: se aplica un proceso de *stemming* o poda a los lemas de los *tweets*.
- 8) Eliminación de datos cuyo resultado del *stemming* está duplicado.
- 9) Agrupación de *tokens*: se colapsan aquellos *tokens* (lemas podados) que dan una puntuación de similaridad superior al 0.85 según un modelo de comparación de *strings* y se realizan las traducciones (cambio de *tokens*) en todos los datos.
- 10) Eliminación de datos cuyo resultado de la agrupación de *tokens* está duplicado.
- 11) Eliminación de *tokens* poco frecuentes: se eliminan de los datos aquellos *tokens* que aparecen en menos de tres datos.
- 12) Eliminación de datos cuyo resultado de la eliminación de *tokens* poco frecuentes está duplicado.
- 13) Detección de bigramas: se aplica un modelo de detección de bigramas al conjunto de datos para detectar los pares de *tokens* que aparecen juntos con una frecuencia muy superior a la esperada y en los datos que aparecen dichos pares de *tokens* se sustituyen por el bigrama (la concatenación de los dos *tokens* unidos por "\_").
- 14) Remuestreo: en cada categoría se realiza un remuestreo aleatorio de 1000 datos para así obtener un conjunto de 10000 datos con las categorías balanceadas.
- 15) Vectorización tf-idf: transformación de la lista de 10000 *tweets* procesados en una matriz de documentos - términos con valores td-idfs.

El resultado de este proceso es un conjunto estructurado de 10000 datos en 7527 columnas, al que se le pueden aplicar los modelos de reducción de dimensionalidad. Además, en este conjunto de datos las categorías de *tweets* están balanceadas, lo que facilitará la evaluación de los modelos.

### 3.3. Reducción de dimensionalidad

Una vez obtenida la matriz de documentos - términos, ya es posible aplicar las técnicas de reducción de dimensionalidad, que posteriormente serán evaluadas. Los modelos que vamos a comparar son los desarrollados en 2.1.2: PCA, t-SNE y PCA + t-SNE (modelo combinado).

Se ha llamado modelo combinado a una combinación en dos etapas de PCA y t-SNE. En este proceso se empieza aplicando PCA y se escoge un conjunto grande de componentes  $\{PCA_i \mid i = 1, \dots, k_0\}$ . A este conjunto se le aplica  $k$ -Means con diferentes valores de  $k$  y se comparan los resultados. Finalmente, para obtener el resultado, se aplica t-SNE al conjunto de  $\{PCA_i \mid i = 1, \dots, k^*\}$  donde  $k^*$  es el valor que ha proporcionado la clasificación de más calidad. Concretamente, el esquema de este modelo es el siguiente:

#### Algoritmo 3.3.0.0.1 (PCA + t-SNE)

1. Escoger un número inicial de componentes,  $k_0 < p$ , y dos ratios de varianza explicada,  $V_{min}, V_{max}$  tal que  $0 < V_{min} < V_{max} < 1$ .
2. Aplicar PCA para obtener el conjunto  $\{PCA_i \mid i = 1, \dots, k_0\}$ .
3. Escoger el número de componentes,  $k^*$ , al que es aplicará t-SNE:
  - 3.1. Calcular el ratio de varianza explicada por el conjunto  $\{PCA_j \mid j = 1, \dots, i\}$ ,  $V_i$ , para cada  $i = 1, \dots, k_0$ .
  - 3.2. Calcular el conjunto de candidatos  $C = \{i = 1, \dots, k_0 \mid V_{min} \leq V_i \leq V_{max}\}$ .
  - 3.3. Para cada  $i \in C$  aplicar  $k$ -Means con  $k = i$  al conjunto  $\{PCA_i \mid i = 1, \dots, k_0\}$  y calcular el coeficiente de silueta del resultado,  $s_i$ .
  - 3.4. Tomar  $k^* = \argmax\{s_i \mid i \in C\}$ .
4. Aplicar t-SNE sobre  $\{PCA_i \mid i = 1, \dots, k^*\}$  para obtener  $\{TSNE_1, TSNE_2\}$ .

Para aplicar este modelo bietápico a la matriz de documentos - términos se ha escogido como valor preliminar del número de componentes principales  $k_0 = 500$ , como cota mínima de varianza a explicar  $V_{min} = 0,04$  y como cota máxima  $V_{max} = 0,1$ . Con estos valores se ha obtenido como conjunto de candidatos  $C = \{5, \dots, 25\}$ . En la tabla 3.2 se muestran los coeficientes de silueta obtenidos al aplicar  $k$ -Means con  $k \in C$ . El valor máximo se ha obtenido para  $k^* = 21$ , luego el modelo combinado es en esencia PCA\_21 + TSNE.

k	5	6	7	8	9	10	11
s(k)	0.02473	0.02756	0.02886	0.02767	0.02195	0.02359	0.02020
k	12	13	14	15	16	17	18
s(k)	0.02187	0.02280	0.02390	0.02458	0.02002	0.02189	0.02363
k	19	20	21	22	23	24	25
s(k)	0.02631	0.02817	0.03084	0.02582	0.02533	0.02993	0.03054

Tabla 3.2: Resultados de los modelos para  $k = 10$ .

El resultado de aplicar estos modelos a la matriz de documentos - términos son tres conjuntos de datos proyectados en un espacio bidimensional. Dado que estas proyecciones se corresponden inequívocamente con los *tweets* originales se pueden representar con su *cluster* original. Estas proyecciones se muestran en la figura 3.1.

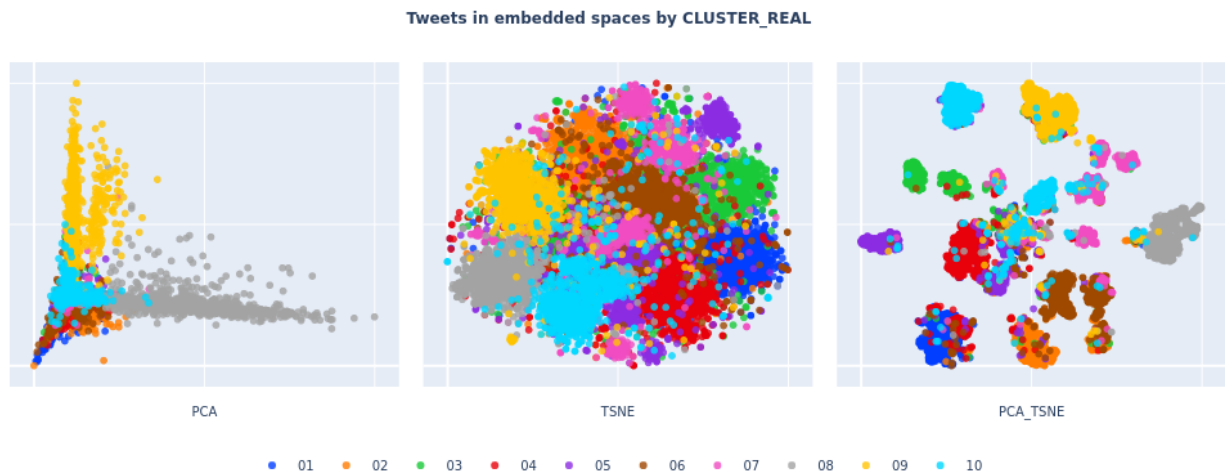


Figura 3.1: Representación de *tweets* en los espacios reducidos.

Es fácil observar que estos conjuntos proyectados son muy diferentes. En el espacio del modelo PCA se observa una nube de puntos con una forma estrellada en la que aparecen algo separados los datos de las categorías 8 y 9, sin embargo, el resto de categorías aparecen mezcladas en torno al origen. En el espacio del modelo t-SNE sí que aparecen segregadas todas las categorías (algunas en varias nubes) pero estas agrupaciones aparecen muy juntas y con mucho ruido entre sí. En el espacio del modelo combinado vuelven a aparecer todas las categorías segregadas, pero esta vez estas agrupaciones sí están separadas. En este caso también se observa ruido en torno al origen que se mezcla con la nubes de datos de la categorías 4, 5 y 10. Estas observaciones parecen corroborar que la eficacia del t-SNE es significativamente superior a la del PCA y que el desempeño de ambos modelos mejora al combinarlos. En el siguiente paso se implementa una metodología para contrastar esta hipótesis.

### 3.4. Clasificación y evaluación

Una vez obtenidos los conjuntos de datos proyectados por los diferentes modelos, ya se puede evaluar el desempeño de estos. Para ello se compara la clasificación original con las clasificaciones obtenidas de aplicar una misma metodología de *clustering* a los tres conjuntos de datos reducidos. De cara a realizar esta comparativa, se plantean tres enfoques:

- Enfoque 1: Se presupone conocido el número real de clases (10). Para cada modelo se aplican los algoritmos *k-Means*, jerárquico aglomerativo y GMM con  $k = 10$  y se escogerá la clasificación que mejor coeficiente de silueta produzca.
- Enfoque 2: Se plantean diferentes valores para  $k$  (hasta un máximo de 30). Para cada modelo se aplican los algoritmos *k-Means*, jerárquico aglomerativo y GMM con  $k = 2, \dots, 30$  y se escogerá la clasificación que mejor coeficiente de silueta produzca.
- Enfoque 3: Se plantea aplicar un algoritmo de segmentación que no requiera prefijar el número de *clusters* o un conjunto de candidatos para este valor. Para cada modelo se aplica el algoritmo DBSCAN, que determinará automáticamente el número de clusters.

Tras obtener, en cada enfoque, las clasificaciones de los modelos, estas se comparan con los *clusters* originales de cara a obtener medidas (*accuracy* y *precision*) de la eficacia de los modelos de reducción de dimensionalidad en cada enfoque. Además se analizarán las matrices de confusión, para estudiar los *clusters* que han sido correctamente detectados, los que han sido mezclados y los que han sido segregados en varios, etc..

En cada enfoque, las medidas son comparables pues han sido obtenidas de una metodología común, y permiten cuantificar y comparar el rendimiento de los modelos. Sin embargo, los resultados de cada enfoque proporcionarían conclusiones diferenciadas. En el caso del enfoque 1, las medidas dan información de lo parecidos que son los *clusters* obtenidos a los de referencia. Sin embargo, en los enfoques 2 y 3, como el número de *clusters* es variable, estas medidas van a cuantificar también si el número de *clusters* obtenidos es parecido al esperado.

Una vez explicados los procedimientos de cada enfoque y las métricas con las que se van a evaluar los modelos, se detalla el desarrollo y los resultados de cada enfoque:

**Enfoque 1:** *k-Means*, jerárquico aglomerativo y GMM con  $k = 10$ :

Como se ha explicado anteriormente, bajo el primer enfoque se supone que  $k = 10$  y se aplican los algoritmos *k-Means*, jerárquico aglomerativo y GMM. En la figura 3.2 se muestran los coeficientes de silueta obtenidos de aplicar estos tres algoritmos a los conjuntos reducidos por los tres modelos de reducción de dimensionalidad.

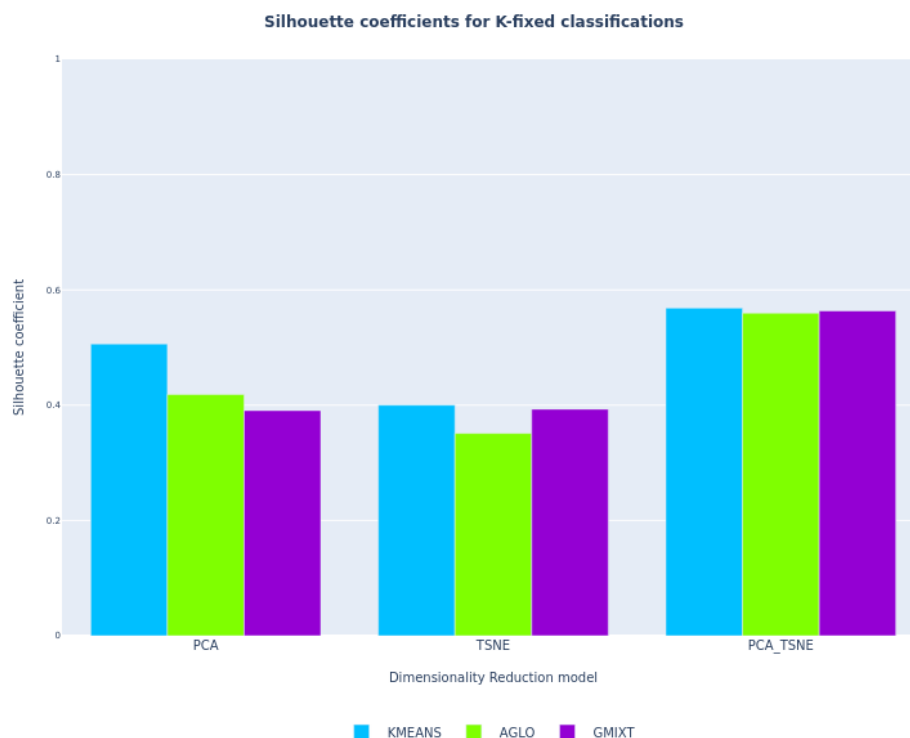


Figura 3.2: Coeficientes de silueta para  $k = 10$ .

Se puede observar que para los tres conjuntos de datos reducidos, el algoritmo *k-Means* es el que produce la clasificación de mayor calidad. Así, las clasificaciones utilizadas para evaluar los modelos son las producidas por este algoritmo. Tras renombrar las etiquetas de estas clasificaciones para maximizar la *accuracy* respecto de la clasificación original se obtienen las clasificaciones finales. En la figura 3.3 se muestran, para los tres modelos, la clasificación original junto a la final.

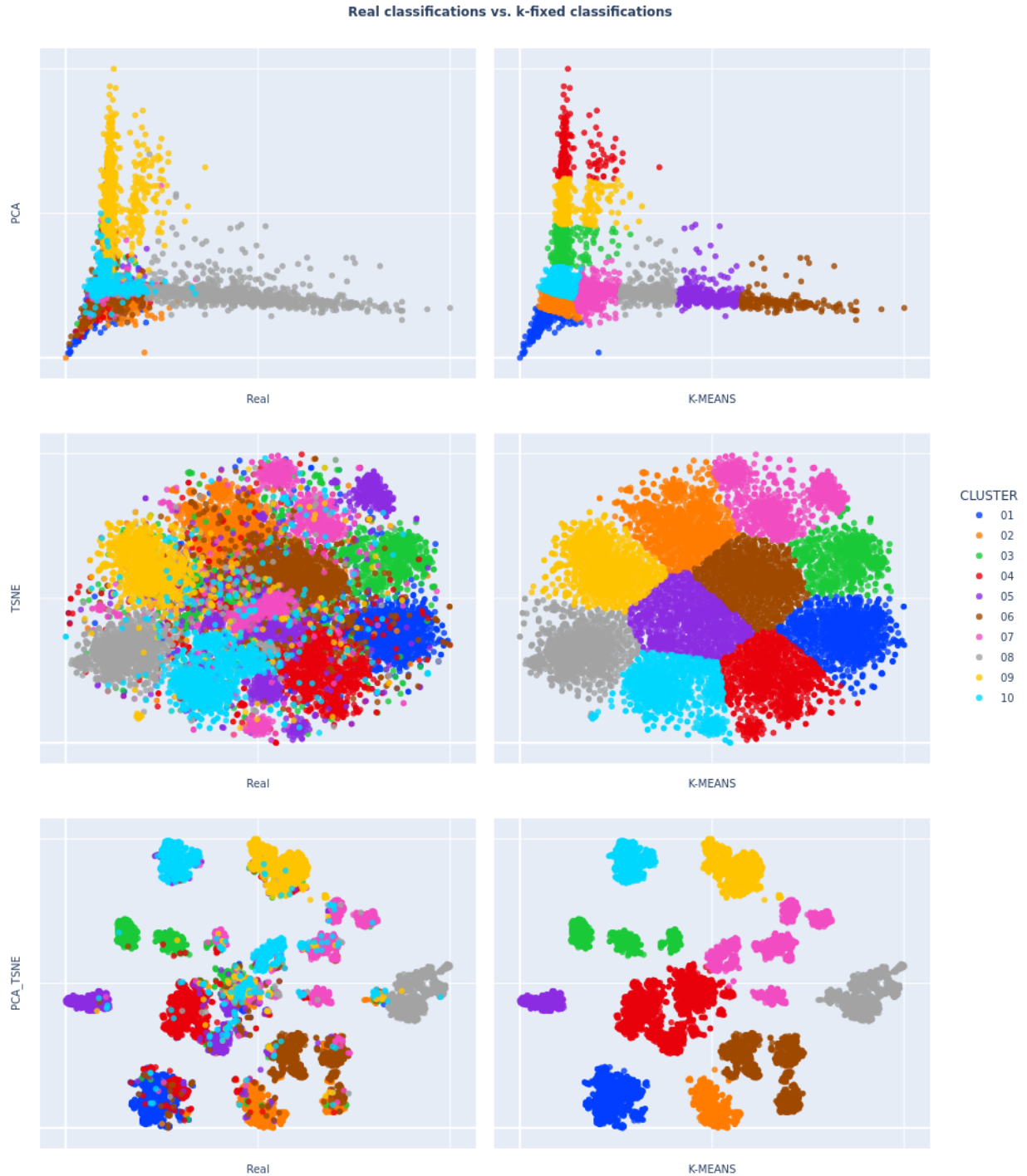


Figura 3.3: Comparación de clasificaciones para  $k = 10$ .

Lo primero que se observa es que las clasificaciones del modelo PCA apenas coinciden, en contraposición con las de los otros modelos. En el modelo t-SNE se observa que la correspondencia es mejor, aunque no para todas las clases. La categoría 5 es la que menos parece corresponderse, aunque esto puede ser porque originalmente esta categoría aparece repartida en varias nubes dispersas. El modelo combinado, por su parte, capta muy bien la estructura de muchas de las clases originales, aunque quizás sobreclasifica *tweets* en el *cluster* 4. Estas observaciones se pueden comprobar con los valores de *accuracy* y *precision* que se muestran en la tabla 3.3.

Modelo	Accuracy	Precision									
		1	2	3	4	5	6	7	8	9	10
PCA	28.52	61.4	30.1	11.3	0.5	0.0	0.0	16.7	97.2	97.3	17.9
t-SNE	67.64	78.7	75.1	85.1	71.4	24.1	66.8	52.2	83.6	80.0	63.0
PCA + t-SNE	73.13	86.5	92.1	82.1	41.5	95.6	68.4	60.8	91.5	90.6	83.7

Tabla 3.3: Resultados de los modelos para  $k = 10$ .

Efectivamente, la *accuracy* de los modelos corrobora que el modelo combinado produce una mejor correspondencia entre las clasificaciones que el t-SNE, que a su vez obtiene mejores resultados que el PCA.

Las medidas de *precision*, por su parte, esclarecen diferentes conclusiones en cada uno de los modelos. En el PCA hay tres categorías con valores altos de *precision*, 1, 8 y 9, que son precisamente las que parecían estar más separadas en la figura 3.1. Sin embargo, el resto de categorías obtienen valores bajos de *precision*. Sin superar ninguna de ellas el 40 %, hay tres categorías que no alcanzan el 1 %. En el modelo t-SNE, los valores de *precision* son más modestos. Sin llegar ninguna categoría al 90 %, todas salvo una superan el 50 %. Es precisamente la categoría 5, que en la figura 3.1 se intuía que no obtenía una correspondencia satisfactoria, la que baja el valor de *precision* hasta un 24,1 %. En cuanto al modelo combinado, se aprecian, por lo general, valores altos de *precision*. Salvo para el *cluster* 4 que tiene un valor de 41,5 %, lo que resulta razonable pues este era en el que se sobreclasifican *tweets*, el resto de valores están por encima del 60 %, llegando en los *clusters* 2, 5, 8 y 9 al 90 % de *precision*.

En la figura 3.4 se muestran las matrices de confusión de los tres modelos para complementar los comentarios de las medidas de *accuracy* y *precision*. Se puede observar que el modelo PCA infraclassifica *tweets* en los *clusters* 1, 8 y 9, lo que implica que sus valores altos de *precision* son menos significativos de una buena eficacia del modelo. Ocurre lo mismo con los valores de *precision* de los *clusters* 2, 3, 5 y 10 del modelo combinado. Sin embargo, los altos valores de *precision* de los *clusters* 8 y 9 en el modelo t-SNE y 1, 6, 8 y 9 en el combinado sí representan la eficacia de los modelos, pues estos *clusters* si agrupan una cantidad de puntos significativa.

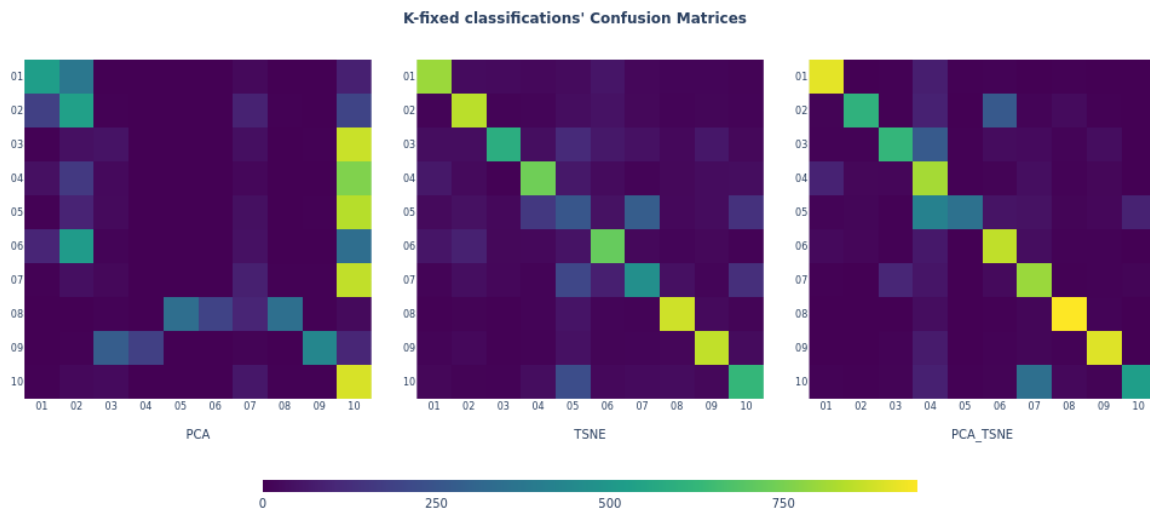


Figura 3.4: Matrices de confusión para  $k = 10$ .

Se puede concluir que, bajo este enfoque, el mejor modelo ha sido el combinado.

**Enfoque 2:**  $k$ -Means, jerárquico aglomerativo y GMM con  $k = 2, \dots, 30$ :

Bajo el segundo enfoque, se varían los valores de  $k$  desde 2 hasta 30 y para cada uno de esos valores se aplican los algoritmos jerárquico aglomerativo,  $k$ -Means y GMM. Una vez aplicados todos los algoritmos con todos los valores de  $k$  a los tres conjuntos proyectados, se escoge, para cada modelo de reducción de dimensionalidad, la mejor clasificación según el criterio del coeficiente de silueta. En la figura 3.5 se muestran los coeficientes de silueta obtenidos.

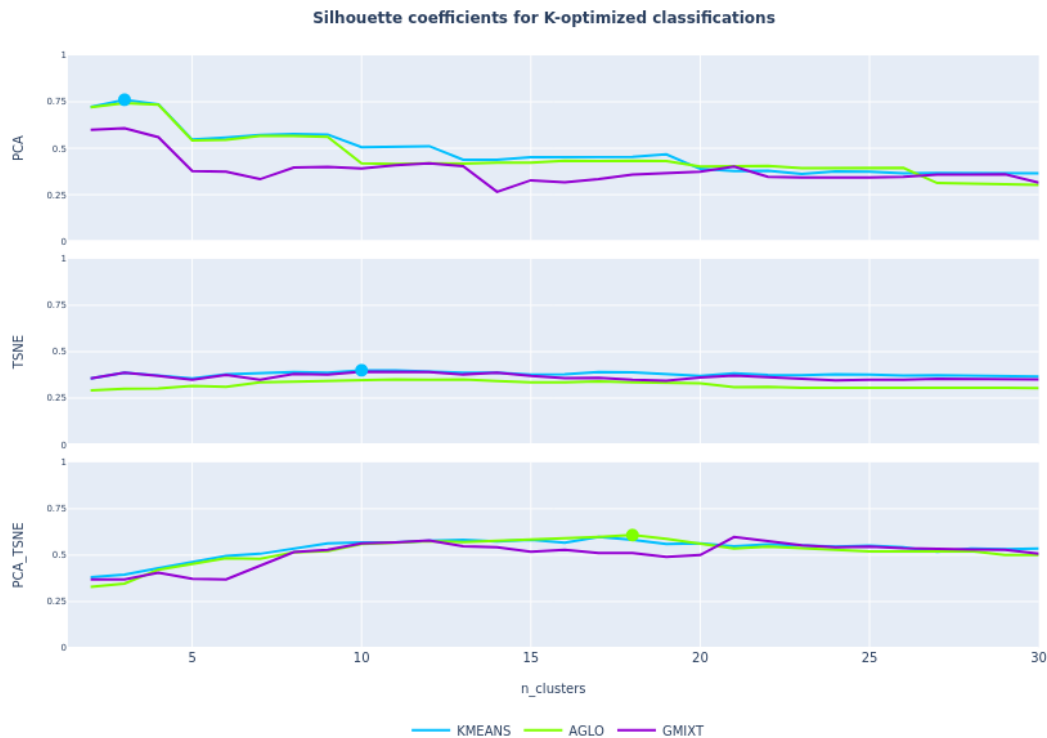


Figura 3.5: Coeficientes de silueta para  $k = 2, \dots, 30$ .

Se puede observar que para el PCA el mejor coeficiente de silueta lo ha proporcionado la clasificación del  $k$ -Means con  $k = 3$ . Esto indica que el conjunto de datos proyectados por el PCA no refleja ningún indicio de que los datos provengan de diez categorías. Esto puede deberse a que en la reducción de dimensionalidad se haya perdido cualquier rastro de la estructura original en diez temáticas de los datos.

En contraposición, los modelos t-SNE y PCA + t-SNE sí parecen haber mantenido una estructura en al menos diez clases de los datos, pues para estos modelos se han obtenido clasificaciones con  $k = 10$  en el caso del t-SNE y  $k = 18$  en el caso del modelo combinado. En el caso del t-SNE esta clasificación ha sido obtenida con  $k$ -Means, mientras que en el modelo combinado se ha obtenido con el *clustering* jerárquico aglomerativo.

Dado que la clasificación del modelo t-SNE se ha obtenido con  $k$ -Means con  $k = 10$ , los resultados de este modelo van a ser los mismos que los del enfoque 1.

Tras renombrar las etiquetas para maximizar la *accuracy* respecto de la clasificación original, se obtienen las clasificaciones finales. En el caso de la clasificación del PCA, como se ha obtenido  $k = 3 < 10$ , este proceso de reetiquetado incluye la elección de las clases que se van a representar por los *clusters* obtenidos. En el caso del modelo combiando, como  $k = 18 > 10$ , en el reetiquetado se eligen los *clusters* que no van a representar a ninguna de las clases originales. En la figura 3.6 se muestran, para los tres modelos, la clasificación original junto a la final.



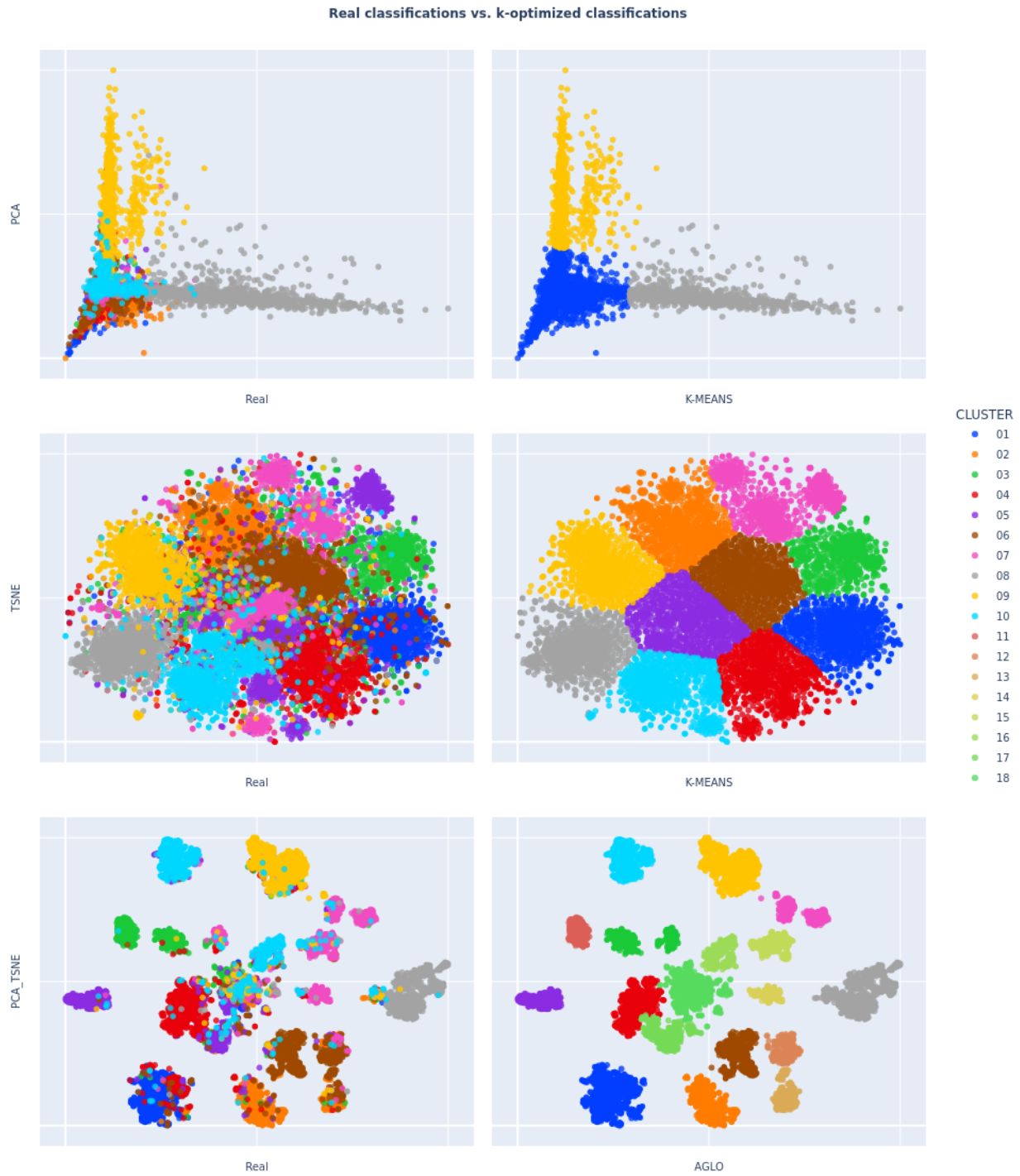


Figura 3.6: Comparación de clasificaciones para  $k = 2, \dots, 30$ .

A pesar del mal desempeño del PCA para mantener la estructura de los datos en diez clases, parece que la clasificación obtenida identifica muy bien los datos de las clase 8 y 9. Sin embargo, junta el resto de clases en el *cluster* 1. En el modelo combinado se obtienen 18 *clusters*. Esto se debe a que la clasificación original tiene varias categorías representadas en varias nubes, que son las que han dado lugar a *clusters* adicionales: la categoría 3 se ha desagregado en los *clusters* 3 y 11, la 5 en los *clusters* 5 y 17, la 6 en los *clusters* 6, 12 y 13, la 7 en los *clusters* 7, 14 y 15 y la 10 en 10 y 16. Este reparto de las categorías originales en los *clusters* obtenidos se puede comprobar observando las matrices de confusión que se muestran en la figura 3.7.



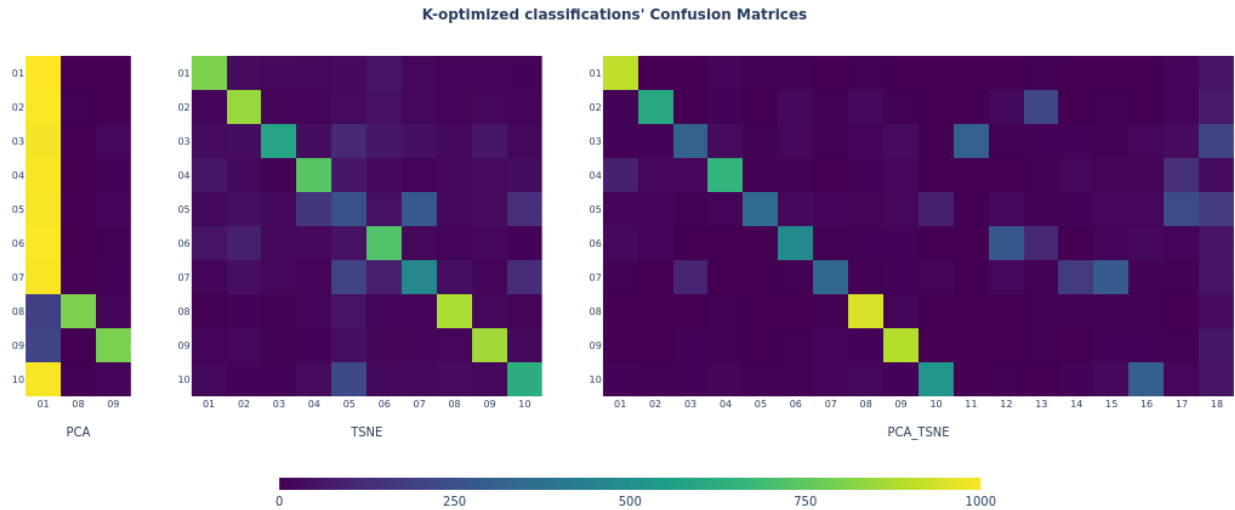


Figura 3.7: Matrices de confusión para  $k = 2, \dots, 30$ .

Para estudiar cómo estos repartos representan la eficacia de los modelos se analizan los valores de *accuracy* y *precision* obtenidos. Estos valores se muestran en la tabla 3.4.

Modelo		PCA	t-SNE	PCA + t-SNE
Accuracy		25.99	67.64	59.83
Precision	1	12.0	78.7	86.5
	2	-	75.1	92.5
	3	-	85.1	70.5
	4	-	71.4	89.6
	5	-	24.1	95.6
	6	-	66.8	84.0
	7	-	52.2	88.7
	8	99.4	83.6	91.5
	9	93.3	80.0	90.5
	10	-	63.0	83.7
	11	-	-	0.0
	12	-	-	0.0
	13	-	-	0.0
	14	-	-	0.0
	15	-	-	0.0
	16	-	-	0.0
	17	-	-	0.0
	18	-	-	0.0

Tabla 3.4: Resultados de los modelos para  $k = 2, \dots, 30$ .

En los modelos PCA y t-SNE los valores de *precision* son bastante altos salvo para uno de los *clusters* (2 en el PCA y 5 en el t-SNE). Esto indica que los *clusters* encontrados están correctamente construidos pero no clasifican todos los puntos, mezclandose todos ellos en un *cluster* restante. En el caso del modelo combinado, este conjunto de puntos sí se ha conseguido separar en *clusters* adicionales. En este caso, los *clusters* principales obtienen valores excelentes de *precision*. Sin embargo, el hecho de haber generado *clusters* originales reduce la *accuracy* llegando a ser peor que la del t-SNE.

En este caso, el modelo que mejores resultados ha obtenido es el t-SNE.

### Enfoque 3: DBSCAN para la detección automática de *clusters*:

Bajo el tercer enfoque, se aplica a cada conjunto de datos proyectados el algoritmo DBSCAN, que no requiere prefiar como parámetro el número de *clusters* a detectar. Así, no se limita el número de *clusters* predichos a un conjunto de candidatos prefijados. Ahora bien, este algoritmo es muy sensible al valor de los parámetros *min\_samples* y *eps*, es por esto que se ha diseñado una metodología basada en el coeficiente de silueta para calibrar sus valores. Para cada modelo de reducción de dimensionalidad se construye un mallado de los parámetros *min\_samples* y *eps* y para cada nodo del mallado se aplica el algoritmo DBSCAN con los parámetros del nodo al conjunto de datos proyectados. Finalmente, se escoge la clasificación con mejor coeficiente de silueta.

Aplicando esta metodología se han obtenido clasificaciones con 1 *cluster* y varios *outliers* para los modelos PCA y t-SNE y una clasificación con 18 *clusters* y también varios *outliers*, para el modelo combinado. Tras renombrar las etiquetas (los *outliers*, marcados con 0, no se renombran) para maximizar los valores de *accuracy* se obtienen las clasificaciones finales. Los resultados en términos de *accuracy* y *precision* se muestran en la tabla 3.5.

Modelo		PCA	t-SNE	PCA + t-SNE
Accuracy		10.0	9.98	60.57
Precision	1	-	-	86.5
	2	-	-	93.0
	3	-	-	98.7
	4	-	-	66.5
	5	-	-	95.6
	6	-	-	84.1
	7	-	-	79.4
	8	-	-	91.5
	9	-	-	90.5
	10	10.0	10.0	83.7
	11	-	-	0.0
	12	-	-	0.0
	13	-	-	0.0
	14	-	-	0.0
	15	-	-	0.0
	16	-	-	0.0
	17	-	-	0.0
	18	-	-	0.0
	0	0.0	0.0	0.0

Tabla 3.5: Resultados de los modelos obtenidos con DBSCAN.

Se puede observar cómo los modelos PCA y t-SNE han obtenido resultados significativamente peores. Esto se debe a que el algoritmo DBSCAN separa en *clusters* diferentes cuando encuentra espacios sin puntos entre nubes de alta densidad. Dado que en las proyecciones de PCA y t-SNE los puntos han quedado apiñados en una sola nube, el algoritmo no ha sido capaz de obtener más de un *cluster*. Como ocurría con el PCA en el primer enfoque, esto puede ser un síntoma de que estos modelos no hayan mantenido la estructura global de los datos al reducir dimensionalidad. Por su parte, el modelo combinado a obtenido una clasificación con 18 *clusters*, muy parecida a la del enfoque 2, aunque con algunas diferencias. A parte de que algunas nubes de puntos han sido clasificadas de forma diferente, en este caso se han obtenido algunos *outliers*.

Para observar explícitamente esta situación, se muestra en la figura 3.6 para los tres modelos, la clasificación original junto a la final.

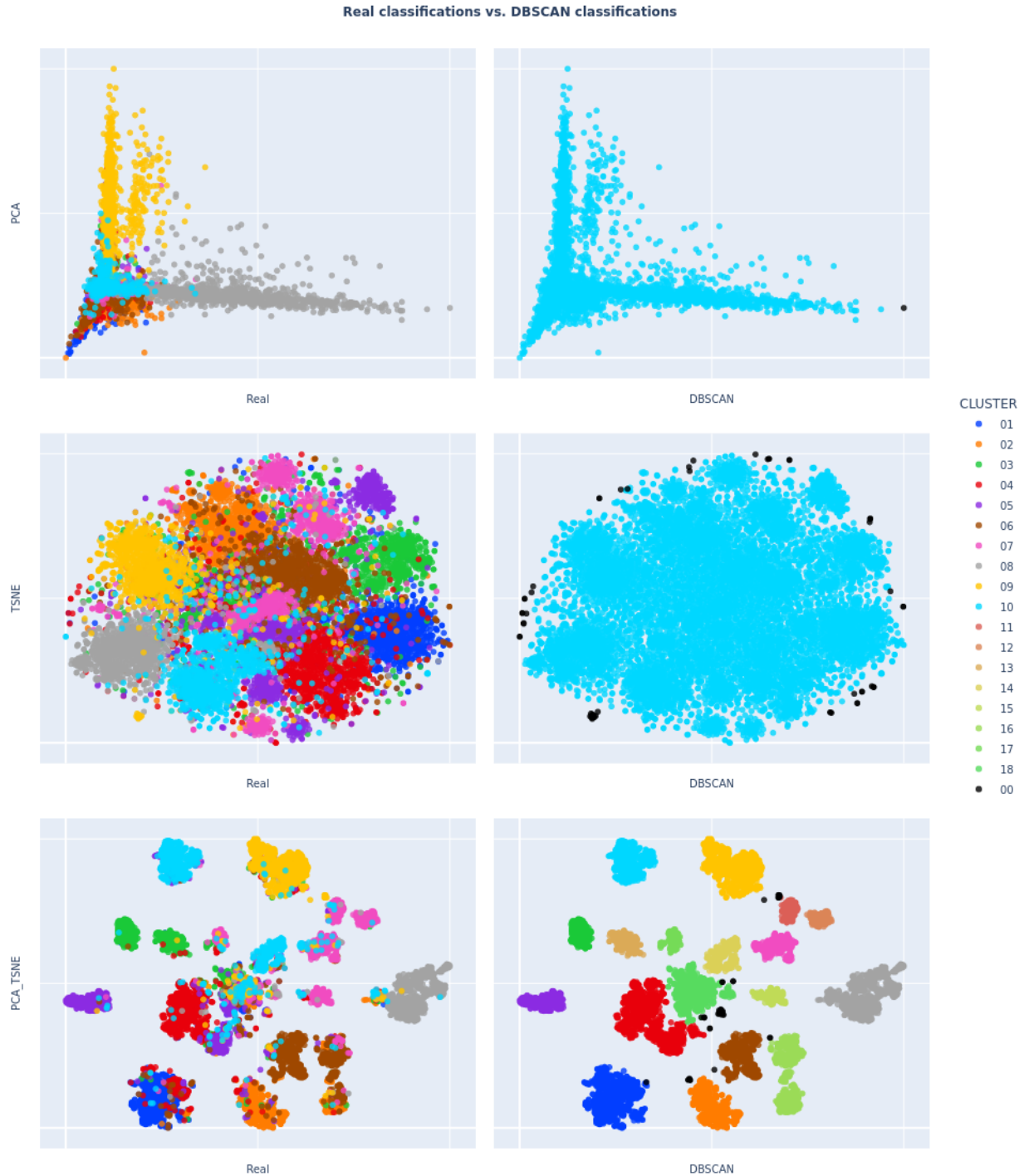


Figura 3.8: Comparación de clasificaciones obtenidas con DBSCAN.

En la figura 3.9 se muestran las matrices de confusión de los tres modelos para complementar los comentarios de las medidas de *accuracy* y *precision*. Se puede observar como los puntos del PCA y t-SNE han quedado agrupados en un solo *cluster*. Por su parte el modelo combinado ha obtenido una clasificación parecido a la del enfoque 2, aunque con un reparto ligeramente diferente: la categoría 3 se ha segregado en los *clusters* 3, 13 y 18, la 5 en los *clusters* 4, 5 y 18, la 7 en los *clusters* 7, 11, 12, 15 y 17 y la 10 en 10 y 16.

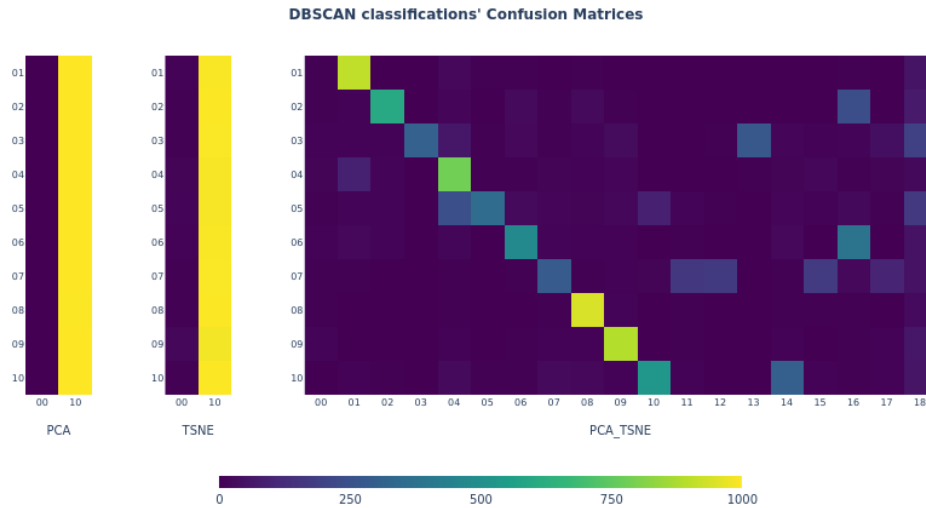


Figura 3.9: Matrices de confusión obtenidas con DBSCAN.

Bajo este enfoque, es claro que el modelo de reducción de dimensionalidad que mejores resultados ha conseguido es el combinado.

### 3.5. Comparación

Gracias a las metodologías de clasificación y extracción de métricas se ha podido evaluar el rendimiento de los modelos bajo diferentes enfoques. Esto sirve para contrastar las hipótesis presentadas al presentar la figura 3.1, cuando se aplicaron los modelos para proyectar los datos:

- Analizando la proyección del PCA se observa que las categorías 8 y 9 son las más discriminantes, o equivalentemente, un *tweet* es más fácil de clasificar si corresponde a una de estas dos categorías. Esto se ha comprobado con los tres enfoques, ya que en todos los modelos estos *clusters* han obtenido un alto valor de *precision*. El resto de categorías han quedado unidas en la proyección del PCA, haciendo que los *clusters* correspondientes obtengan valores de *precision* bajos o que no hayan sido ni detectados.
- Analizando la proyección del t-SNE se observan nubes diferenciadas para cada una de las categorías. Las categorías 5 y 7 están presentes en varias nubes diferenciadas y separadas lo que indica que pueden tener *tweets* de subtemáticas suficientemente diferentes como para aparecer segregadas. Esto es algo que se confirma analizando la proyección del modelo combinado, aunque en este caso, las nubes de la categoría 7 aparecen en una misma zona.
- Analizando la proyección del modelo combinado observamos que las categorías 3 y 6, además de la 5 y la 7, aparecen repartidas en varias nubes, aunque estas son adyacentes. El hecho de que estas categorías aparezcan en varias nubes ha hecho que el modelo combinado obtenga métricas peores que el modelo t-SNE que las presentaba en una sola nube. En este caso, el hecho de captar más relaciones intra-*cluster* ha empeorado las métricas del modelo combinado, lo que no se puede traducir directamente como una pérdida de rendimiento.

Dados los resultados obtenidos, es claro que la combinación del PCA y el t-SNE para este conjunto de datos y con este objetivo en concreto ha supuesto un refinamiento a la hora de trasladar la geometría, tanto local como global, del corpus de *tweets* a la proyección bidimensional.

## 4. Conclusiones

El objetivo principal de este trabajo ha sido comparar la eficacia de las técnicas de reducción de dimensionalidad PCA y t-SNE en el mantenimiento de la estructura de datos en un conjunto de textos procesados mediante técnicas de NLP. Para llevar a cabo este estudio, se han aplicado tres modelos de reducción de dimensionalidad: PCA, t-SNE y un modelo combinado, PCA + t-SNE. La evaluación y comparación de los resultados se ha realizado mediante la obtención de una clasificación de referencia de los datos y la aplicación de algoritmos de *clustering* a los datos proyectados para contrastarse con los *clusters* de referencia.

Los resultados obtenidos indican que el PCA no es capaz de mantener la estructura en las temáticas de un corpus textual. En contraposición, el algoritmo t-SNE, cuando es calibrado con cuidado, sí es capaz de mantener dichas relaciones geométricas que caracterizan las temáticas dentro del corpus. También hemos comprobado que al aplicar una simplificación previa de la geometría de los datos con PCA, los resultados del t-SNE mejoran sustancialmente en la tarea de mantener la estructura local y global de los datos. Esto puede ser un indicio de la existencia de ruido en los corpus textuales que nubla las verdaderas relaciones geométricas que configuran su estructura en temáticas.

Gracias al experimento realizado y a estas conclusiones tan nutritivas, se ha podido comprobar el potencial del algoritmo t-SNE, ya sea solo o en combinación con otras técnicas, para detectar patrones en un corpus de textos. Podemos interpretar los resultados obtenidos como una prueba de que la metodología presentada en la figura 2.5, con una buena elección y calibración de los modelos, puede suponer una posibilidad muy atractiva para abordar el reto del *Topic Modeling*.

Esta comparativa ha permitido identificar un considerable potencial en ciertos algoritmos y metodologías dentro del ámbito de la ciencia de datos. No obstante, esto no implica que todas las cuestiones relativas a la comparación entre PCA y t-SNE hayan sido resueltas, ni que no persistan interrogantes sobre estas técnicas en el campo de la reducción de dimensionalidad. Dadas las limitaciones de tiempo y recursos de este trabajo, el alcance ha sido necesariamente restringido. Para profundizar en esta área y ampliar los hallazgos del estudio, se proponen diversas líneas de investigación:

- Exploración de otros procesos y/o modelos de NLP para procesar el corpus.
- Exploración de otros procesos de estructuración de textos (word2vec, GloVe, BERT, etc.).
- Inclusión de otros corpus textuales de diferente tipos (diferentes tamaños, temáticas, etc.).
- Optimización de los procesos de calibración.
- Concatenación de varios procesos de t-SNE con diferentes calibraciones.
- Implementación del t-SNE en combinación con otras técnicas que no sean PCA.









En conclusión, este estudio ha proporcionado una visión comparativa de la eficacia de PCA y t-SNE en el contexto de datos textuales procesados mediante NLP, destacando la ventaja de una combinación de ambas técnicas. Las sugerencias propuestas para futuros trabajos abren la puerta a una investigación más profunda y detallada, que puede llevar a desarrollos significativos en la representación y análisis de datos textuales.

## 5. Repositorio

El código implementado para realizar el experimento de este estudio se puede consultar en el siguiente enlace:

[https://github.com/carlosrod17/TFM\\_UnEstudioComparativodePCAyt-SNEenTextosProcesadosconNLP](https://github.com/carlosrod17/TFM_UnEstudioComparativodePCAyt-SNEenTextosProcesadosconNLP)














Este repositorio tiene la siguiente estructura:

- ✓  caso\_practico
  - >  code
  - >  data
  - >  logs
-  .gitignore
-  executions.sh
-  requirements.txt
-  setup.sh

Los ficheros *setup.sh* y *requirements.txt* son necesarios para recrear el entorno de ejecución necesario dentro de la imagen de python3.11.8. En particular, *requirements.txt* contiene la información de las librerías necesarias y sus versiones. El script *setup.sh* es un ejecutable de *bash* que instala las librerías en las versiones especificadas y descarga modelos auxiliares de NLP en alguna de esas librerías.

El script *executions.sh* es un orquestador de *bash* con el que se realiza una ejecución de los códigos de python de la carpeta *code* y se guarda un archivo de *logging* de la ejecución en la carpeta *log*.










La carpeta *code* tiene el siguiente contenido:

- ✓  code
  - >  \_\_pycache\_\_
  -  \_\_init\_\_.py
  -  0\_download\_tweets.R
  -  1\_PROCESSING.py
  -  2\_DIMENSIONALITY\_REDUCTION.py
  -  3\_1\_CLUSTERING\_MODELS\_WITH\_K\_FIXED.py
  -  3\_2\_CLUSTERING\_MODELS\_WITH\_K\_OPTIMIZED.py
  -  3\_3\_DBSCAN\_MODELS.py
  -  4\_1\_GET\_INTERACTIVE\_FIGURES.py
  -  4\_2\_GET\_TFM\_FIGURES.py
  -  functions.py
  -  var\_def.py

El primer script *0\_download\_tweets.R* es el que se usó el día 22/11/2022 para realizar la descarga de *tweets* por temáticas.

El archivo *var\_def.py* contiene variable de configuración, parámetros y constantes de la ejecución. El archivo *funcions.py* contiene funciones auxiliares. El resto son *scripts* de *python* que implementan los diferentes pasos del experimento. El *script 1\_PROCESSING.py* realiza la limpieza y estructuración del corpus de *tweets*, el *script 2\_DIMENSIONALITY\_REDUCTION.py* aplica los modelos de reducción de dimensionalidad para obtener los conjuntos proyectados, los *scripts 3\_1\_CLUSTERING\_MODELS\_WITH\_K\_FIXED.py*, *3\_2\_CLUSTERING\_MODELS\_WITH\_K\_OPTIMIZED.py* y *3\_3\_DBSCAN\_MODELS.py* clasifican los conjuntos proyectados y evalúan los modelos bajos los tres enfoques y los *scripts 4\_1\_GET\_INTERACTIVE\_FIGURES.py* y *4\_2\_GET\_TFM\_FIGURES.py* extraen las gráficas.

La carpeta *data* tiene la siguiente estructura:

- ✓  data
  - >  auxiliar
  - >  figures
  - >  figures\_tfm
- ✓  processed
  - >  1\_TWEETS\_RAW\_BY\_CLUSTER
  - >  2\_TWEETS\_PROCESSED
  - >  3\_TWEETS\_EMBEDDED
  - >  4\_RESULTS

En la carpeta auxiliar hay archivos de datos auxiliares, como el conjunto de *stopwords* utilizado. En las carpetas *figures* y *figures\_tfm* se han almacenado las gráficas. En la carpeta *processed* se han guardado los resultados de cada paso del experimento (*tweets* en crudo, *tweets* procesados, *tweets* proyectados y *tweets* clasificados y métricas finales).

# Bibliografía

- [1] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [2] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 1(10):e2, 2016.
- [3] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545–552, 2013.
- [4] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.
- [5] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15, 2002.
- [6] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.
- [7] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [8] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 421–436. Springer, 2012.
- [9] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [11] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The journal of machine learning research*, 15(1):3221–3245, 2014.
- [12] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [13] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.
- [14] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002.
- [15] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.
- [16] Godfrey N Lance and William Thomas Williams. A general theory of classificatory sorting strategies: 1. hierarchical systems. *The computer journal*, 9(4):373–380, 1967.



- [17] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.
- [18] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.
- [19] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.
- [20] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2:169–194, 1998.
- [21] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.