IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Fecha: Mayo 11, 2023

Reducción de Dimensionalidad y Algoritmos de Clustering

Roberto de J. Alfaro López[1]
[1]Universidad Autónoma de Querétaro, MX

Correspondiente al autor: Roberto de J. Alfaro López (e-mail: robertoalfaro14@hotmail.com).

RESUMEN El presente reporte muestra el desarrollo de algunos algoritmos de clustering y reducción de dimensionalidad, escritos sin usar algún framework o biblioteca externa más que los indispensables para operaciones matemáticas. Además, se presentan las técnicas usadas para el preprocesamiento de una base de datos y un breve análisis de este, usando los algoritmos antes descritos. Las funciones se escribieron usando solamente las dependencias más básicas como Numpy y Pandas. Los algoritmos de clustering desarrollados fueron Kmeans y Agglomerative, mientras que para reducción de dimensionalidad se eligió PCA (Principal Component Analysis). Finalmente se hace un breve análisis de los resultados obtenidos al usar clustering después de aplicar PCA y también sin haberlo hecho.

PALABRAS CLAVE Normalización, Imputación, PCA, Clustering Kmeans, Clustering Aglomerativo.

I. OBJETIVO

Utilizar las técnicas de preprocesamiento que se consideren más adecuadas para un dataset, tales como normalización e imputación, y usar algoritmos de clustering y reducción de dimensionalidad para analizarlo.

II. INTRODUCCIÓN

El manejo de datos ha sido una tarea que (consciente o inconscientemente) le ha permitido al hombre, no solo sobrevivir hasta nuestros días, sino también, realizar grandes avances científicos y tecnológicos. Actualmente, la manipulación y el análisis de datos es aún más relevante, pues gracias a la era digital la cantidad de información que se genera es exorbitante en cualquier área, desde biología y medicina, hasta las redes sociales y publicidad. Es por esto que se ha vuelto indispensable saber manejar y examinar correctamente esta información, con la esperanza de encontrar patrones y relaciones que puedan ser de utilidad para ciertos trabajos y actividades.

A pesar de que la generación de muchos datos a gran velocidad, tiene varios aspectos positivos; en el apartado técnico, esto puede generar complicaciones, pues se llegan a crear bases de datos de alta dimensionalidad, cuyo análisis e interpretación se convierte en un gran desafío. Es aquí donde entran en juego técnicas como los algoritmos de clustering y de reducción de dimensionalidad, los cuales ayudan a afrontar estas complicaciones y permiten aprovechar al máximo el potencial de estos grandes volúmenes de datos.

De forma muy general, los algoritmos de clustering son técnicas que permiten agrupar conjuntos de datos en base a su similitud. Son útiles para identificar patrones subyacentes en los datos y proporcionar una visión intuitiva de los mismos.

Por otro lado, los algoritmos de reducción de dimensionalidad, permiten simplificar los datos de alta dimensionalidad en una representación de menor dimensión, sin perder su estructura ni sus propiedades esenciales. Estas técnicas facilitan la visualización y el análisis de los datos, mejoran la eficiencia computacional y ayudan a eliminar ciertos problemas en los cálculos.

De esta manera se puede observar la importancia no sólo del análisis de datos, sino del conocimiento de las técnicas usadas para extraer información de ellos, ya que estas permiten aprovechar e interpretar de mejor manera (y más sencilla) lo que contienen.

III. MARCO TEÓRICO

Se ha logrado vislumbrar hasta el momento la gran relevancia que tienen las técnicas de clustering y de reducción de dimensionalidad en el Machine Learning. Por esto se ha creído necesario en esta sección, dar un breve repaso por ambos tipos de técnicas. A continuación, se verá en qué consiste cada una, cuáles tipos de algoritmos existen dentro de cada método y finalmente cuáles son sus principales aplicaciones

A. Clustering

También llamados algoritmos de agrupamiento, son técnicas de aprendizaje no supervisado que se utilizan para agrupar elementos similares en conjuntos o 'clusters'. El objetivo de esto es que los objetos pertenecientes a un mismo conjunto sean más parecidos entre sí que los pertenecientes a otros conjuntos. Existen muchos algoritmos de clustering y muchas formas de clasificarlos, por ejemplo, se pueden dividir según su técnica de agrupamiento como:

a) Clustering Jerárquico: Crean una jerarquía de clusters que pueden ser representados en un dendrograma. Los dos tipos principales de este método son el aglomerativo y el divisivo.
b) Clustering Basado en Centroides: En este tipo de algoritmos, los clusters se representan por un punto central o centroide. Un ejemplo clásico es K-means.
c) Clustering Basado en Densidad: Estos tipos de algoritmos definen los clusters como áreas de alta densidad separadas por áreas de baja densidad. DBSCAN es un algoritmo común de este tipo.

Como se mencionó, puede haber muchas formas más para clasificar las diversas técnicas de clustering, pero esta suele ser una de las maneras más sencillas y comunes de hacerlo. Por ultimo, resultará útil para el lector saber en qué casos se suelen aplicar estos algoritmos, y es que realmente tienen un gran campo de aplicación, siendo los más relevantes la recomendación de productos, la agrupación de documentos, la bioinformática, la detección de anomalías, la segmentación de mercado, entre otras.

B. Reducción de Dimensionalidad

La reducción de dimensionalidad, es un proceso de transformación de una base de datos, en la que el objetivo es representar la misma información conservando su estructura y propiedades esenciales, pero en menores dimensiones. Esto otorga varias ventajas a los analistas de datos, pues permite mejorar la eficiencia computacional de los algoritmos de aprendizaje automático, facilita la visualización de los datos, ayuda a eliminar el ruido y puede hacer que ciertos algoritmos sean más eficaces al eliminar las dimensiones irrelevantes y reduciendo el sobreajuste. Igualmente existen varios métodos destacando entre ellos:

a) Análisis de Componentes Principales (PCA): Busca las direcciones (componentes principales) en las que los datos varían más. Luego proyecta los datos en estas direcciones para reducir la dimensionalidad manteniendo la máxima varianza posible.
b) Análisis Discriminante Lineal (LDA): Es una técnica de reducción de dimensionalidad, que a su vez es un clasficador lineal. A diferencia de PCA, LDA busca las componentes que maximizan la separación entre múltiples clases.

Igualmente hay que decir que existen muchas más técnicas de reducción de dimensionalidad. En el presente trabajo se usará PCA.

IV. MATERIALES Y MÉTODOS

En esta sección se repasará brevemente el preprocesamiento de los datos que se llevo a cabo, antes de la aplicación de los algoritmos de clustering y PCA.

El dataset que se utilizó tiene por nombre "Spotify-Youtube", y contiene las 10 canciones más populares de diversos artistas en las plataformas Youtube y Spotify. La tabla posee 28 atributos y 20718 instancias. Los datos de los atributos corresponden a variables tanto musicales (como tonalidad, tempo, volumen (dB), energía, acusticidad, etc.) como descriptivas (artista, álbum, link, descripción en YouTube, entre otros).

A. Limpieza de Datos

El primer paso para realizar esta actividad, fue juzgar con qué atributos se trabajaría y visualizar si habían instancias con datos faltantes.

Se debe recordar que el objetivo de esta actividad era aplicar reducción de dimensionalidad y algoritmos de clustering a una base de datos, por lo que se procedió a eliminar aquellos atributos no numéricos y que proporcionaban información que no se consideraba relevante (como el enlace al video de YouTube), más específicamente, los atributos que se eliminaron fueron los siguientes:

1. *Unnamed 0:* Índice original del dataframe.
2. *Album*: Álbum de la canción.
3. *Album_type*: Si la canción se lanzó en Spotify dentro de un álbum o como 'single'.
4. *Key*: Tonalidad de la canción.
5. *Url_spotify*: Link de la canción en spotify.
6. *Uri*: Link de la canción para la API de spotify.
7. *Url_youtube*: Link del video musical en youtube.
8. *Title*: Título del video musical.
9. *Channel*: Canal de youtube del vídeo.
10. *Description*: Descripción del video en youtube.
11. *Licensed*: Si el video representa contenido con licencia.
12. *official_video*: Si el video es o no oficial.

Con respecto a la cantidad de valores faltantes, se encontraron que los atributos *Views*, *Likes*, *Comments*, *Streams*, son los que carecían de algunos datos, aunque todos los demás atributos coincidían en que había dos valores NaN en su información. Se analizó esto último y se encontró que había dos instancias, correspondientes a los artistas 'Natasha Bedingfield' y 'White Noise for Babies', cuyos atributos estaban casi todos en NaN, por lo que al ser tan pocas y aportar tan poca información a comparación de las 20716 instancias restantes, se procedió a eliminarlas. Con respecto a los demás atributos e instancias se procedió a realizar imputación, se probó con diversos métodos, como imputación por media, moda, etc. Aunque esto se explica de manera más detallada a continuación.

B. Imputación y Normalización

Es bien sabido que dependiendo de la base de datos con la que se trabaje son las técnicas que se usarán, esto en ocasiones conlleva probar varios métodos hasta encontrar el que mejor se adecúe al objetivo. En el caso de la imputación para este dataset, se probó con los métodos de imputación por media, imputación aleatoria e imputación por moda. Los resultados se pueden observar en las siguientes gráficas:

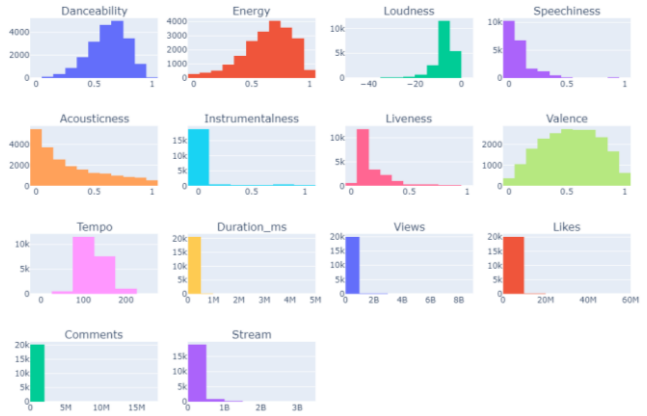

FIGURA 1. Distibución original de los datos de cada atributo.

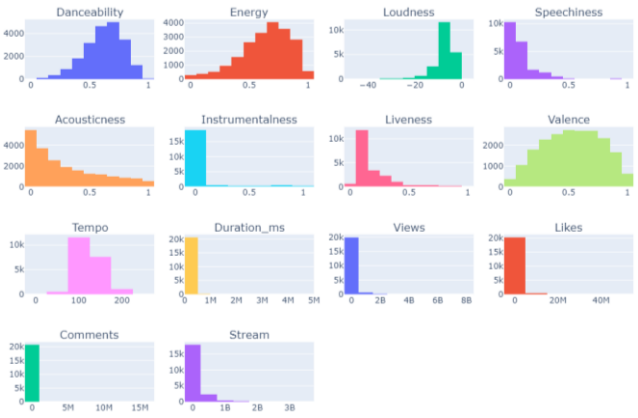

FIGURA 2. Distribución de los datos al aplicar imputación por media.

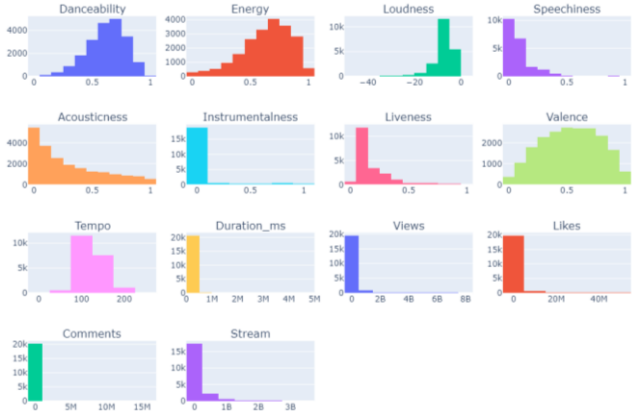

FIGURA 3. Distribución de los datos al aplicar imputación aleatoria.

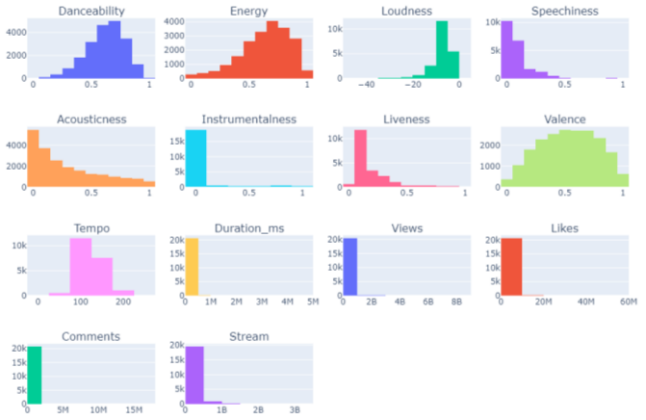

FIGURA 4. Distribución de los datos al aplicar imputación por moda.

Recordar que los atributos que se imputaron fueron *Views*, *Likes*, *Comments*, *Stream*. Se puede observar que, si bien los cambios de un método a otro son mínimos, realmente el que mejor funciona es la imputación por moda, esto se nota más claramente en el atributo *Stream*, pues los otros dos métodos aumentaban ligeramente las barras inferiores de la distribución, mientras que en la imputación por moda quedan casi idénticas.

Por último, se aplicó normalización a los datos con el fin de colocar todos en una escala parecida y afectar lo menos posible a los algoritmos de clustering que se usarían. El método usado fue la normalización min-max, y los valores se escalaron entre 0 y 1 para los atributos *Loudness*, *Tempo*, *Duration_ms*, *Views*, *Likes*, *Comments*, *Stream*; los demás atributos ya se encontraban en este rango de valores.

V. PSEUDOCÓDIGO

Como ya se dijo en secciones anteriores, los algoritmos utilizados fueron: para clustering K-means y Agglomerative, y para reducción de dimensionalidad PCA. Las clases y funciones que implementan estas técnicas se escribieron desde cero en Python y el pseudocódigo de cada uno de ellos se muestra a continuación:

ALGORITMO 1: K-MEANS

```
K = (k1, k2, …, kn) num clusters
C = (c1, c2, …, ck) centroides
D = (d1, d2, …, dn) dataframe
i = num iteraciones
t = umbral

c1, c2, …, ck ∈ D

for i=1 hasta i=I:
  foreach c ∈ C
    foreach d ∈ D
      argminDistance(d, c)
      d →k
    end
  end
  foreach k ∈ K
    c = promedio(d ∈ D)
  end
  if c – c anteriores < t
```

```
    break
  end
end

return K, C
```

ALGORITMO 2: AGGLOMERATIVE

```
D = (d1, d2, ..., dn) dataframe
C = cantidad de clusters

# Inicializar cada instancia como un cluster individual en un
diccionario
K = {i: [di] ∀ di ∈ D}

while cantidad(K) > C:
  (k1, k2) = argminDistancia(K)

  K' = union(K[k1], K[k2])

  remove(K[k1], K[k2] de K)

  K = K ∪ K'

return K
```

ALGORITMO 3: PCA

```
D = (d1, d2, ..., dn) dataframe
C = cantidad de componentes

μ = media(D)
D' = D - μ

CovM = covarianza(D')

# Calcular eigenvalores y eigenvectores
λ, V = eigen(CovM)

# Ordenar eigenvectores
idx = argsort(λ)[descendente]
V' = V[idx]

# Seleccionar los mayores
P = V'[:C]

# Transformar
D'' = D'P

return D''
```

VI. RESULTADOS

A. K-means

La primera técnica de clustering que se implementó fue K-means. El cálculo de los clusters se realizó usando los datos originales y también usando los datos transformados. Es importante mencionar también, que para usar K-means el usuario debe establecer una cantidad de clústers concreta en la que K-means dividirá los datos; para saber esto, un método práctico es el método del codo, en el que a través de una gráfica se puede apreciar cuál sería el mejor número de grupos, el resultado se aprecia a continuación:

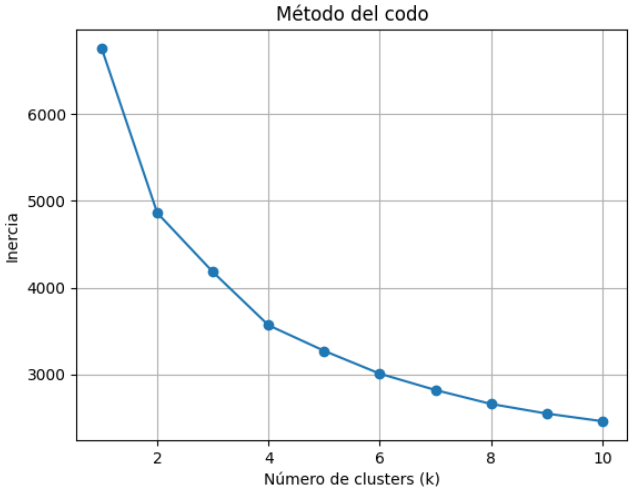

FIGURA 5. Método del codo para estimar la mejor cantidad de clusters.

Para elegir el número de grupos el usuario debe observar en qué parte de la curva se encuentra el mayor punto de inflexión (el codo), en donde el descenso del valor de inercia es más drástico. En este caso es en el valor 4, por lo que este será el número de clusters que se establecerán.

Hecho lo anterior, se procedió a usar K-means en el dataset. Para poder comparar la diferencia de los resultados de manera más directa entre usar PCA y no usarlo antes de aplicar K-means, lo que se hizo fue aplicar estos dos algoritmos de manera normal, pero los resultados se graficaron proyectándolos en los datos transformados por PCA, esto debido a que el orden de las instancias sigue siendo la misma sólo que con sus valores transformados. En resumen, a pesar de que se hubiera calculado K-means sin PCA, los resultados de dicha operación se proyectaron en los resultados de los datos transformados de PCA de tal forma que se pudiera comparar de manera más visual si había algún resultado muy distinto que si se calculaba reduciendo las dimensiones.

En los resultados, los cuáles se pueden observar en las figuras 6 y 7, se puede observar que los clústeres resultantes fueron prácticamente iguales, sin ninguna diferencia (al menos visual) significativa. De esta manera se puede decir, que los resultados que se obtienen al aplicar PCA y al no aplicarlo (pero usando los mismos hiperparámetros), son prácticamente iguales, al menos para este caso, por lo que aplicar técnicas de reducción de dimensionalidad es una buena estrategia, pues permite obtener resultados similares, pero optimizando el tiempo de ejecución y reduciendo la dimensionalidad, de tal forma que la interpretabilidad de los resultados sea más clara.

Como punto negativo de aplicar PCA transformando los datos (pues también podrían seleccionarse las características más relevantes sin hacer ninguna transformación), es que resulta más complicado interpretar los resultados, por lo que se debe analizar de manera más detallada las instancias que pertenecen a cada clúster.

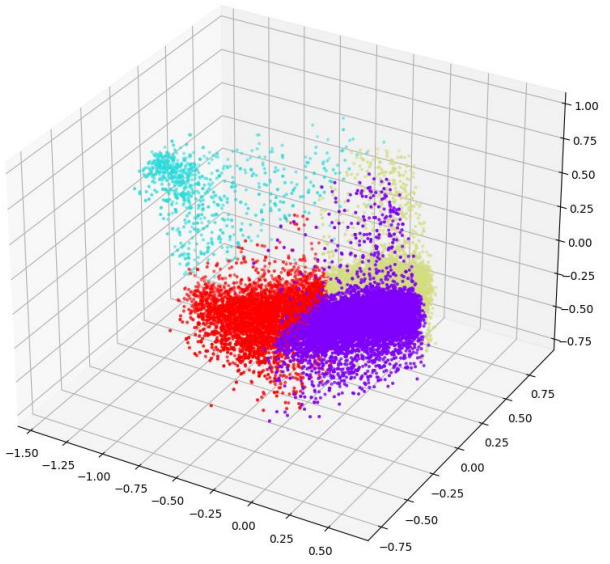

FIGURA 6. **Resultado de K-means sin aplicar PCA.**

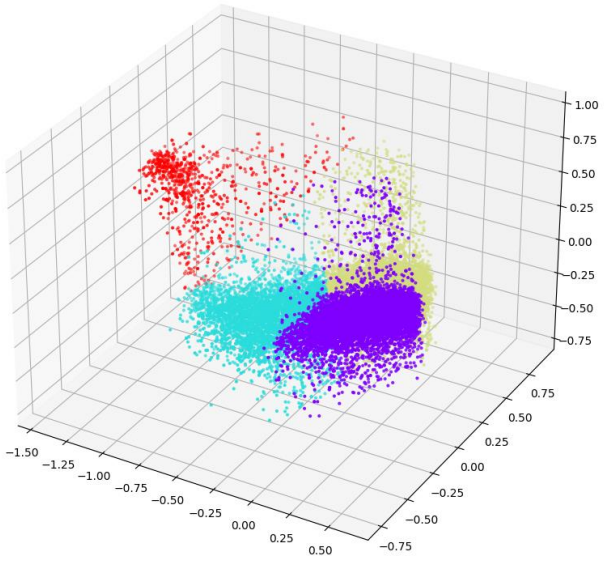

FIGURA 7. **Resultado de K-means después de aplicar PCA.**

B. Agglomerative

Por último, se utilizó el algoritmo Agglomerative para igualmente agrupar los datos en 4 clústers, tal como indicaba el método del codo. Sin embargo, el algoritmo que se construyó, no fue muy eficiente con respecto a la velocidad de cómputo, prueba de esto es que para sólo 500 instancias, el algoritmo tardaba dos horas en converger, por esto se tuvo que trabajar sólo con una muestra de los 20716 datos, esta muestra fue de 500.

Los resultados de Agglomerative con y sin PCA se muestran a continuación:

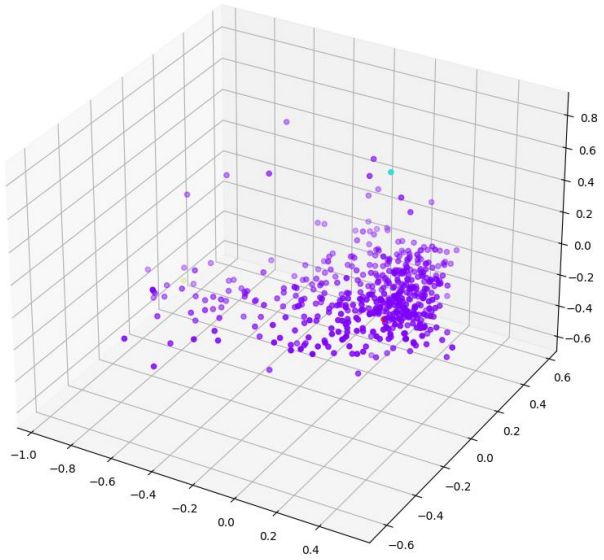

FIGURA 8. **Resultado del algoritmo Agglomerative en la muestra sin aplicar PCA.**

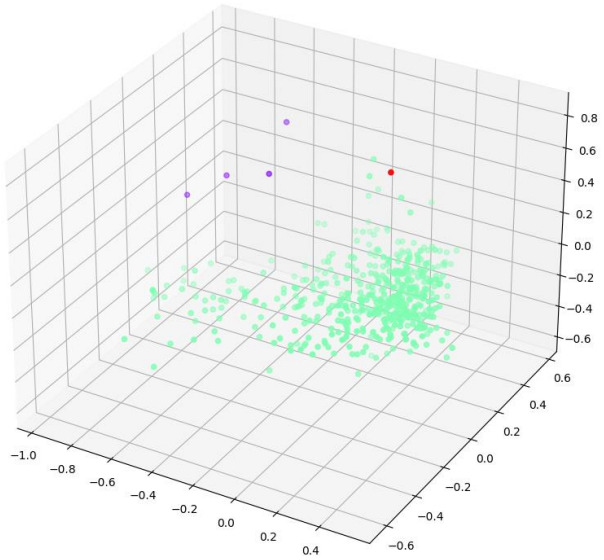

FIGURA 9. **Resultado del algoritmo Agglomerative en la muestra al haber aplicado PCA.**

En estos resultados se pueden apreciar dos diferencias principales (visualmente), la primera es que en ambos hay un clúster predominante que abarca prácticamente todos los datos, y la segunda es que si bien la cantidad de clústers es la misma (en el segundo resultado se encuentra oculto uno de ellos), la cantidad de datos en cada uno es distinta.

Por último, para poder comparar de forma más directa, se quiso aplicar K-means en la misma muestra, de tal forma que se pudieran observar las diferencias entre ambos resultados.

En las figuras siguientes se aprecia que la diferencia entre cada método es notable, mientras que en K-means los clústers están ligeramente mejor delimitados, en Agglomerative las diferencias pueden tornarse confusas. Esto último pudiera deberse a que la muestra es muy pequeña a comparación de la cantidad total de instancias, pudiendo

inclinarse mucho hacia algún grupo en específico debido a la manera de trabajar de Agglomerative.

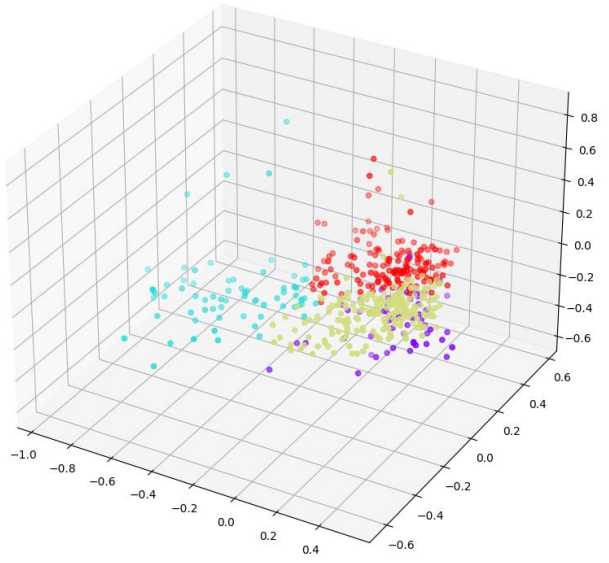

FIGURA 10. Resultado del algoritmo K-means en la muestra sin aplicar PCA.

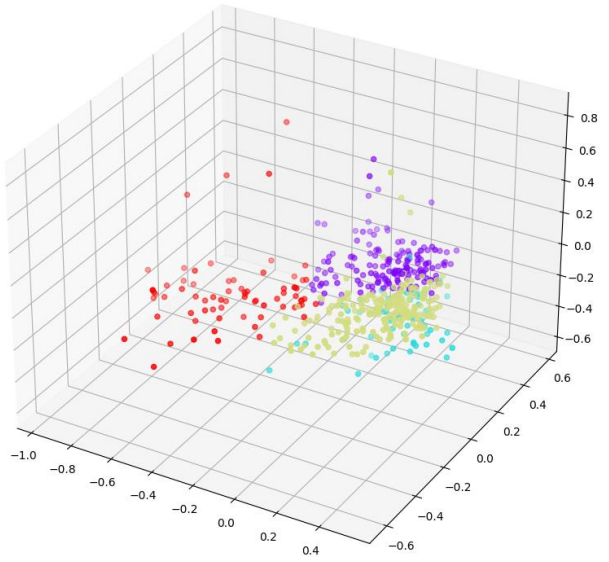

FIGURA 11. Resultado del algoritmo K-means en la muestra al haber aplicado PCA.

VII. CONCLUSIONES

En el presente reporte se ha mostrado la aplicación de funciones de los algoritmos PCA, Agglomerative y K-means, que fueron creadas usando exclusivamente bibliotecas básicas. Los resultados con respecto a PCA y K-means fueron los esperados, pues los clusters resultaban bien delimitados y gracias a PCA se pudieron visualizar de mejor manera los grupos creados. Con respecto al algoritmo Agglomerative, si bien el método lograba converger, este tomaba mucho tiempo computacional haciéndolo factible sólo para analizar pequeñas muestras; además los resultados eran poco interpretables para el usuario.

Dados estos puntos, se puede afirmar que aplicar técnicas de reducción de dimensionalidad puede ser de mucha ayuda para reducir gasto computacional y para poder interpretar mejor los resultados, además de que no parecen afectar demasiado a los algoritmos de clustering. Por su parte usar clustering jerárquico (Agglomerative) parece ser mejor idea para bases de datos que no sean muy extensas, aunque muy probablemente haya mucho margen de mejora en la escritura del código que se creó.

BIBLIOGRAFÍA

[1] M. A. Aceves Fernandez, Inteligencia Artificial para Programadores con Prisa. Independently Published, 2021.

[2] F. Berzal, Redes Neuronales & Deep Learning, vol. 1, 2019.