CS2601
INTELIGENCIA ARTIFICIAL

Proyecto 3: Clustering de información genética en tejidos humanos

---

Mauricio Pinto
Francesco Uccelli
Juan Manuel Navarro

20 de Junio del 2021

1 Introducción

En este proyecto se utilizarán diferentes métodos de clustering para encontrar tejidos en una base de datos que contiene información genérica humana. Dado que la dimensionalidad de los datos es demasiado alta para trabajarla con los métodos directamente, se propone un método de reducción de dimensionalidad mediante análisis de componentes principales. Para el clustering se usarán los métodos de Gaussian Mixture Model, K-means, DBSCAN y Agglomerative Hierarchical Clustering. Finalmente se realizaran experimentos y se compararán los resultados.

2 Análisis

2.1 Datos

El dataset consiste en observaciones de medición de presencia de genes en tejidos humanos. En particular esta data representa expresiones de RNA 8 tejidos distintos, cada uno con múltiples individuos. Esta data es altamente dimensional lo cual dificulta mucho el análisis de su información bruta. Por esto se utiliza el método de ánalisis de componentes principales que se expone a continuación.

2.2 Componentes Principales

El método de reducción de dimensionalidad por análisis de componentes principales consiste en representar la información que brindan las variables en altas dimensiones en dimensiones bajas asumiendo una pérdida de información lo más baja posible. A continuación mostraremos como definimos este método. Para el algoritmo usamos PCA del módulo *sklearn.descomposition*.

A continuación podemos ver el gráfico que nos muestra con que cantidad de componentes se puede explicar que porcentaje de la varianza de los datos:

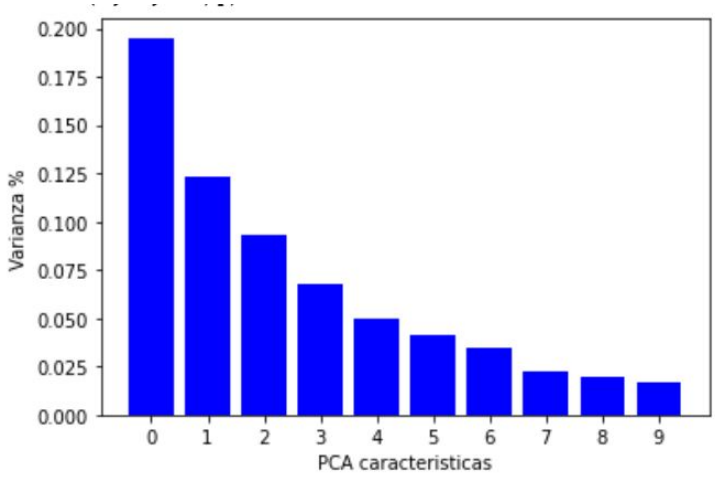


Figure 1: Features del PCA vs Varianza

Como podemos ver en la imagen, es suficiente un con 3 componentes (desde 0 en el gráfico) para representar un poco más de 50% de la varianza en los datos. Esto quiere decir que con esta cantidad de dimensiones deberíamos poder obtener resultados de clustering decentes. Además si mostramos la información que contienen los componentes 0 y 1 podemos ver claramente clusters definidos.

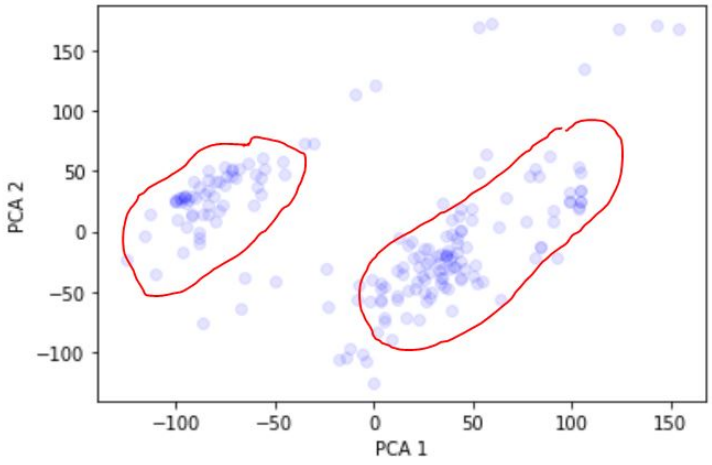


Figure 2: Features 1 y 2

Por otro lado, observemos también con que cantidad de clusters podemos explicar la data en su mayoría. En el siguiente gráfico se muestra la "inercia"

que tiene el significado de cuanto ayuda un nuevo cluster a la distribución de los datos y el número de clusters.

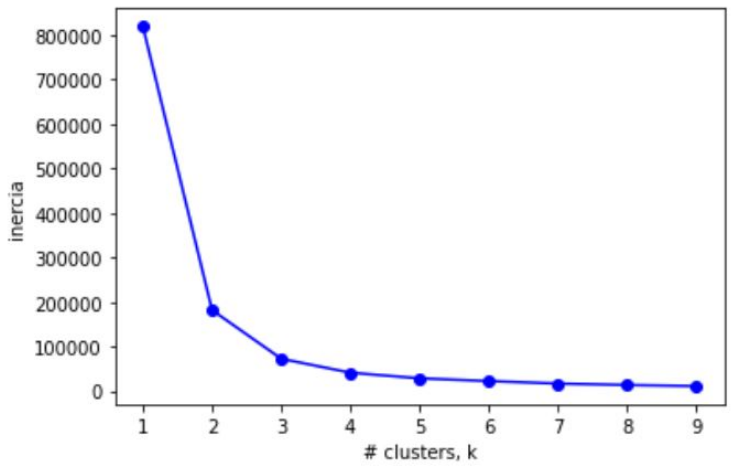


Figure 3: Inercia vs cantidad de clusters

Este gráfico explica que para una cantidad de clusters mayor a 4, aumentar un cluster más va a ayudar muy poco a explicar la distribución de los datos. Es por esto que, entonces, usando 3 dimensiones y 4 clusters se deberían de poder obtener resultados confiables.

3 Algoritmos Utilizados

3.1 K-Means

K-Means es un algoritmo de agrupación en clústeres no supervisado. Se usa cuando tenemos muchos datos sin etiquetar. El objetivo de este algoritmo es encontrar grupos "K" (clusters) entre los datos sin procesar.

El algoritmo funciona de forma iterativa para asignar a cada "punto" (las filas de nuestro conjunto de entrada forman una coordenada) uno de los grupos "K" en función de sus características, los cuales se agrupan segun la similitud de tales caractetisticas, devolviendonos los centroides de cada grupo.

```
def classify (el, centers):
    minn = 1000000
    for key in centers:
        if dist(el, centers[key]) < minn:
            minn = dist(el, centers[key])
            cluster = key
    return cluster
```

```
def kmeans(df,k,cluster_centers):
    classes = {}
    for i in range (k):
        classes[str(i)] = []
    for x in range(len(df[0])):
        element = list ([df[0][x],df[1][x]])
        assigned_cluster =
        classify (element, cluster_centers)
        classes[assigned_cluster].append (element)
    return classes
```

3.2 GMM

Gaussian mixture model intenta encontrar una mezcla de distribuciones de probabilidad gaussianas multidimensionales que modelen mejor cualquier conjunto de datos de entrada. En el caso más simple, los GMM se pueden usar para encontrar clusters de la misma manera que k-means. Pero debido a que GMM contiene un modelo probabilístico bajo el capó, también es posible encontrar asignaciones de conglomerados probabilísticos.

```
def run_gmm (self, iterations, x, aux=None):
    means = self.gmm_generate_means ()
    prior = self.gmm_generate_prior ()
    covariance = self.gmm_generate_covariance (means, x)

    for i in range (iterations):
        gamma = self.gmm_calculate_gamma (prior,
        covariance, x, means)
        means = self.gmm_calculate_means (gamma, x)
        prior = self.gmm_calculate_prior (gamma, x)
        covariance = self.gmm_calculate_covariance (gamma,
        x, means)

    return means
```

3.3 DBSCAN

es un algoritmo básico para el agrupamiento basado en densidad. Puede descubrir grupos de diferentes formas y tamaños a partir de una gran cantidad de datos, que contienen ruido y valores atípicos.

K-means puede agrupar observaciones poco relacionadas. Cada observación eventualmente se convierte en parte de algún grupo, incluso si las observaciones están dispersas en el espacio vectorial. Un ligero cambio en los puntos de datos podría afectar el resultado de la agrupación. El DBSCAN reduce el problema debido a la forma en que se forman los clusters.

Una ventaja del DBSCAN es que no tienes que especificar el numero de clusters a utilizar, solo necesitas la función para calcular la distancia entre valores y definir la distancia que se pueda considerar cercana (minPoints y radius). DBSCAN también produce resultados mas razonables a comparación de K-means.

```
def dbscan(puntos,clusters,radius,tree):
    for i in range (len(puntos)):
        assigned_cluster = -1
        if clusters[i] == -1:
            assigned_cluster = i
        else:
            assigned_cluster = clusters[i]
        ind = tree.query_radius([puntos[i]], r=radius,
        count_only=False, return_distance=False)
        is_cluster = False
        if len(ind[0]) >= 10:
            is_cluster = True
        if is_cluster:
            for m in ind[0]:
                clusters[m] = assigned_cluster
    return clusters
```

3.4 AHC

Agglomerative Heriarchical Clustering trae también ciertas ventajas sobre el resto. Funciona a partir de las diferencias entre los objetos a agrupar. Un tipo de disimilitud se puede adaptar al tema estudiado y a la naturaleza de los datos. Además, uno de los resultados es el dendrograma que muestra la agrupación progresiva de los datos. Entonces es posible tener una idea de un número adecuado de clases en las que se pueden agrupar los datos.

AHC funciona de manera iterativa, empezando por calcular la disimilitud entre N objetos. Luego juntando 2 objetos para ver si es que juntos minimizan un criterio de aglomeración, y finalmente calculando la disimilitud entre la clase y el resto de objetos utilizando el mismo criterio.

4 Experimentación

Para el caso de los gráficos se plantean diferentes cantidades de clusters en el k-means y en el DBSCAN los clusters se generan automáticamente. Por esto, puede resultar no completamente útil comparar medidas de precisión en casos donde la cantidad de clusters difiere. En cambio, planteamos a continuación el gráfico con los valores de las labels reales y comparamos los resultados de cada experimento visualmente0.

Figure 4: Comparación entre diferentes algoritmos de clustering

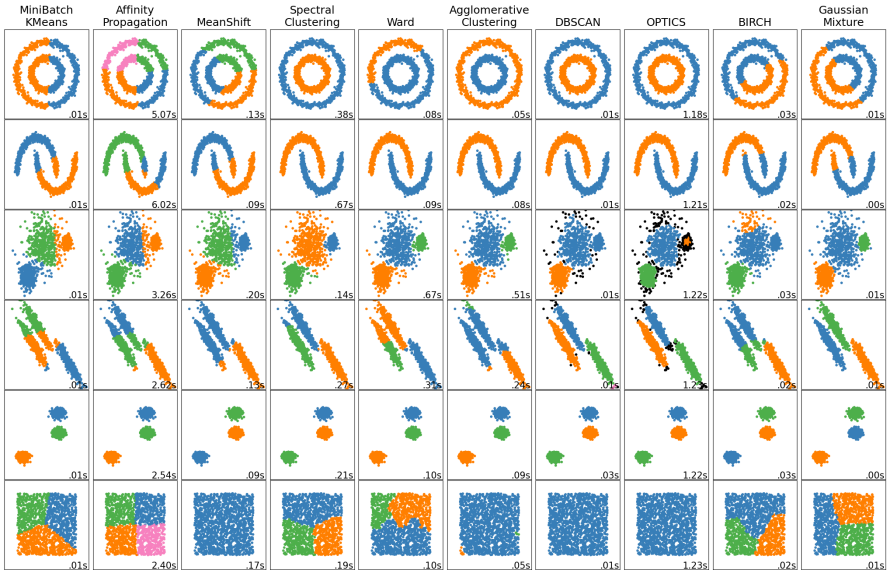
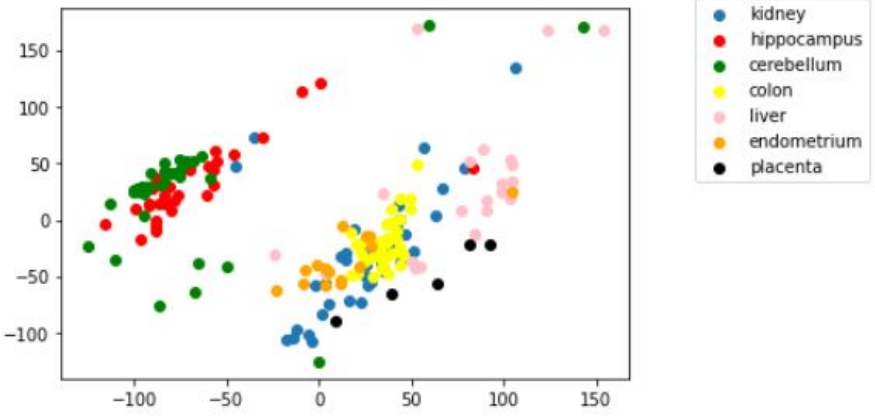


Figure 5: 3 clusters



4.1 K-Means

En este algoritmo variamos el parámetro k y mostramos los resultados

Figure 6: 3 clusters

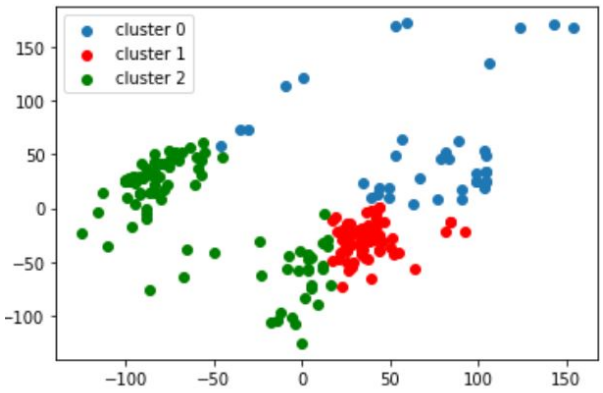


Figure 7: 5 clusters

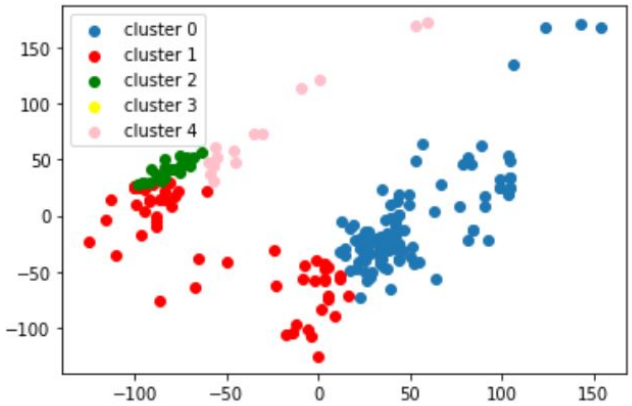
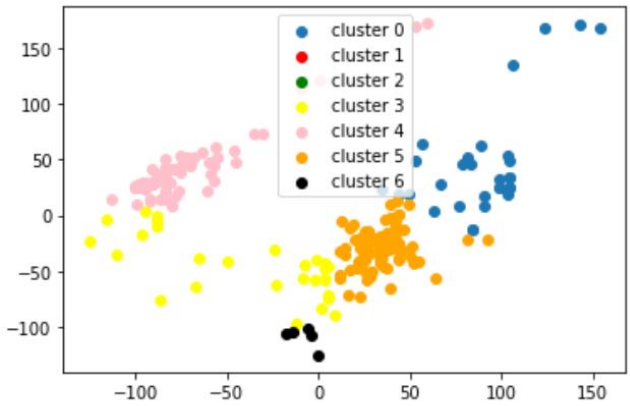


Figure 8: 7 clusters



4.2 DBSCAN

Figure 9: radio 10 minPoints 5

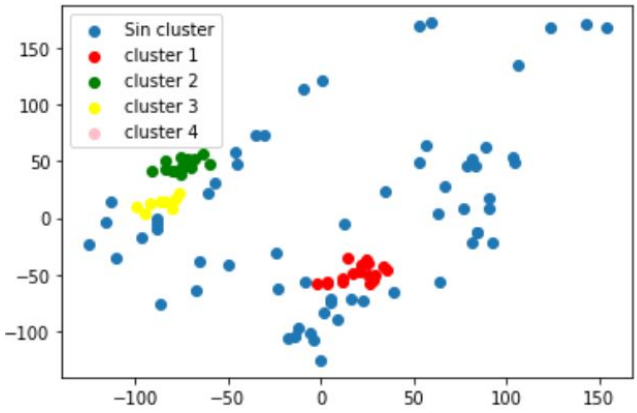


Figure 10: radio 20 minPoints 8

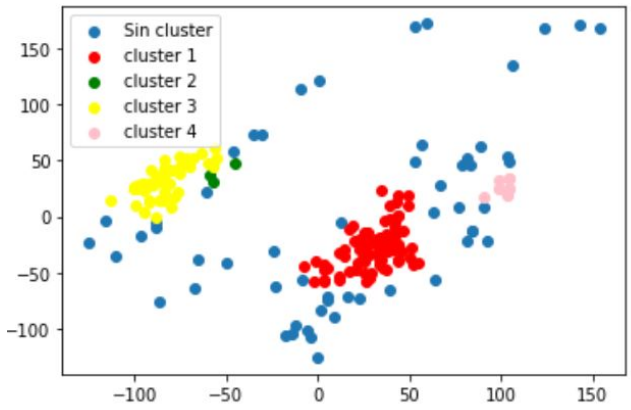
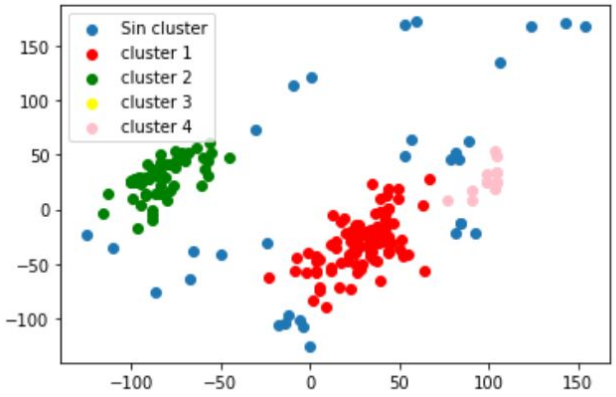


Figure 11: radio 10 minPoints 5



4.3 AHC

Figure 12: 5 clusters

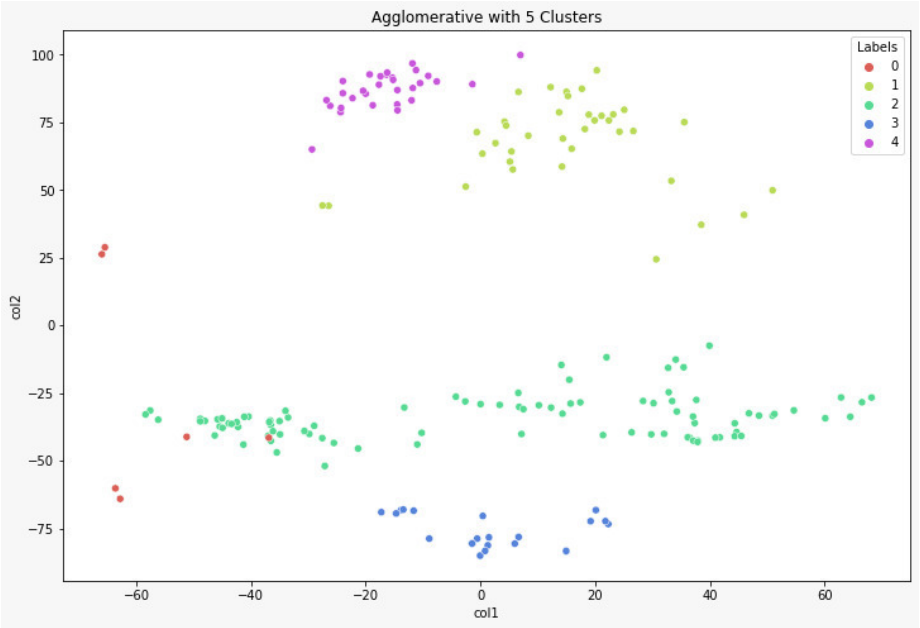


Figure 13: dendograma con average

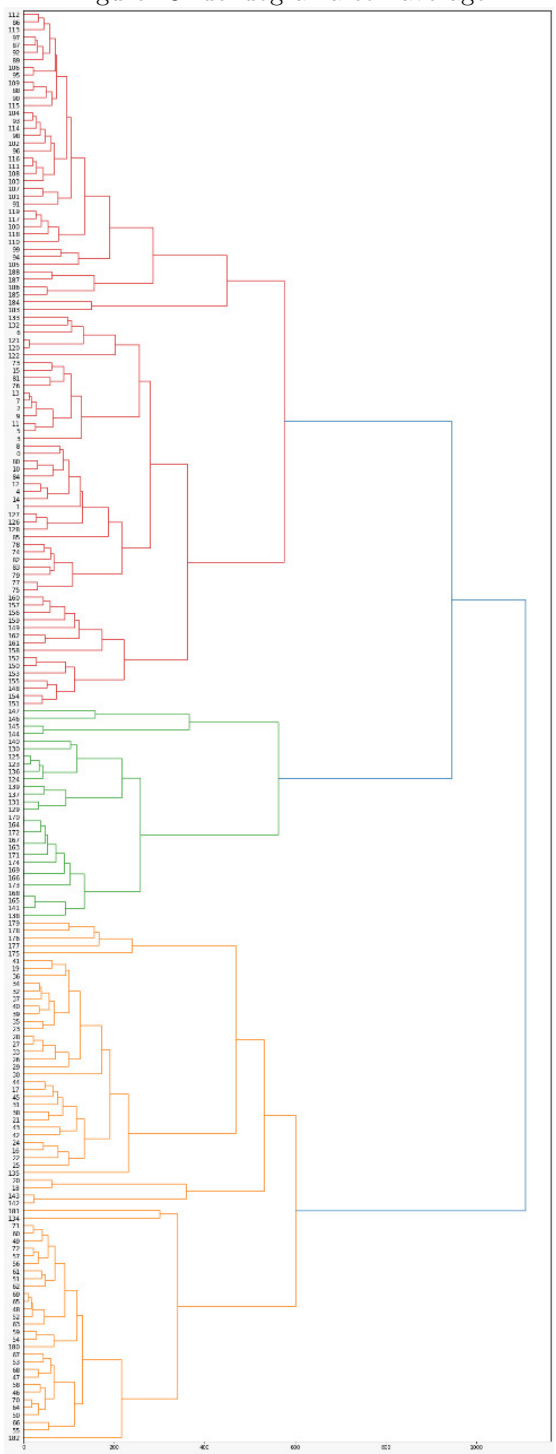
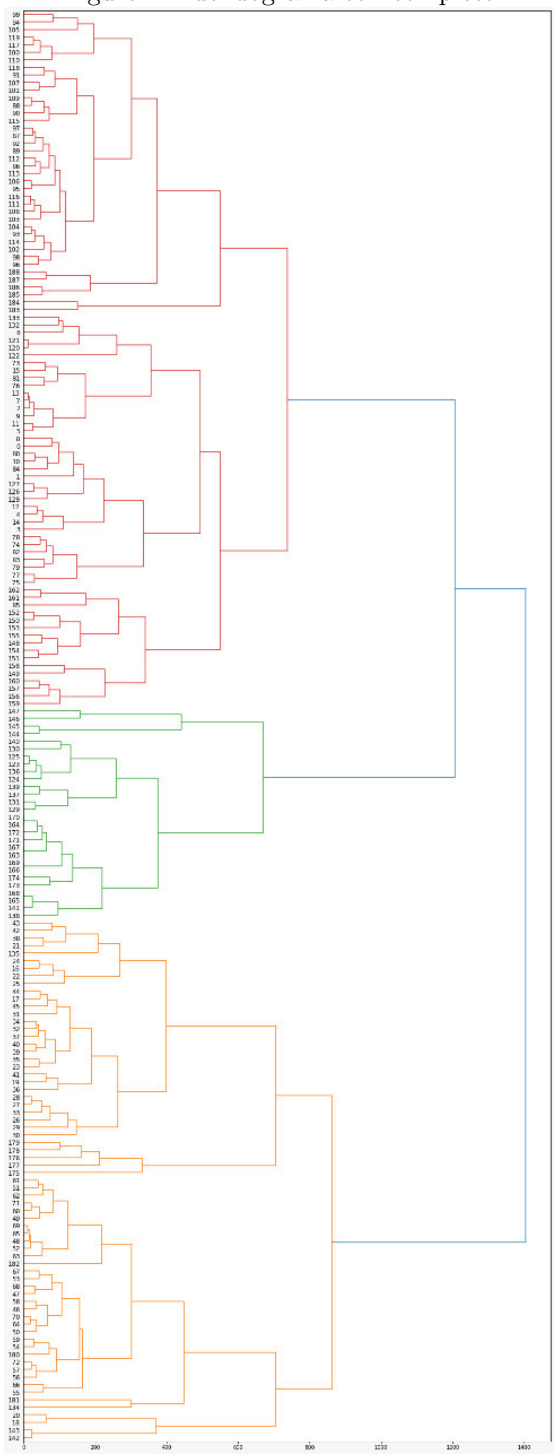


Figure 14: dendograma con complete



4.4 GMM

Los resultados obtenidos con GMM no reflejaban su funcionamiento correcto. El código se encuentra en el repositorio anexado.

5 Conclusiones

En conclusión, podemos afirmar que el rendimiento de cada algoritmo es diferente y puede variar significativamente según la implementación utilizada. Por ejemplo, AHC es excelente cuando no necesariamente tenemos clusters circulares y no sabemos el numero de clusters de antemano. Con este podemos decidir el numero de clusters cortando el dendograma, pero con una complejidad cuadrática en la mayoría de casos.

Por otro lado, el DBSCAN es bueno para agrupar puntos estén muy cerca y expendiendo los clusters en dirección a puntos cercanos, lidiando así con diferentes formas de clusters.

Al final, como mencionamos previamente, el rendimiento de cada uno de estos algoritmos esta ligado fuertemente al dataset y para que se le decida utilizar.

6 Anexos

- Github: https://github.com/mauriciopinto/proyecto3_IA
- Kaggle: https://www.kaggle.com/flrotm/clustering

7 Bibliografía

[1] https://www.cienciadedatos.net/documentos/py19-pca-python.html
[2] https://rstudio-pubs-static.s3.amazonaws.com/238819_bc76e21ff51248f7833cf5c90e68197a.html