**RECONOCIMIENTO DE FORMAS**

**MEMORIA DE CLUSTERING**



**Autores:**

González Rodríguez, Enrique; Matrícula: y160329

Jiménez Martín, Carlos; Matrícula: y160190

**FACULTAD DE INGENIERÍA INFORMÁTICA UNIVERSIDAD POLITÉCNICA DE MADRID**

**CURSO 2019-2020**

**Índice**

[1. Chain-Map 3](#_Toc31286168)

[1.1 Iris Dataset 4](#_Toc31286169)

[1.2 Wine Dataset 4](#_Toc31286170)

[1.3 Cancer Dataset 4](#_Toc31286171)

[1.4 Isolet Dataset 5](#_Toc31286172)

[1.5 MNIST Dataset 5](#_Toc31286173)

[2. Algoritmo K-Medias 5](#_Toc31286174)

[2.1 Iris Dataset 7](#_Toc31286175)

[2.2 Wine Dataset 7](#_Toc31286176)

[2.3 Cáncer Dataset 7](#_Toc31286177)

[2.4 Isolet Dataset 7](#_Toc31286178)

[2.5 MNIST Dataset 8](#_Toc31286179)

[3. Algoritmo secuencial 8](#_Toc31286180)

[3.1 Cancer Dataset 10](#_Toc31286181)

[3.2 Wine Dataset 10](#_Toc31286182)

[3.3 Cancer Dataset 10](#_Toc31286183)

[3.4 Isolet Dataset 10](#_Toc31286184)

[3.5 MNIST Dataset 11](#_Toc31286185)

[4. Compactación 11](#_Toc31286186)

[5. Ensembles 13](#_Toc31286187)

[5.1 Iris Dataset 13](#_Toc31286188)

[5.2 Wine Dataset 14](#_Toc31286189)

[5.3 Cancer Dataset 15](#_Toc31286190)

[6. MeanShift 17](#_Toc31286191)

[6.1 Iris Dataset 17](#_Toc31286192)

[6.2 Wine Dataset 17](#_Toc31286193)

[6.3 Cancer Dataset 17](#_Toc31286194)

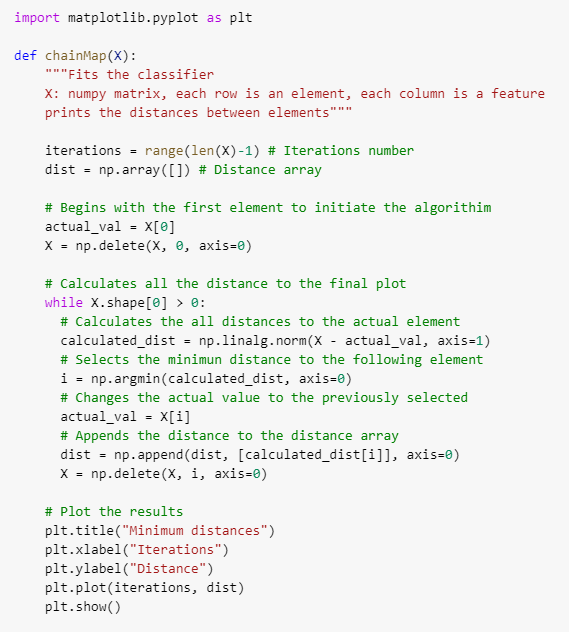
[6.4 Isolet Dataset 18](#_Toc31286195)

[6.5 MNIST Dataset 18](#_Toc31286196)

# Chain-Map

El algoritmo Chain-Map es un algoritmo en el que dado una serie de elementos (X1, X2, X3...) a clasificar, se escoge uno al azar (Xi) y se establece una sucesión Xi(0), Xi(1), Xi(2)... donde Xi(1) es el elemento más próximo a Xi(0), Xi(2) el más próximo a Xi(1), y así sucesivamente.

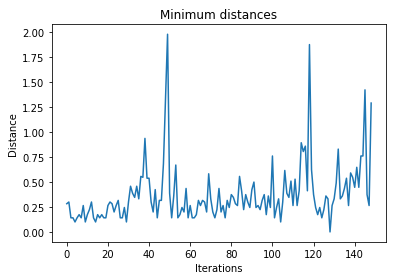
La implementación en Python es la siguiente:



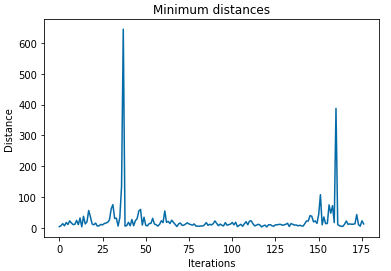
A partir de la representación gráfica de las distancias entre los elementos de la sucesión, se pueden detectar una aproximación del número de clusters o clases, dividiéndose estas a través de los saltos grandes entre elementos, como puede verse en las gráficas mostradas a continuación sobre los distintos datasets. Además, se puede deducir también un umbral “máximo” a partir del cual los elementos dejan de formar parte de una clase.

Por ejemplo, en el caso de Iris, se puede deducir que hay un total de 3 clases al haber principalmente dos saltos extensos entre elementos y que el umbral máximo en el que un dato deja de formar parte de un cluster es alrededor de 1.60.

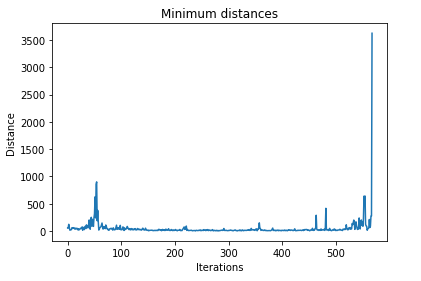
## Iris Dataset



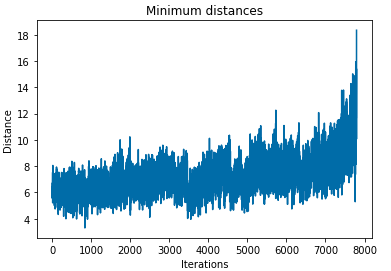
## Wine Dataset



## Cancer Dataset

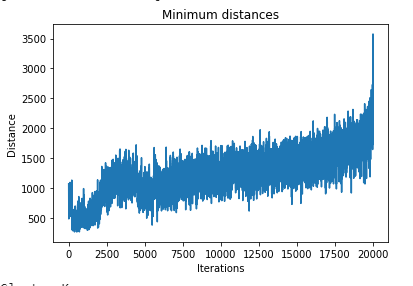


## Isolet Dataset



## MNIST Dataset

Para este caso, al ser un dataset tan grande, hemos seleccionado únicamente 20000 datos para probar nuestro algoritmo.

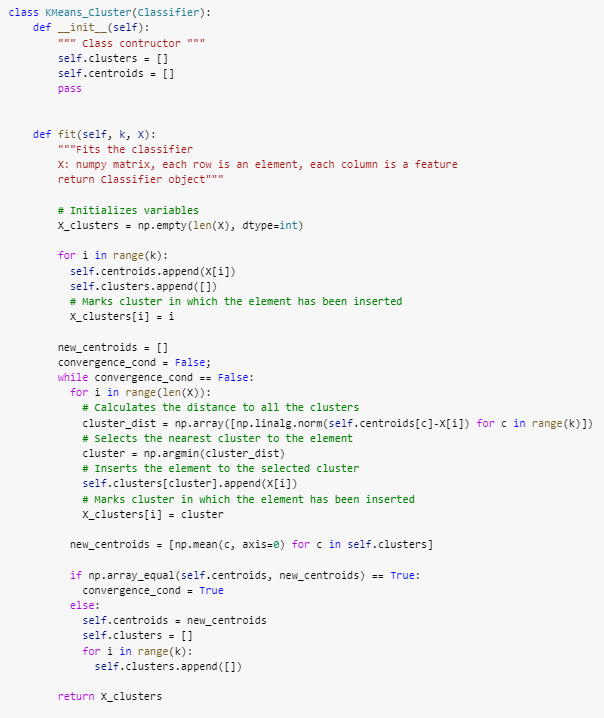


# Algoritmo K-Medias

El algoritmo K-Medias es un algoritmo de clasificación no supervisada, en el que se conoce previamente el número de clases exactas. Consta de los siguientes pasos:

1. Inicialmente se calculan los centroides de las clases.
2. Se distribuyen los elementos a clasificar entre las clases según la distancia mínima entre el elemento y los centroides de las clases.
3. Se actualizan los centroides.
4. Si se alcanza una distribución estable se termina, en caso contrario se vuelve al paso número 2.

La implementación en Python es la siguiente:



Primeramente, hemos aplicado el algoritmo de Chain-Map, para deducir el parámetro “K” de entrada del algoritmo K-means para cada dataset, al ser casos distintos entre sí. Se ha seguido el criterio explicado en el apartado 1 para la suposición del número de clusters.

Después, aplicamos el algoritmo K-means con el parámetro “K” elegido, obteniendo los siguientes resultados para cada dataset:

## Iris Dataset

k = 3

**Cluster 0:** Label: 0, Compactation: 100.00%

**Cluster 1:** Label: 1, Compactation: 77.42%

**Cluster 2:** Label: 2, Compactation: 94.74%

## Wine Dataset

k = 3

**Cluster 0:** Label: 1, Compactation: 72.46%

**Cluster 1:** Label: 0, Compactation: 97.87%

**Cluster 2:** Label: 2, Compactation: 46.77%

## Cáncer Dataset

k = 2

**Cluster 0:** Label: 1, Compactation: 81.28%

**Cluster 1:** Label: 0, Compactation: 99.24%

## Isolet Dataset

k = 26

**Cluster 0:** Label: 4, Compactation: 40.06%

**Cluster 1:** Label: 18, Compactation: 33.75%

**Cluster 2:** Label: 16, Compactation: 64.29%

**Cluster 3:** Label: 0, Compactation: 90.28%

**Cluster 4:** Label: 12, Compactation: 60.35%

**Cluster 5:** Label: 16, Compactation: 93.33%

**Cluster 6:** Label: 11, Compactation: 61.74%

**Cluster 7:** Label: 21, Compactation: 38.01%

**Cluster 8:** Label: 13, Compactation: 61.56%

**Cluster 9:** Label: 19, Compactation: 40.47%

**Cluster 10:** Label: 2, Compactation: 51.33%

**Cluster 11:** Label: 24, Compactation: 99.66%

**Cluster 12:** Label: 8, Compactation: 90.13%

**Cluster 13:** Label: 4, Compactation: 26.23%

**Cluster 14:** Label: 10, Compactation: 35.74%

**Cluster 15:** Label: 17, Compactation: 94.22%

**Cluster 16:** Label: 16, Compactation: 85.37%

**Cluster 17:** Label: 21, Compactation: 19.03%

**Cluster 18:** Label: 14, Compactation: 97.86%

**Cluster 19:** Label: 22, Compactation: 100.00%

**Cluster 20:** Label: 11, Compactation: 66.81%

**Cluster 21:** Label: 22, Compactation: 82.54%

**Cluster 22:** Label: 7, Compactation: 99.30%

**Cluster 23:** Label: 10, Compactation: 46.51%

**Cluster 24:** Label: 6, Compactation: 37.90%

**Cluster 25:** Label: 20, Compactation: 96.75%

## MNIST Dataset

Para este caso, al ser un dataset tan grande, hemos seleccionado únicamente 20000 datos para probar nuestro algoritmo.

k = 10

**Cluster 0:** Label: 8, Compactation: 91.93%

**Cluster 1:** Label: 6, Compactation: 50.50%

**Cluster 2:** Label: 3, Compactation: 61.61%

**Cluster 3:** Label: 7, Compactation: 89.26%

**Cluster 4:** Label: 9, Compactation: 72.26%

**Cluster 5:** Label: 1, Compactation: 94.15%

**Cluster 6:** Label: 4, Compactation: 43.46%

**Cluster 7:** Label: 1, Compactation: 31.48%

**Cluster 8:** Label: 2, Compactation: 36.91%

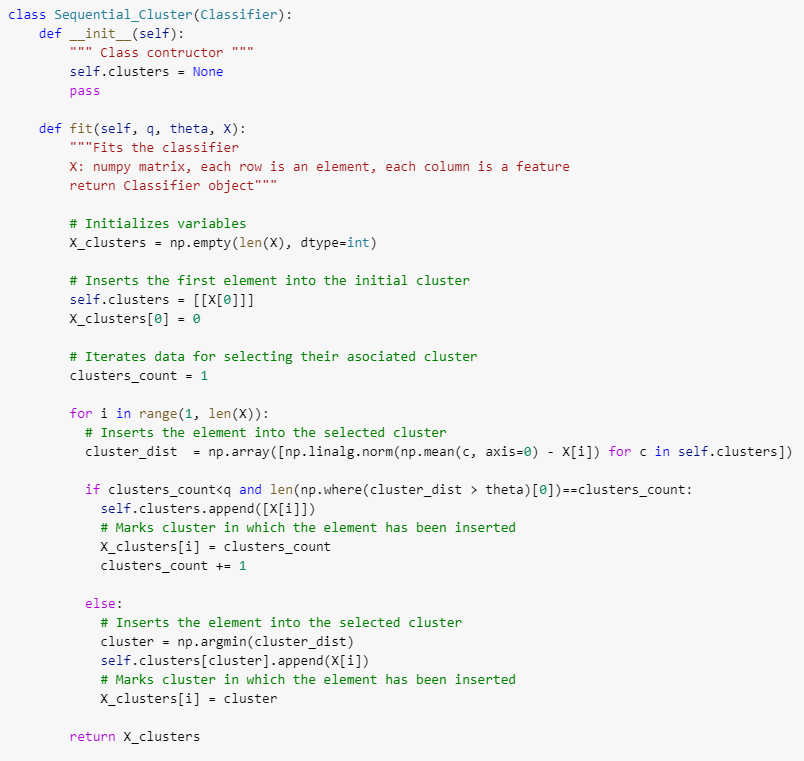
**Cluster 9:** Label: 5, Compactation: 90.80%

# Algoritmo secuencial

El algoritmo secuencial de clustering es un algoritmo de clasificación no supervisada. Tiene como parámetro un umbral de distancia, y el número máximo de cluster que puede haber (según el criterio de la persona que ejecute el código a partir del Chain-Map). Consta de los siguientes pasos:

1. Inserta el primer elemento del dataset en el primer cluster.
2. Con los siguientes elementos del dataset busca el cluster cuya distancia al centroide sea mínima. En el caso de que dicha distancia no supere el umbral dado como parámetro, se incluirá el elemento en el cluster. Por otro lado, si la distancia mínima supera el umbral de distancia, significa que se supera el umbral para todos los clusters, por lo que se crea un cluster nuevo para el elemento. En el caso de que se haya alcanzado el límite de clusters y la distancia mínima supere el umbral, el elemento se introduce en el cluster con la distancia mínima.

La implementación en Python es la siguiente:



Primeramente, al igual que en el caso de K-Means, se aplica el algoritmo de Chain-Map para deducir los parámetros de entrada del algoritmo secuencial. La diferencia con el algoritmo anterior es que, en vez de dar un número fijo de clusters a crear (como era el parámetro “K”), se usa para dar un número máximo de clusters (parámetros “Q”), aunque no obliga al algoritmo a crear todos. Además, se le pasa también el parámetro “θ”, que sirve a modo de umbral máximo de distancia a la hora de crear los clusters. Se ha seguido el criterio explicado en el apartado 1 para la suposición del número de clusters.

Después, aplicamos el algoritmo secuencial con los parámetros “Q” y “θ” elegido, obteniendo los siguientes resultados para cada dataset:

## Cancer Dataset

Q = 3

θ = 1.5

**Cluster 0:** Label: 2, Compactation: 90.70%

**Cluster 1:** Label: 0, Compactation: 100.00%

**Cluster 2:** Label: 1, Compactation: 80.70%

## Wine Dataset

Q = 3

θ = 100

**Cluster 0:** Label: 2, Compactation: 47.62%

**Cluster 1:** Label: 1, Compactation: 73.53%

**Cluster 2:** Label: 0, Compactation: 97.87%

## Cancer Dataset

Q = 2

θ = 800

**Cluster 0:** Label: 0, Compactation: 99.26%

**Cluster 1:** Label: 1, Compactation: 82.03%

## Isolet Dataset

Q = 26

θ = 7.75

**Cluster 0:** Label: 4, Compactation: 22.37%

**Cluster 1:** Label: 18, Compactation: 33.45%

**Cluster 2:** Label: 16, Compactation: 84.00%

**Cluster 3:** Label: 12, Compactation: 49.68%

**Cluster 4:** Label: 13, Compactation: 46.59%

**Cluster 5:** Label: 20, Compactation: 72.22%

**Cluster 6:** Label: 11, Compactation: 69.41%

**Cluster 7:** Label: 4, Compactation: 33.23%

**Cluster 8:** Label: 13, Compactation: 59.18%

**Cluster 9:** Label: 19, Compactation: 33.33%

**Cluster 10:** Label: 2, Compactation: 50.62%

**Cluster 11:** Label: 24, Compactation: 99.29%

**Cluster 12:** Label: 8, Compactation: 82.51%

**Cluster 13:** Label: 6, Compactation: 21.99%

**Cluster 14:** Label: 7, Compactation: 50.99%

**Cluster 15:** Label: 17, Compactation: 81.62%

**Cluster 16:** Label: 16, Compactation: 87.55%

**Cluster 17:** Label: 21, Compactation: 42.67%

**Cluster 18:** Label: 14, Compactation: 98.55%

**Cluster 19:** Label: 22, Compactation: 100.00%

**Cluster 20:** Label: 11, Compactation: 48.67%

**Cluster 21:** Label: 19, Compactation: 43.19%

**Cluster 22:** Label: 7, Compactation: 100.00%

**Cluster 23:** Label: 9, Compactation: 34.53%

**Cluster 24:** Label: 10, Compactation: 39.79%

**Cluster 25:** Label: 20, Compactation: 100.00%

## MNIST Dataset

Para este caso, al ser un dataset tan grande, hemos seleccionado únicamente 20000 datos para probar nuestro algoritmo.

Q = 10

θ = 1000

**Cluster 0:** Label: 8, Compactation: 53.95%

**Cluster 1:** Label: 6, Compactation: 43.05%

**Cluster 2:** Label: 3, Compactation: 33.92%

**Cluster 3:** Label: 2, Compactation: 42.99%

**Cluster 4:** Label: 3, Compactation: 59.10%

**Cluster 5:** Label: 1, Compactation: 92.17%

**Cluster 6:** Label: 4, Compactation: 34.31%

**Cluster 7:** Label: 9, Compactation: 50.39%

**Cluster 8:** Label: 8, Compactation: 59.80%

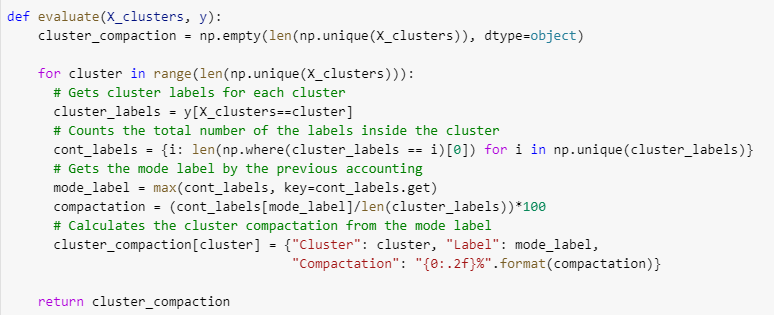
**Cluster 9:** Label: 5, Compactation: 51.02%

# Compactación

Con el fin de evaluar el rendimiento de los algoritmos de clustering propuestos, se ha utilizado el método de compactación, el cual se basa en calcular la moda de un cluster de datos, y el número de datos del cluster cuya etiqueta coincide con la moda.

Para medir el grado de calidad de compactación de un algoritmo de clustering, se divide el número de datos que están etiquetados igual que la moda del propio cluster, respecto al total de datos que lo constituyen.

Implementación en Python:



Realizando varias pruebas sobre los distintos datasets, hemos podido observar como la compactación mejoraba cuando el número de clusters elegidos (parámetros “K” y “Q”) se acercaba al número real de clusters del dataset, aunque no en todos los casos con la misma magnitud de mejora.

También ha mejorado cambiando el umbral máximo en el algoritmo secuencial, pudiendo observarse que, cuanto más lo optimizásemos más mejoraba el porcentaje de compactación.

Para ello, nos fijamos en las gráficas impresas por el algoritmo Chain-Map, observando principalmente los grandes saltos del histograma, de manera que elijamos un umbral que no sea lo suficientemente amplio como para evitar la creación de las clases en los casos donde no hay mucha distancia entre una clase y otra, ya que como se puede observar en la gráficas de los cinco datasets, cuantos más elementos haya, mayor es la diferencia de distancias entre clases, siendo más fácil elegir este valor en los tres primero casos al ser la gráfica mucho más comprensible a simple vista.

# Ensembles

Se ha implementado el método de Ensembles para mejorar el rendimiento del conjunto de clasificadores, buscando alcanzar una mejora del 5% respecto al inicial.

Después de implementar el método y comprobar los resultados, se ha observado que no se han conseguido las mejoras que se querían obtener, debido a razones que desconocemos ya que se ha comprobado el mismo método con el algoritmo del Perceptrón y con éste funciona correctamente alcanzando la mejora esperada. También se ha tenido que establecer un límite de generación de clasificadores (que en este caso ha sido seis), ya que se generaban clasificadores idénticos por lo que se provocaba un bucle infinito, siendo necesario cortar dicho bucle al no alcanzarse la convergencia buscada.

Por otro lado, con los dataset Isolet y MNIST no se pudo aplicar este método, ya que al ejecutar el algoritmo nos devolvía un error por tiempo de ejecución debido a la gran cantidad de datos y características que contienen dichos datasets.

A continuación, se muestran las gráficas resultado:

## 5.1 Iris Dataset

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Iris algoritmo K-Means**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Iris algoritmo K-Means Secuencial**

## 5.2 Wine Dataset

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Wine algoritmo K-Means**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

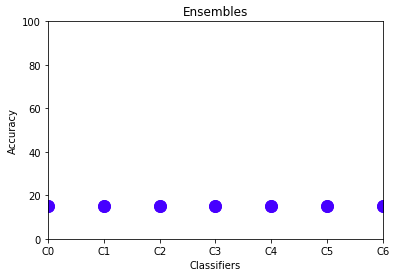
**Wine algoritmo K-Means Secuencial**

## 5.3 Cancer Dataset

Imagen que contiene captura de pantalla

Descripción generada automáticamente

**Cancer algoritmo K-Means**

****

**Cancer algoritmo K-Means Secuencial**

# MeanShift

MeanShift realiza el clustering mediante el uso de un núcleo plano. Esta técnica tiene como objetivo descubrir “gotas” en una densidad uniforme de muestras.

Es un algoritmo basado en centroides, que funcionan actualizando candidatos para los mismos, siendo los puntos medios de una región determinada o “ventana”. Luego, estos candidatos se filtran en una etapa de procesamiento posterior para eliminar duplicados y formar el conjunto final de centroides.

Tiene dos parámetros importantes que se deben tener en cuenta.

* El primero es el “bandwidth”, el cual define el radio de las regiones o ventanas, mencionadas previamente, que una observación utiliza para determinar la dirección de desplazamiento. Puede configurarse manualmente, pero de forma predeterminada estima un ancho razonable de forma automática, aunque conlleva un significativo costo computacional.
* El segundo es el “cluster\_all”. El cual sirve para determinar que ocurre con los datos que se quedan “huérfanos” respecto a los núcleos de las distintas ventanas. Si tras todas las observaciones quedan datos fuera de un cluster, de forma predeterminada se asignan a los clusters más cercanos, pero en caso de que no se busque este comportamiento, se pude asignar a *False* este parámetro, asignándose entonces estos datos a la etiqueta de “-1”.

A continuación, se muestran los resultados de aplicar este algoritmo a las bases de datos que hemos estado utilizando durante el resto de la práctica:

## 6.1 Iris Dataset

**Cluster 0:** Label: 1, Compactation: 50.00%

**Cluster 1:** Label: 0, Compactation: 100.00%

## 6.2 Wine Dataset

**Cluster 0:** Label: 1, Compactation: 38.51%

**Cluster 1:** Label: 1, Compactation: 100.00%

## 6.3 Cancer Dataset

**Cluster 0:** Label: 1, Compactation: 68.88%

**Cluster 1:** Label: 0, Compactation: 100.00%

**Cluster 2:** Label: 0, Compactation: 100.00%

**Cluster 3:** Label: 0, Compactation: 100.00%

**Cluster 4:** Label: 0, Compactation: 100.00%

**Cluster 5:** Label: 0, Compactation: 100.00%

**Cluster 6:** Label: 0, Compactation: 100.00%

**Cluster 7:** Label: 0, Compactation: 100.00%

**Cluster 8:** Label: 0, Compactation: 100.00%

**Cluster 9:** Label: 0, Compactation: 100.00%

**Cluster 10:** Label: 0, Compactation: 100.00%

**Cluster 11:** Label: 1, Compactation: 85.71%

**Cluster 12:** Label: 1, Compactation: 100.00%

**Cluster 13:** Label: 1, Compactation: 100.00%

## 6.4 Isolet Dataset

**Cluster 0:** Label: 0, Compactation: 3.88%

**Cluster 1:** Label: 22, Compactation: 80.00%

**Cluster 2:** Label: 24, Compactation: 71.43%

**Cluster 3:** Label: 15, Compactation: 66.67%

**Cluster 4:** Label: 22, Compactation: 100.00%

**Cluster 5:** Label: 22, Compactation: 100.00%

**Cluster 6:** Label: 5, Compactation: 50.00%

**Cluster 7:** Label: 5, Compactation: 100.00%

**Cluster 8:** Label: 5, Compactation: 100.00%

**Cluster 9:** Label: 22, Compactation: 100.00%

**Cluster 10:** Label: 22, Compactation: 31.58%

**Cluster 11:** Label: 22, Compactation: 93.33%

## 6.5 MNIST Dataset

Debido a la enorme cantidad de datos de esta base de datos, no es posible aplicar este algoritmo en MNIST, dando un error de procesamiento por exceder el máximo tiempo de ejecución permitido.