Instituto Politécnico Nacional

Escuela Superior de Cómputo

Profesor:

Alumnos: García Floriano Andrés

Hernández Jiménez Erick Yael

Patiño Flores Samuel

Robert Garayzar Arturo

5BV1

Ejercicio 2: K-Means 2

Índice

1. Introducción

La práctica presentada en este documento se centra en la implementación y análisis del algoritmo K-Means, una técnica de agrupamiento ampliamente utilizada en el aprendizaje automático y la minería de datos. A través de este ejercicio, se busca no solo explorar el funcionamiento del código que aplica K-Means a imágenes utilizando datasets de Scikit-Learn, sino también profundizar en la optimización del proceso y la evaluación de su desempeño.

El proceso inicia con la carga y preprocesamiento de la imagen, donde los valores de los píxeles se normalizan para mejorar la precisión del algoritmo. Luego, se selecciona una muestra aleatoria de píxeles, lo que permite optimizar el rendimiento evitando el procesamiento completo de la imagen. Una vez entrenado el modelo, cada píxel es etiquetado y la imagen se reconstruye, lo que posibilita realizar una comparación visual con una paleta de colores aleatoria.

A lo largo del documento se analizan también diferentes métodos de inicialización del algoritmo, como K-Means con inicialización aleatoria y K-Means++. Esto permite estudiar su impacto en la estabilidad y calidad de los clústeres formados, destacando la relevancia de una adecuada selección de parámetros y técnicas para mejorar el rendimiento. Se presentan resultados visuales que ilustran la efectividad de los métodos empleados, ofreciendo una comprensión más profunda de las técnicas de agrupamiento aplicadas al análisis de datos.

En este contexto, el coeficiente de silueta juega un papel fundamental al evaluar la calidad del agrupamiento obtenido con K-Means. Esta métrica mide la similitud de un punto con su propio clúster en comparación con otros, generando valores entre -1 y 1: cuanto más cercano a 1, mejor está agrupado el punto; valores negativos sugieren una asignación incorrecta. La visualización de las siluetas para distintas configuraciones de clústeres permite identificar el número óptimo que maximiza la cohesión interna y la separación entre grupos, facilitando la optimización de resultados. La inclusión de este análisis refuerza la importancia de una evaluación rigurosa, guiando la selección de parámetros para asegurar un agrupamiento significativo y representativo.

2. K-Means

K-Means es un algoritmo de agrupamiento no supervisado que busca dividir un conjunto de datos en un número predefinido de clusters, donde cada cluster se caracteriza por su centroide, que es el promedio de todos los puntos asignados a ese grupo. El proceso comienza con la selección aleatoria de k centroides iniciales, seguido de la asignación de cada punto de datos al cluster cuyo centroide esté más cercano. Una vez que todos los puntos han sido asignados, se recalculan los centroides como el promedio de los puntos en cada cluster. Este proceso de asignación y actualización se repite iterativamente hasta que los centroides ya no cambian significativamente o se alcanza un número máximo de iteraciones.

El algoritmo K-Means es popular debido a su simplicidad y eficiencia, especialmente en conjuntos de datos grandes. Sin embargo, su rendimiento puede verse afectado por la elección de los centroides iniciales, lo que puede llevar a resultados subóptimos. Para mitigar este problema, se han desarrollado métodos de inicialización como K-Means++, que selecciona centroides más representativos. Además, el uso del coeficiente de silueta permite evaluar la calidad del agrupamiento, ayudando a determinar el número adecuado de clusters y a mejorar la interpretación de los resultados. A pesar de sus limitaciones, como la sensibilidad a la forma de los clusters y la necesidad de especificar el número de clusters de antemano, K-Means sigue siendo una herramienta valiosa en el análisis de datos y la minería de datos.

3. Selección del número de clusters a partir de la silueta

El método utilizado para perfilar las siluetas de los clusters en el código proviene de las funciones *silhouette_samples* y *silhouette_score* de Scikit, las cuales se utilizan para medir la calidad del agrupamiento en algoritmos como K-means. Estas funciones se basan en el coeficiente de silueta, que evalúa qué tan bien se agrupan las muestras dentro de su cluster asignado y qué tan separadas están de los clusters vecinos.

3.1. Coeficiente de silueta

El coeficiente de silueta es una métrica que cuantifica qué tan bien se ajustan los puntos dentro de su cluster y se define para cada muestra individual i de la siguiente manera:

- $a(i)$: La distancia promedio entre i y todos los demás puntos dentro del mismo cluster.
- $b(i)$: La distancia promedio entre i y los puntos en el cluster más cercano (que no sea el suyo).

El coeficiente de silueta $s(i)$ para una muestra i se calcula como:

$$s(i) = \frac{b(i) - a(i)}{\text{máx}(a(i), b(i))} \tag{1}$$

Donde $s(i) \in [-1, 1]$ y:

- Si $s(i)$ está cerca de 1, significa que la muestra está bien agrupada dentro de su cluster y está lejos de los otros clusters.
- Si $s(i)$ es cercano a 0, la muestra está en la frontera entre dos clusters.
- Si $s(i)$ es negativo, la muestra puede estar mal asignada a un cluster, ya que está más cerca de un cluster vecino.

El análisis del coeficiente de silueta es útil para elegir el número adecuado de clusters porque indica qué tan bien separados están los clusters y cuán coherentes son las asignaciones dentro de cada cluster. Al observar la silueta promedio de un conjunto de datos para diferentes cantidades de clusters, podemos comparar la calidad del agrupamiento. Un mayor valor promedio de silueta sugiere un mejor agrupamiento.

3.2. Resultados de código

En el ejemplo del código propuesto por la documentación (alterado también para mostrar el sobreajuste o *over-fitting*) se puede visualizar mejor la interpretación de estos valores.

Figura 1: Resultados con el clustering de 2 centroides

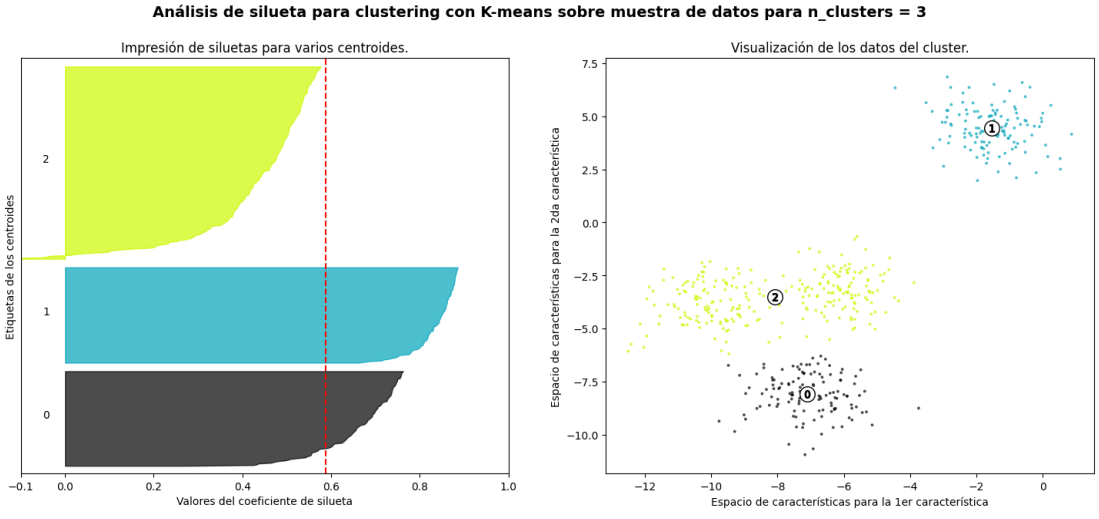

Figura 2: Resultados con el clustering de 3 centroides

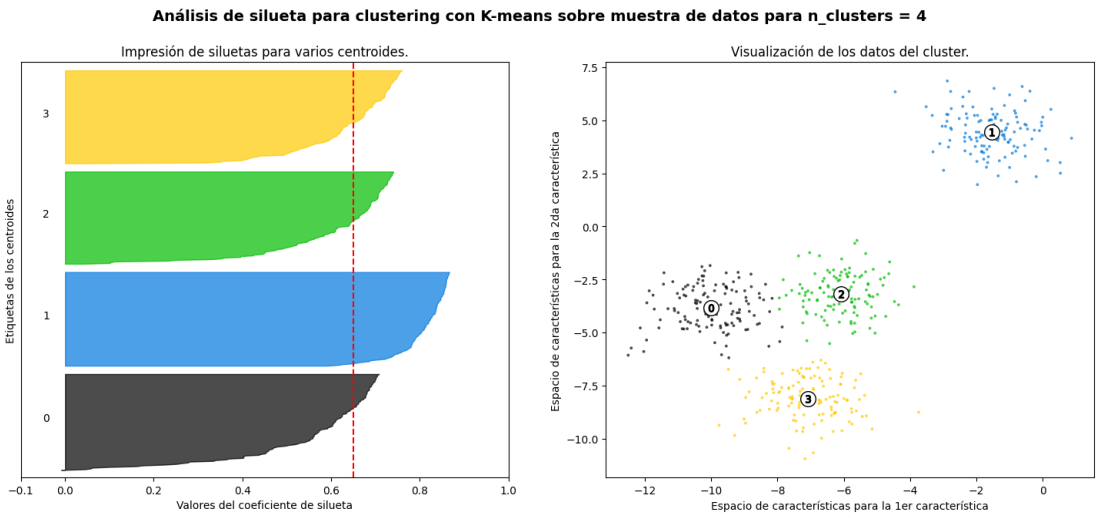

Figura 3: Resultados con el clustering de 4 centroides

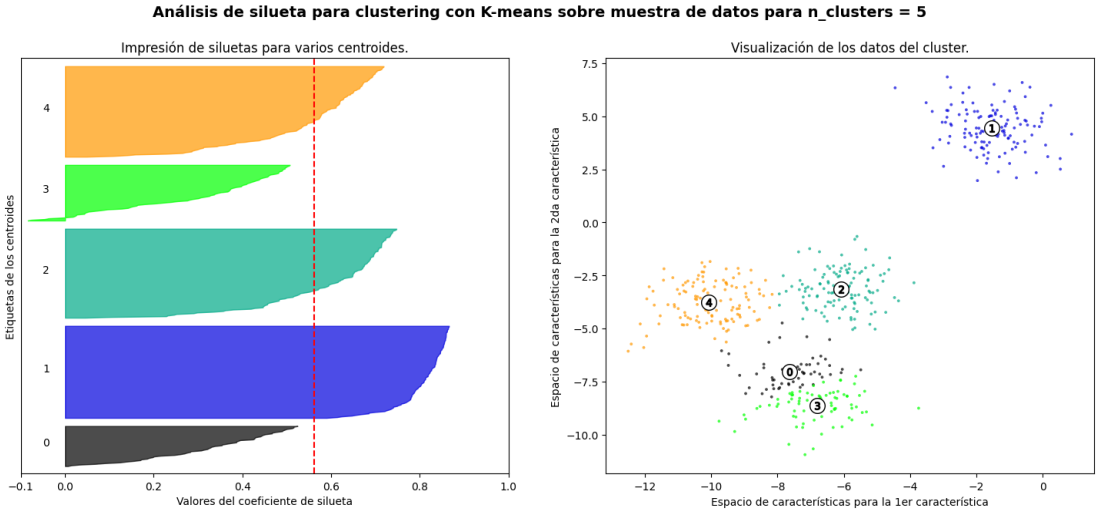

Figura 4: Resultados con el clustering de 5 centroides

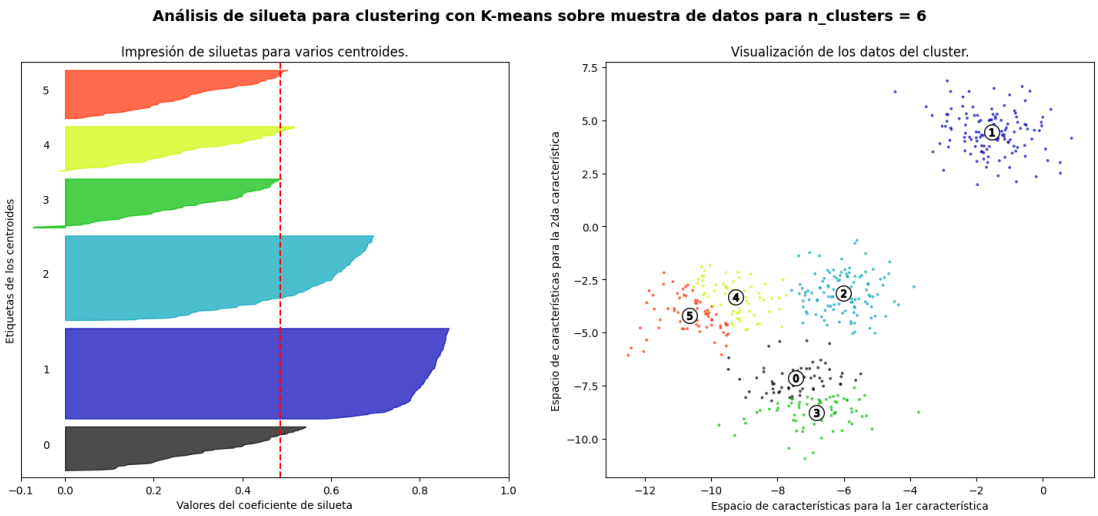

Figura 5: Resultados con el clustering de 6 centroides

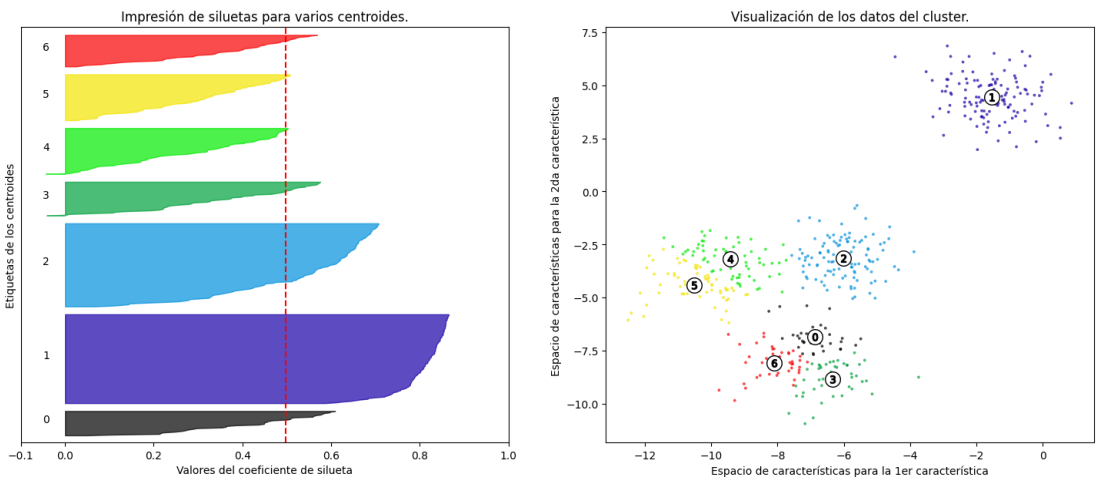

Figura 6: Resultados con el clustering de 7 centroides

A continuación, se muestran lo resultados de los puntajes de las siluetas:

```
Para n_clusters = 2 El silhouette_score promedio es : 0.7049787496083262
Para n_clusters = 3 El silhouette_score promedio es : 0.5882004012129721
Para n_clusters = 4 El silhouette_score promedio es : 0.6505186632729437
Para n_clusters = 5 El silhouette_score promedio es : 0.561464362648773
Para n_clusters = 6 El silhouette_score promedio es : 0.4857596147013469
Para n_clusters = 7 El silhouette_score promedio es : 0.49731550839901734
```

Figura 7: Resultados numéricos de los coeficientes de silueta

Al graficar las siluetas para cada cluster, obtenemos una representación visual que nos ayuda a comparar:

- Si usamos 2 clusters, las siluetas podrían mostrar valores más bajos, indicando que algunos puntos no están bien ajustados a su cluster y que podrían estar mejor divididos.
- Con 4 clusters (como en el ejemplo del código), podríamos observar que las siluetas son más uniformes y tienen un valor promedio más alto. Esto significa que la mayoría de las muestras están más cerca de su propio cluster que de cualquier otro cluster vecino, sugiriendo que este número de clusters es más adecuado.
- Si probamos con 7 clusters, podríamos observar que algunos clusters tienen siluetas muy pequeñas o incluso negativas, lo que sugiere que algunos puntos están asignados incorrectamente y que demasiados clusters podrían causar problemas de sobreajuste o asignación ineficiente.

4. Estabilidad de métodos de inicialización

Este ejemplo trata sobre la evaluación y comparación de la estabilidad de los métodos de inicialización de K-means y MiniBatchKMeans cuando se aplican a datos generados en un espacio bidimensional enfocada en en comparar dos enfoques de inicialización en modelos de clustering, en particular el algoritmo de K-means:

- K-means con inicialización k-means++
- K-means con inicialización aleatoria

Y dos versiones del modelo:

- **K-means estándar**: que se entrena en lotes completos (es decir, el modelo se ajusta con todos los datos).
- **MiniBatchKMeans**: que se entrena usando pequeños lotes de datos para mejorar el rendimiento en grandes conjuntos de datos, sacrificando ligeramente la precisión.

El objetivo de este ejemplo es ilustrar cómo estos métodos de inicialización y modelos influyen en:

- **La estabilidad de la convergencia**: al realizar múltiples ejecuciones, se observa cómo cambia la ïnercia"(una medida de calidad del ajuste) dependiendo del número de inicializaciones.
- **La calidad del agrupamiento**: representada visualmente mediante la distribución de los datos y los centros de los clusters.

4.1. Descripción general del funcionamiento

4.1.1. K-means con inicialización k-means++ y aleatoria

El algoritmo K-means agrupa los datos en un número predefinido de clusters (en este caso, igual al número de centros generados) al minimizar la distancia entre los puntos de datos y los centros de los clusters. La diferencia clave entre k-means++ y la inicialización aleatoria es la forma en que se seleccionan los puntos iniciales de los clusters:

- **Inicialización aleatoria**: selecciona los centros de forma completamente aleatoria, lo que puede conducir a una convergencia a mínimos locales subóptimos y una mayor variabilidad entre ejecuciones.
- **Inicialización k-means++**: mejora la selección inicial de los centros al elegir puntos que estén más alejados entre sí, lo que tiende a reducir la probabilidad de convergencia subóptima y, en general, mejora la calidad de la solución en comparación con la inicialización aleatoria.

4.1.2. MiniBatchKMeans

MiniBatchKMeans es una variante de K-means que entrena el modelo utilizando pequeñas porciones o "mini lotes"del conjunto de datos en lugar de usar todos los datos a la vez. Esto lo hace mucho más eficiente en términos de tiempo de cómputo y memoria, especialmente con grandes conjuntos de datos. Sin embargo, como usa solo una parte de los datos en cada iteración, su rendimiento en términos de calidad del clustering puede ser ligeramente inferior al de K-means estándar, aunque sigue siendo bastante robusto, especialmente cuando se combina con una inicialización adecuada como k-means++.

4.2. Resultados

El criterio principal que se evalúa es la inercia, que mide qué tan bien se ajustan los puntos de datos a sus clusters (es decir, la suma de las distancias al cuadrado entre cada punto de datos y su centro de cluster más cercano). En general, un valor de inercia más bajo indica un mejor ajuste.

- **Más inicializaciones mejoran la estabilidad**: El ejemplo muestra que al aumentar el número de inicializaciones del algoritmo (con el parámetro n_init), se reduce la variabilidad en la inercia. Esto es particularmente importante para la inicialización aleatoria, que es más sensible a malas selecciones iniciales de los centros.
- **La inicialización k-means++ es más estable**: Como se observa en los gráficos generados, la inicialización k-means++ resulta en una menor inercia en promedio y una menor variabilidad entre ejecuciones en comparación con la inicialización aleatoria, tanto para K-means como para MiniBatchKMeans.

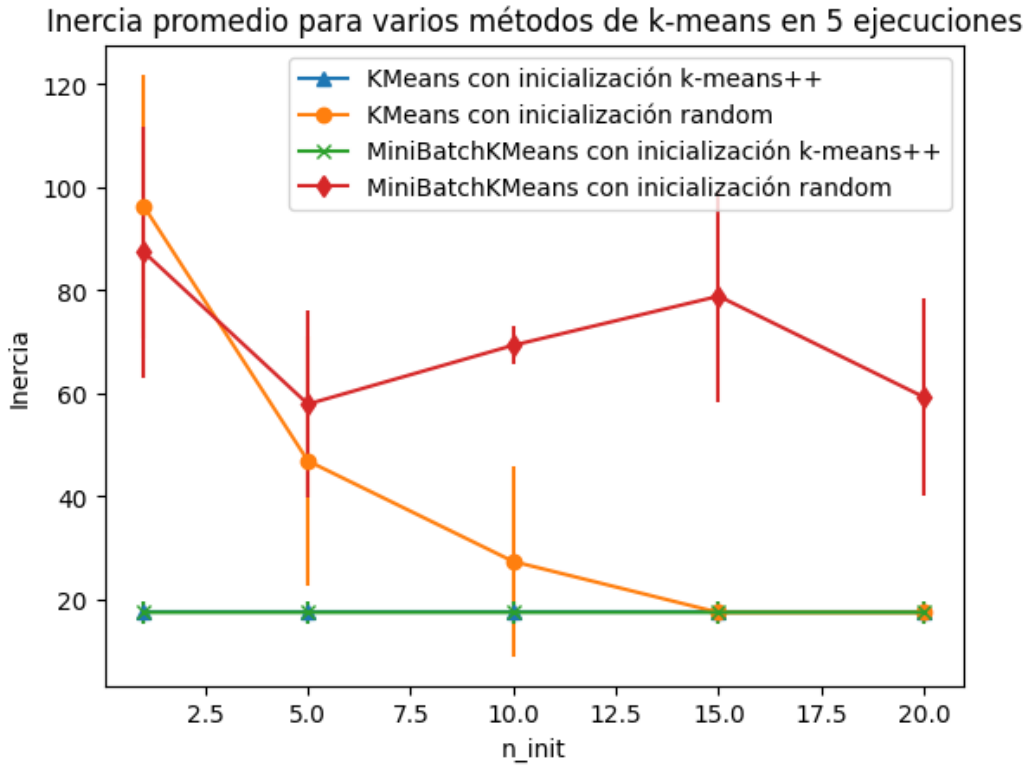

Figura 8: Resultados de la inicialización con los 4 métodos de K-means

4.3. Visualización de clusters

La segunda parte del ejemplo muestra gráficamente cómo se agrupan los datos utilizando MiniBatchKMeans con una única inicialización aleatoria. Se visualizan los puntos de datos y los centros de los clusters, lo que permite una inspección cualitativa de cómo se agrupan los puntos bajo este modelo.

- **Clusters bien formados**: A pesar de usar solo una inicialización aleatoria y el enfoque de mini lotes, los clusters resultantes se ajustan razonablemente bien a los datos generados, con los centros cercanos al "verdaderoçentro de cada cluster generado.
- **Diferencias visuales menores**: Aunque MiniBatchKMeans es un método aproximado en comparación con K-means, en este ejemplo con datos de baja dimensionalidad, las diferencias en los resultados visuales son mínimas.

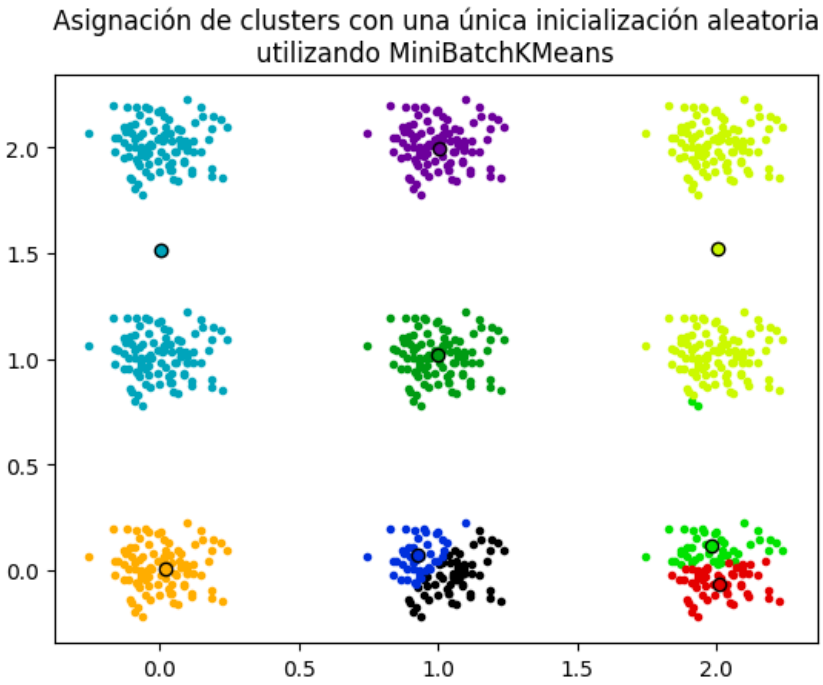

Figura 9: Visualización de los clusters formados

Cuadro 1: Comparación entre métodos de K-means y MiniBatchKMeans

Aspecto	**K-means (k-means++)**	**K-means (aleatoria)**	**MiniBatch KMeans (k-means++)**	**MiniBatch KMeans (aleatoria)**
Calidad de los clusters	Alta estabilidad, baja inercia	Mayor variabilidad, inercia más alta	Buena estabilidad, ligera pérdida de calidad	Mayor variabilidad, menor estabilidad
Velocidad	Más lento que MiniBatchKMeans	Más lento que MiniBatchKMeans	Mucho más rápido en grandes conjuntos de datos	Mucho más rápido en grandes conjuntos de datos
Uso de memoria	Alto	Alto	Bajo	Bajo
Escalabilidad	Limitada en grandes datos	Limitada en grandes datos	Escalable	Escalable

5. Cuantización de color usando K-Means

5.1. Descripción

En este ejemplo se habla sobre la reducción del uso de colores en una imagen mediante K-Means. Se reduce la cantidad de colores únicos de una imagen de 96,615 a solo 64, aplicando esta reducción sin perder la calidad de la imagen y manteniendo la "forma original", únicamente cambiando sus colores en función de los clústeres obtenidos.

5.2. Funcionamiento del código

5.2.1. Ingreso de la imagen

Este código toma como ejemplo una "muestra"de los datasets de Scikit-Learn. Se carga la imagen y se prepara para que el proceso con K-Means sea más preciso, dividiendo los valores de cada píxel entre 255, que es el rango de valores en el modelo RGB.

5.2.2. Obtención de la información de la imagen

Una vez obtenida la imagen como un arreglo 2D, se toma una muestra aleatoria de 1000 píxeles para optimizar el procesamiento de K-Means, evitando así procesar toda la imagen.

5.2.3. Tratamiento de la imagen

Una vez entrenado el modelo con la muestra aleatoria de 1000 píxeles, se etiqueta cada píxel, y según la muestra obtenida, se reconstruye la imagen.

5.3. Comparación con paleta de colores aleatoria

Para comparar el modelo, se toma una muestra aleatoria de la imagen original y se seleccionan 64 colores. Esta nueva paleta se aplica en la imagen teniendo en cuenta la distancia mínima a cada color.

5.4. Ejecución del código

Como se explicó en las secciones anteriores, se realizaron dos muestras donde se aplicó K-Means a una muestra aleatoria de 1000 píxeles, y el resultado es el siguiente:

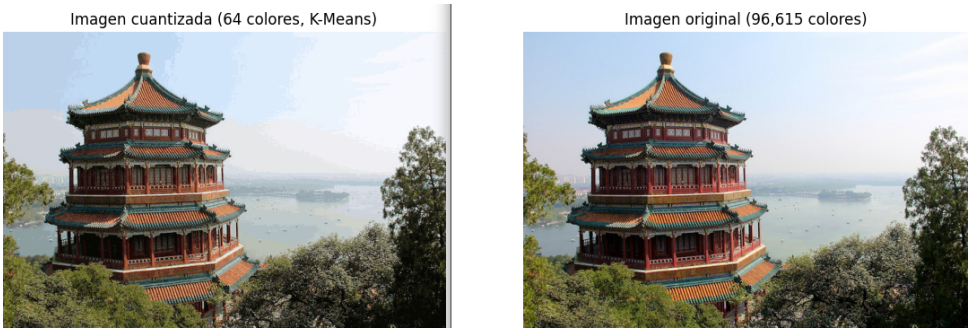

Figura 10: Comparación imagen 64 colores (K-Means) vs 96,615 colores

En comparación con la paleta de colores aleatoria, estos son los resultados:

Figura 11: Comparación imagen 64 colores (Aleatorio) vs 96,615 colores

6. Conclusiones

6.1. Hernández Jiménez Erick Yael

Con lo anterior, el coeficiente de silueta es esencial para evaluar la efectividad del agrupamiento en el algoritmo K-means. Al aplicar esta métrica, se puede determinar con precisión la calidad del clustering al medir cómo se agrupan los puntos dentro de sus clusters y cuán distintos están de los clusters vecinos. Esto refuerza la importancia de utilizar análisis de silueta para seleccionar un número de clusters que maximice la calidad del agrupamiento y mejore el rendimiento del modelo.

6.2. Robert Garayzar Arturo

En conclusión, la comparación entre los métodos de inicialización revela la importancia de elegir el método adecuado para optimizar el rendimiento del algoritmo ya que tiene un impacto significativo en la estabilidad y la calidad del clustering. La inicialización k-means++ es preferible debido a su capacidad para seleccionar puntos iniciales más representativos, resultando en una mayor estabilidad y calidad del clustering. En contraste, la inicialización aleatoria puede llevar a una mayor variabilidad y a una convergencia menos óptima. Además, para grandes conjuntos de datos, el uso de MiniBatchKMeans, ofrece una opción más eficiente en términos de tiempo y memoria, aunque con una ligera pérdida en precisión.

6.3. Patiño Flores Samuel

El uso de K-Means es muy variado, ya que puede ayudarnos en el agrupamiento. En el tratamiento de imágenes, es especialmente útil, ya que nos permite reducir la cantidad de colores sin necesidad de un proceso demasiado pesado. En otros ámbitos, incluso puede ayudarnos a identificar los colores más utilizados, eliminar fondos o detectar formas.

7. Enlace a los codigos desarrollados

`https://github.com/Yokai-Chz/Lab2-ML`