

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Inteligencia Artificial II (CI5438)

Algoritmo K-means

Estudiantes:

Giacomo Tosone 14-11085

Rommel Llanos 15-10789

Profesor: Carlos Infante

Contenidos

Contenidos	2
Introducción	3
Implementación del Algoritmo K-means	3
Aplicación en el Dataset Iris	3
Segmentación de Imágenes con K-means	4
Sobre la instalación	6
Sobre los sets de datos	7
Dataset de Iris	7
Imágenes	7
Sobre las funciones	9
k_means:	9
euclidean_distance:	9
assign_clusters:	9
calculate_centroids:	9
Resultados obtenidos	10
Parte 2: Iris Dataset	10
K = 3	10
K = 2	12
K = 4	13
K = 5	14
Parte 3: Segmentación de Imágenes	15
Parte 3.1: The_Quag.png	15
K = 3	16
K = 5	18
K = 7	19
Parte 3.2: cr7.png	22
K = 3	23
K = 5	24
K = 7	25
Conclusión	29

Introducción

En este informe, presentamos la implementación y evaluación del algoritmo de k-means para clustering. Este algoritmo es un método de aprendizaje no supervisado que se utiliza para agrupar datos en un número predefinido de clusters. Seleccionamos el dataset Iris, un conjunto de datos clásico en el campo del machine learning, para evaluar nuestra implementación. Además, aplicamos el algoritmo para la segmentación de imágenes, demostrando su versatilidad y efectividad en diferentes tipos de datos.

Implementación del Algoritmo K-means

El primer problema abordado en este proyecto es la implementación del algoritmo k-means, un método de clustering ampliamente utilizado en el aprendizaje automático no supervisado. El desafío principal radica en desarrollar un algoritmo eficiente y efectivo para agrupar un conjunto de datos en k clusters, donde k es un número predefinido.

El algoritmo k-means opera iterativamente y sigue un enfoque simple para agrupar datos basado en características numéricas. Los pasos clave del algoritmo incluyen:

Selección Inicial de Centroides: Elegir aleatoriamente k puntos del conjunto de datos como centroides iniciales.

Asignación de Datos a Clusters: Asignar cada punto de datos al cluster más cercano, basado en la distancia euclidiana entre el punto de datos y los centroides.

Actualización de Centroides: Recalcular el centroide de cada cluster como la media de todos los puntos asignados a ese cluster.

Iteración Hasta Convergencia: Repetir los pasos de asignación y actualización hasta que los centroides no cambien entre iteraciones sucesivas.

Aplicación en el Dataset Iris

El problema consiste en aplicar el algoritmo k-means implementado al dataset Iris. El dataset Iris contiene mediciones de 150 muestras de tres especies de flores de Iris, cada una con cuatro características: longitud y ancho del sépalo, y longitud y ancho del pétalo.

Preprocesamiento de Datos: Preparar el dataset para el análisis de clustering. Esto incluye la eliminación de etiquetas de clase para transformar el conjunto de datos en un formato adecuado para el aprendizaje no supervisado.

Análisis y Evaluación: Aplicar el algoritmo k-means al dataset preprocesado con diferentes valores de k (2, 3, 4, 5). El objetivo es evaluar cómo el algoritmo agrupa los datos y si estos clusters tienen alguna relación con las clases originales de las especies de Iris. Este análisis requiere una comparación entre los clusters formados y las etiquetas originales para determinar la eficacia del algoritmo en encontrar patrones naturales en los datos.

Este problema proporciona una oportunidad valiosa para evaluar la aplicabilidad y efectividad del algoritmo k-means en un conjunto de datos real y bien estudiado en el campo de la inteligencia artificial.

Segmentación de Imágenes con K-means

Este problema se centra en la aplicación del algoritmo k-means para la segmentación de imágenes. La segmentación de imágenes es un proceso crucial en el procesamiento de imágenes y visión por computadora, donde el objetivo es dividir una imagen en segmentos o regiones que sean homogéneos o significativos en cierto contexto.

Objetivos:

Representación de Datos de Imágenes: Cada píxel de una imagen puede considerarse como un punto en un espacio tridimensional (3D) basado en sus valores de intensidad en los canales de color rojo, verde y azul (RGB). La adecuada manipulación y representación de estos datos es fundamental para la efectividad del algoritmo.

Selección del Valor de K: Determinar el número óptimo de clusters (K) para segmentar la imagen. Un valor de K muy bajo puede llevar a una

sobre-generalización, mientras que un valor muy alto puede resultar en una segmentación excesiva.

Interpretación y Visualización de Resultados: Una vez que se han formado los clusters, cada píxel se reasigna al color del centroide de su cluster, lo que resulta en una versión simplificada de la imagen original. La interpretación de estos resultados y su visualización efectiva son cruciales para evaluar la calidad de la segmentación.

En este proyecto, seleccionaremos dos imágenes de diferente naturaleza y aplicaremos el algoritmo k-means con varios valores de K para segmentarlas. El análisis de las imágenes resultantes nos permitirá evaluar la efectividad del algoritmo en la segmentación de imágenes, proporcionando una visión sobre cómo diferentes valores de K afectan la calidad y utilidad de la segmentación.

Sobre la instalación

Para el correcto funcionamiento de los scripts de Python utilizados en este proyecto, es esencial seguir el siguiente procedimiento de instalación de los paquetes necesarios:

Python:

- Asegurarse de tener instalada la versión de Python 3.11 o superior.

Instalación de Módulos:

- Pandas: Ejecutar `pip3 install pandas` para instalar la biblioteca Pandas, utilizada para la manipulación y análisis de datos.
- Matplotlib: Instalar Matplotlib mediante `pip3 install matplotlib`, necesario para la visualización de datos y gráficos.

Ejecución del Script:

- El script principal, `main.py`, se encuentra en la carpeta `/src`.

Sobre los sets de datos

Este proyecto utiliza dos conjuntos de datos principales: el Dataset de Iris y dos imágenes de naturalezas distintas.

Dataset de Iris

El Dataset de Iris es un conjunto de datos clásico y ampliamente reconocido en el campo del aprendizaje automático. Este dataset contiene 150 observaciones de flores de iris, divididas en tres especies: Iris Setosa, Iris Versicolor e Iris Virginica. Cada observación incluye cuatro características: longitud y ancho del sépalo, y longitud y ancho del pétalo.

Se usará el dataset sin la columna de clases para ilustrar y comparar la capacidad del algoritmo para segmentar correctamente dadas estas características.

Imágenes

Se usarán dos imágenes, una con poca variación en los colores (imagen plana o vector) y otra con una alta variación en los colores (fotografía), de esta manera se podrá visualizar la capacidad del algoritmo de segmentar conjuntos de datos con diferentes varianzas.

La primera imagen a utilizar es una con una baja gama de colores, así como un diseño sencillo, dicha imagen se encuentra en la carpeta src bajo el nombre de "The_Quag.png", dicha imagen es la siguiente:



Para la segunda imagen, se utilizó una foto con una peor calidad de imagen, pero con una gran cantidad de colores diferentes, dicha imagen se encuentra en la carpeta de src bajo el nombre de “cr7.png” y esta imagen es la siguiente:



Sobre las funciones

En la implementación del algoritmo k-means, desarrollamos varias funciones para realizar el proceso de clustering. A continuación, describimos las funciones utilizadas:

k_means:

Objetivo: Esta es la función principal que implementa el algoritmo k-means.

Proceso:

1. **Inicialización de Centroides:** Selecciona aleatoriamente k centroides del conjunto de datos.
2. **Asignación de Clusters (assign_clusters):** Asigna cada punto de datos al cluster más cercano basándose en la distancia euclidiana.
3. **Actualización de Centroides (calculate_centroids):** Recalcula los centroides de cada cluster como el promedio de los puntos asignados a ese cluster.
4. **Iteración Hasta Convergencia:** Repite los pasos de asignación y actualización hasta que los centroides no cambien significativamente.

euclidean_distance:

Objetivo: Calcula la distancia euclidiana entre dos puntos.

assign_clusters:

Objetivo: Asigna cada punto de datos a un cluster basado en la distancia al centroide más cercano.

Proceso:

1. Calcula la distancia de cada punto a cada centroide utilizando euclidean_distance.
2. Asigna cada punto al cluster cuyo centroide es el más cercano.

calculate_centroids:

Objetivo: Recalcula los centroides de cada cluster.

Proceso:

1. Para cada cluster, si contiene puntos, calcula el promedio de estos puntos para determinar el nuevo centroide.
2. Si un cluster no tiene puntos, mantiene el centroide actual.

Resultados obtenidos

Parte 2: Iris Dataset

En esta sección se realizó un proceso de pruebas con base en el set de datos iris.csv, con la finalidad de comparar la capacidad de segmentación del algoritmo en relación a las clases originales del dataset, “Iris Setosa”, “Iris Versicolor” y “Iris Virginica”.

También se hicieron experimentos donde se segmenta en 2, 4 y 5 clusters para evaluar si pudiesen clasificarse de otra manera y si hay alguna relación entre los resultados y el dataset original.

K = 3

Se realizaron 3 corridas del algoritmo con 3 clusters para evaluar la consistencia de los resultados ya que los centroides iniciales son aleatorios.

A continuación los resultados:

En azul: Iris Setosa. En rojo: Iris Versicolor. En verde: Iris Virginica

Primera corrida:

Cluster	0	1	2
Centroide inicial	6, 2.2, 4, 1	6.1, 3, 4.6, 1.4	6.3, 2.8, 5.1, 1.5
Índices en dataset original (Mal clasificados en rojo)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49	50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 101, 106, 113, 114, 119, 121, 123, 126, 127, 133, 138, 142, 146, 149	52, 77, 100, 102, 103, 104, 105, 107, 108, 109, 110, 111, 112, 115, 116, 117, 118, 120, 122, 124, 125, 128, 129, 130, 131, 132, 134, 135, 136, 137, 139, 140, 141, 143, 144, 145, 147, 148
Centroide final	5.006, 3.418, 1.464, 0.244	5.901, 2.748, 4.393, 1.433	6.85, 3.073, 5.742, 2.071
Acierto	89.33%		
Convergencia	4 iteraciones		

Segunda corrida:

Cluster	0	1	2
Centroide inicial	4.4, 2.9, 1.4, 0.2	5, 2.3, 3.3, 1	5.8, 2.7, 5.1, 1.9
Índices en dataset original (Mal clasificados en rojo)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49	51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 101, 106, 113, 114, 119, 121, 123, 126, 127, 133, 138, 142, 146, 149	50, 52, 77, 100, 102, 103, 104, 105, 107, 108, 109, 110, 111, 112, 115, 116, 117, 118, 120, 122, 124, 125, 128, 129, 130, 131, 132, 134, 135, 136, 137, 139, 140, 141, 143, 144, 145, 147, 148
Centroide final	5.006, 3.418, 1.464, 0.244	5.883, 2.740, 4.388, 1.434	6.853, 3.076, 5.715, 2.053
Acierto	88.66%		
Convergencia	12 iteraciones		

Tercera corrida:

Cluster	0	1	2
Centroide inicial	5.8, 2.8, 5.1, 2.4	6.7, 3.1, 5.6, 2.4	7.2, 3.6, 6.1, 2.5
Índices en dataset original (Mal clasificados en rojo)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49	50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 101, 106, 113, 114, 119, 121, 123, 126, 127, 133, 138, 142, 146, 149	52, 77, 100, 102, 103, 104, 105, 107, 108, 109, 110, 111, 112, 115, 116, 117, 118, 120, 122, 124, 125, 128, 129, 130, 131, 132, 134, 135, 136, 137, 139, 140, 141, 143, 144, 145, 147, 148
Centroide final	5.006, 3.418, 1.464, 0.244	5.901, 2.748, 4.393, 1.433	6.85, 3.073, 5.742, 2.071
Acierto	89.33%		
Convergencia	8 iteraciones		

Podemos observar que el método es notablemente consistente, ya que converge a centroides con mínimas variaciones, lo que implica una definición prácticamente invariable de los clusters. No obstante, se identificaron limitaciones en la capacidad de clasificación de todas las instancias del dataset. Específicamente, se observó como en el cluster 1, correspondiente a las **Iris-Versicolor**, se clasificaron múltiples ejemplos de las **Iris-Virginica**, y en el cluster 2, correspondiente a las **Iris-Virginica**, se clasificaron múltiples ejemplos de **Iris-Versicolor**

Este patrón sugiere que la inclusión de dimensiones o características adicionales podría ser necesaria para una clasificación más precisa mediante el algoritmo K-means.

K = 2

Se realizó un experimento con 2 clusters.

A continuación los resultados:

Cluster	0	1
Centroide inicial	5.6, 3, 4.1, 1.3	6.4, 3.2, 5.3, 2.3
Índices en dataset original (Mal clasificados en rojo)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 57, 93, 98	50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 94, 95, 96, 97, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149
Centroide final	5.005, 3.360, 1.562, 0.288	6.301, 2.886, 4.958, 1.695
Convergencia	6 iteraciones	

K = 4

Se realizó un experimento con 4 clusters.

A continuación los resultados:

Cluster	0	1	2	3
Centroide inicial	5.1 3.4 1.5 0.2	5.7 2.8 4.1 1.3	5. 3.5 1.6 0.6	4.8 3. 1.4 0.3
Índices en dataset original (Mal clasificados en rojo)	0, 4, 5, 7, 10, 14, 15, 16, 17, 18, 19, 20, 21, 23, 26, 27, 28, 31, 32, 33, 36, 39, 40, 43, 44, 46, 48	50, 52, 77, 100, 102, 103, 104, 105, 107, 108, 109, 110, 111, 112, 115, 116, 117, 118, 120, 122, 124, 125, 128, 129, 130, 131, 132, 134, 135, 136, 137, 139, 140, 141, 143, 144, 145, 147, 148	51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 101, 106, 113, 114, 119, 121, 123, 126, 127, 133, 138, 142, 146, 149	1, 2, 3, 6, 8, 9, 11, 12, 13, 22, 24, 25, 29, 30, 34, 35, 37, 38, 41, 42, 45, 47, 49
Centroide final	5.255, 3.670, 1.503, 0.288	6.853, 3.076, 5.715, 2.053	5.883, 2.740, 4.388, 1.434	4.713, 3.121, 1.417, 0.191
Convergencia	16 iteraciones			

K = 5

Se realizó un experimento con 5 clusters.

A continuación los resultados:

Cluster	0	1	2	3	
Centroide inicial	5, 3.5, 1.6, 0.6	6.7, 3.1, 4.4, 1.4	6.9, 3.2, 5.7, 2.3	5.7, 2.9, 4.2, 1.3	5, 2.3, 3.3, 1
Índices en dataset original (Mal clasificados en rojo)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49	50, 51, 52, 54, 56, 58, 65, 70, 72, 75, 76, 77, 83, 86, 101, 103, 110, 111, 113, 114, 115, 116, 119, 121, 123, 126, 127, 128, 132, 133, 134, 137, 138, 141, 142, 145, 146, 147, 148, 149	100, 102, 104, 105, 107, 108, 109, 112, 117, 118, 120, 122, 124, 125, 129, 130, 131, 135, 136, 139, 140, 143, 144	55, 61, 62, 63, 66, 67, 68, 71, 73, 74, 78, 82, 84, 85, 87, 88, 90, 91, 92, 94, 95, 96, 97, 99, 106	53, 57, 59, 60, 64, 69, 79, 80, 81, 89, 93, 98
Centroide final	5.006, 3.418, 1.464, 0.244	6.322, 2.905, 5.022, 1.785	7.086, 3.126, 6.013, 2.143	5.848, 2.784, 4.316, 1.332	5.341, 2.458, 3.625, 1.133
Convergencia	16 iteraciones				

La segmentación en 2, 4 y 5 clusters muestra una clara demarcación de los datos correspondientes a la especie **Iris Setosa**, indicando que se agrupan en clusters exclusivos sin mezclarse con las otras especies. Esto resalta su distinción morfológica en comparación con las especies **Iris Versicolor** e **Iris Virginica**.

Notablemente, en la segmentación de 4 clusters, la especie **Iris Setosa** se divide de manera equitativa en dos clusters, sugiriendo la posibilidad de diferenciar dos subespecies dentro de este grupo.

Parte 3: Segmentación de Imágenes

Para esta parte se utilizaron las imágenes mencionadas anteriormente en el apartado de set de datos para luego transformar estas en unas matrices $N \times M$ en donde cada elemento representa a un pixel de la imagen en su representación rgb, luego de esto dichas matrices son utilizadas para para calcular sus k centroides por medio del algoritmo k-mean, los k centroides resultantes representan k pixeles los cuales se consideran como representantes del resto de píxeles en dichas imágenes. Una vez obtenidos dichos resultados, se puede re-renderizar las imágenes sustituyendo a los pixeles con sus respectivos valores de los centroides, obteniendo como resultado una nueva imagen representada exclusivamente por k colores.

En vista que para cada imagen se realizan diferentes pruebas y cada imagen tiene ciertas particularidades, los resultados obtenidos por imagen se van a discutir en secciones separadas.

Parte 3.1: The_Quag.png


La imagen utilizada para este experimento se encuentra bajo el nombre de "The_Quag.png", esta imagen posee las dimensiones de 1109×1200 por lo cual cuenta con un total de 1330800 pixeles, por lo cual a pesar de ser una imagen sencilla el set de datos es bastante amplio, lo cual ocasionó que en algunos casos la cantidad necesaria de re-selecciones de los centroides incrementara exponencialmente al aumentar el k , con $k = 3$ el programa podría concluir en 4 selecciones, mientras que con $k = 7$ el programa necesitaba de 20 selecciones de centroides para concluir. Para realizar estos experimentos se utilizó a la siguiente imagen:





Para esta imagen se realizaron 3 experimentos variando la cantidad de k para poder representar esta imagen utilizando k colores diferentes, adicional a esto se utilizó la librería opencv (pip install opencv-python) en python para manejar la carga de la imagen, así como su representación en píxeles. Por último todas las diferentes manipulaciones de la matriz de píxeles fueron realizadas utilizando pandas. Los resultados obtenidos son los siguientes:

K = 3

Para este primer experimento se utilizó el algoritmo k_means para calcular 3 centroides para representar la imagen, los centroides obtenidos y sus colores adyacentes son los siguientes:

-  [225.18433407 205.39587705 152.98128483]

-  [173.11093754 159.35741859 134.40142444]
-  [1.24688166 1.27597424 0.95364477]

Utilizando estos resultados se puede re-renderizar utilizando los colores de los centroides, la imagen resultante es la siguiente:


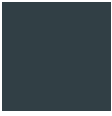


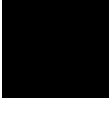


Podemos ver que los detalles de la imagen se pierden completamente debido a la limitante de solo utilizar 3 tonalidades, podemos ver que en esta representación el algoritmo trata de englobar las 3 tonalidades más prominentes en la figura, sin embargo no utiliza exactamente las tonalidades más representadas, sino unas que permitan que los errores se minimicen, esto se puede ver referenciado en el tercer centroide el cual representa al color negro, revisando los valores mas comunes de la matriz, nos encontramos que el color negro [0,0,0] es el valor más repetido en todo el dataset con un total de 704448

repeticiones pero el centroide que lo representa es el [1.24688166, 1.27597424, 0.95364477]

K = 5

Similar al experimento anterior, se utilizó el algoritmo k_means para calcular 5 centroides para representar la imagen, los centroides obtenidos y sus colores adyacentes son los siguientes:

-  [2.25225428e+02, 2.05433790e+02, 1.53006089e+02]
-  [6.93383785e+01, 6.26685558e+01, 4.89928657e+01]
-  [1.73694422e+02, 1.59890613e+02, 1.34787981e+02]
-  [5.52891514e+00, 8.19804552e+00, 6.44161306e+00]
-  [4.21465555e-03, 2.38496768e-02, 1.61775655e-02]


La imagen resultante de re-renderizar la imagen original con los colores de los centroides es la siguiente:





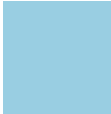



Podemos ver que en este segundo caso al tener una mayor cantidad de centroides disponibles se presentan una mayor cantidad de detalles, en particular se presentan 2 centroides que ambos representan al color negro y una de estas el $[4.21465555e-03, 2.38496768e-02, 1.61775655e-02]$ ya se acerca mucho al $[0,0,0]$ que es el valor que es el valor mas representado en el set de datos.





K = 7




Para este último experimento con esta imagen, se quiere aumentar el k a 7 para ver de qué forma se representan los pixeles con más opciones, este fué el último resultado obtenido con este set de datos debido a que el tamaño complica mucho el cálculo si se aumenta mas el k, los centroides obtenidos son los siguientes:

-  $[7.29237078e+01, 6.54810022e+01, 4.97323001e+01]$

-  [1.21711002e+02, 1.22288415e+02, 1.79777572e+02]
-  [3.34437799e+01, 3.26619617e+01, 2.82801435e+01]
-  [3.52850682e+00, 6.44460170e+00, 4.85981970e+00]
-  [3.66591744e-04, 7.96169329e-03, 5.14785395e-03]
-  [2.25599769e+02, 2.05779813e+02, 1.53227563e+02]
-  [1.80116953e+02, 1.64576817e+02, 1.29549603e+02]

Para este experimento podemos ver que la inclusión de 2 centroides más permite que las tonalidades de rosado y azul puedan ser mejor representadas, en particular la distribución de valores en cada centroide es la siguiente:

Centroide	Color	Cantidad
0		13291
1		37720
2		4180
3		32501

4		706508
5		217197
6		319403

Podemos ver que con 7 centroides el algoritmo puede agrupar los datos de manera eficiente mas no uniformemente, podemos ver que con esta distribución ya comienza a asemejarse a la imagen original teniendo una distribución similar, en particular en los apartados de azul y negro, por ultimo la imagen resultante de usar estos centroides para representar la imagen original es la siguiente:



Parte 3.2: cr7.png

La imagen utilizada para este experimento se encuentra bajo el nombre de “cr7.png”, esta imagen posee las dimensiones de 371*640 por lo cual cuenta con un total de 237440 pixeles, esta imagen cuenta con una cantidad de píxeles menor a la utilizada en el experimento anterior, sin embargo la complejidad de los colores de esta imagen hace que la cantidad necesaria de re-selecciones de los


centroides es muy superior al experimento anterior para k bajos, para el caso de $k = 3$ se necesitan 20 de re-selecciones y pero la cantidad de re-selecciones parece bajar a medida que se aumenta el k , esto puede deberse a la cantidad de colores diferentes presentes en la imagen. Para realizar estos experimentos se utilizó a la siguiente imagen:





Similar al experimento anterior, para esta imagen se realizaron 3 experimentos variando la cantidad de k para poder representar esta imagen utilizando k colores diferentes, adicional a esto se utilizó la librería opencv (pip install opencv-python) en python para manejar la carga de la imagen, así como su representación en píxeles. Por último todas las diferentes manipulaciones de la matriz de píxeles fueron realizadas utilizando pandas. Los resultados obtenidos son los siguientes:

K = 3

Para este primer experimento se utilizó el algoritmo `k_means` para calcular 3 centroides para representar la imagen, los centroides obtenidos y sus colores adyacentes son los siguientes:

-  [83.64680825 99.54955628 113.25691903]

-  [53.31749197 44.90058013 29.59127471]
-  [24.61696726 12.06798284 219.39240063]


Utilizando estos centroides, se puede re-renderizar la imagen base, obteniendo como resultado la siguiente imagen:







Podemos observar que la imagen resultante resulta muy sencilla, pero lo sorprendente de esta es la cantidad de colores que el algoritmo puede representar con solo 3 colores, esto debido a que la mayoría de estos son exclusivamente representados con el color gris.

K = 5

Similar al ejemplo anterior, este primer experimento se utilizó el algoritmo k_means para calcular 5 centroides para representar la imagen, los centroides obtenidos y sus colores adyacentes son los siguientes:

-  [48.64206628 35.80568216 13.24325027]

-  [62.40057005 62.68718651 61.48678971]
-  [80.1407754 97.28567609 108.38850267]
-  [22.69634918 10.1575284 218.71363779]
-  [132.12092624 159.88507719 207.84934248]








Similar a los casos anteriores, la imagen resultante de utilizar los centroides para representar la imagen base es la siguiente:



En este resultado podemos ver como el recién representado color carne se usa como el color para representar los detalles, punto que anteriormente era representado por el color gris, otro punto interesante es que similar a los experimentos anteriores con k elevado, el valor más prominente (en este caso gris) es representado por 2 tonalidades, sin embargo por la cantidad de detalles en esta imagen, la diferencia en tonalidades de gris es evidente.








K = 7

Para este último experimento con esta imagen, se quiere aumentar el k a 7 para ver de qué forma se representan los pixeles con más opciones, este fué el último resultado obtenido con este set de datos debido a que se busca realizar paralelismos con el experimento de la imagen anterior con el mismo k, luego de realizar el experimento se obtuvieron los siguientes centroides:

-  [28.17165261 13.28299049 233.38879639]
-  [5.08281032 3.58612273 135.36488842]
-  [133.72866217 163.19620881 206.37842398]
-  [61.26590222 63.2839915 64.64399844]
-  [43.24675455 32.7219061 14.26116608]
-  [81.134897 98.79259637 109.51064654]
-  [90.12165614 64.15042344 10.63792176]

Para este experimento podemos ver que la inclusión de 2 centroides más permite que las tonalidades de rojo y gris puedan ser mejor representadas, así

como ahora aparece la tonalidad de azul, en particular la distribución de valores en cada centroide es la siguiente:

Centroide	Color	Cantidad
0		18619
1		5736
2		10076
3		69157
4		69944
5		49030
6		14878

En comparación con el ejemplo de $k = 7$ anterior, este caso tiene una distribución más uniforme en sus centroides, esto debido a que esta imagen posee más heterogeneidad de colores posibles, podemos ver que el color gris es muy prominente en este dataset siendo representado por 3 centroides diferentes, otro punto interesante es que en este ejemplo cada aumento de k es representado por un aumento de tonalidades distinta y fáciles de diferenciar, en particular al aumentar el k , podemos ver que tenemos un color rojo y uno vinotinto mientras

que antes solo se tenía un color rojo. Por último la imagen resultante de representar la imagen original con los colores de los centroides es la siguiente.



Conclusión

Podemos concluir que la eficacia del algoritmo K-means tanto en el dataset de Iris como en la segmentación de imágenes, destaca por la consistencia del algoritmo al converger a centroides con variaciones mínimas, lo que resulta en una definición uniforme (en casos en donde la data está bien distribuida) de los clusters. Sin embargo, se identificaron limitaciones en la clasificación precisa de todas las instancias del dataset de Iris, especialmente en la diferenciación entre **Iris-Versicolor** e **Iris-Virginica**. Se sugiere la necesidad de considerar dimensiones o características adicionales para mejorar la clasificación. Además, se observa una clara segmentación de los datos de **Iris Setosa** en su propio cluster, lo que resalta su distinción morfológica en comparación con las otras especies. En la segmentación de 4 clusters, **Iris Setosa** se divide equitativamente, lo que sugiere la posibilidad de diferenciarla en dos subespecies.

En la segmentación de imágenes se observó que con un aumento en el valor de K, el algoritmo podía capturar mejor los detalles y colores variados de las imágenes. Este análisis demuestra la capacidad del algoritmo K-means para segmentar imágenes en representaciones simplificadas, manteniendo en algunos casos características visuales clave. El estudio sugiere que la elección del valor óptimo de K es crucial para equilibrar la simplificación y la representatividad de los colores en la segmentación de imágenes sin comprometer el rendimiento del algoritmo y el tiempo de procesamiento de las imágenes, en particular es primordial que el valor de k sea lo suficientemente grande poder obtener las diferentes tonalidades claves de la imagen y lo suficientemente pequeño para no sobre segmentar las tonalidades más comunes.