Documentación del algoritmo no supervisado: Clustering con K-Means

1. Objetivo del Proyecto

El objetivo de este proyecto es aplicar el algoritmo de clustering no supervisado **K-Means** para agrupar datos en varios clusters y visualizar los resultados. Este enfoque es útil para identificar patrones y segmentaciones dentro de conjuntos de datos sin etiquetas. En este caso, utilizamos datos generados sintéticamente para probar la eficacia del algoritmo.

2. Algoritmo Seleccionado: K-Means Clustering

¿Por qué K-Means?

K-Means es un algoritmo popular para el análisis de clustering debido a su simplicidad, velocidad y capacidad para manejar grandes volúmenes de datos. Es particularmente adecuado para conjuntos de datos donde se espera que los puntos de datos se agrupen en torno a centros bien definidos, lo cual es el caso en este ejemplo. K-Means tiene las siguientes ventajas para este proyecto:

- **Facilidad de interpretación**: Al asignar cada punto de datos al cluster más cercano, permite una visualización clara de los grupos.
- **Eficiencia**: K-Means es computacionalmente eficiente y escalable, lo cual lo hace adecuado para conjuntos de datos de tamaño moderado a grande.
- **Aplicabilidad**: Es ideal para problemas donde se pueden esperar clusters con una forma compacta (simétrica y no entrelazados).

Este algoritmo agrupa los datos en k clusters al minimizar la distancia entre los puntos y sus respectivos centroides, proporcionando una asignación clara y estructurada para la interpretación visual.

3. Preparación del Conjunto de Datos

Para este proyecto, se ha utilizado el conjunto de datos sintético make_blobs de Scikit-Learn, que genera datos aleatorios distribuidos alrededor de un número predefinido de centros. Este conjunto de datos es adecuado para K-Means ya que permite controlar características como el número de clusters y la dispersión de los puntos de datos alrededor de cada centro.

Generación del conjunto de datos:

```python
from sklearn.datasets import make_blobs
# Generamos un conjunto de datos con 300 muestras, 4 centros y 2 características
X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
```

Además, los datos se escalan usando **StandardScaler (**La clase **sklearn.preprocessing.StandardScaler** estándariza los datos eliminando la media y escalando los datos de forma que su varianza sea igual a 1.**)** para mejorar el rendimiento del algoritmo y asegurar que las características tengan una magnitud comparable.

4. Implementación de K-Means en Scikit-Learn

El algoritmo K-Means se implementa con la clase K-Means de Scikit-Learn. Los pasos clave incluyen:

1. **Inicialización del Modelo**: Se define el número de clusters (n_clusters=4) en función de los datos generados.
2. **Entrenamiento del Modelo**: Se ajusta el modelo a los datos escalados.
3. **Predicción**: Se generan etiquetas para cada punto de datos en función del cluster al que pertenece.
4. **Visualización de los Clusters y sus Centros**: Se grafican los puntos de datos con colores que representan los clusters y se destacan los centroides.

```python
from sklearn.cluster import KMeans
# Definimos el modelo K-Means con el número de clusters que queremos
kmeans = KMeans(n_clusters=4)
# Ajustamos el modelo a los datos
kmeans.fit(X_scaled)

# Obtenemos las etiquetas de cada punto de datos y los centros de los clusters
y_kmeans = kmeans.predict(X_scaled)
centers = kmeans.cluster_centers_
```

5. Visualización de Resultados

Para interpretar los resultados de K-Means, se utiliza un gráfico de dispersión, donde cada color representa un cluster distinto. Los centroides, que representan el centro de cada cluster, se destacan con un marcador X rojo, lo que facilita observar cómo el algoritmo ha agrupado los puntos en función de sus características.

Código de Visualización:

```
import matplotlib.pyplot as plt
# Nos sirve para la visualización de los clusters y los centros
plt.figure(figsize=(8, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y_kmeans, s=50, cmap='viridis') # puntos de datos

# Marcamos los centros de los clusters
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X', label='Centroides')
plt.title("K-Means Clustering")
plt.xlabel("Caracteristica 1")
plt.ylabel("Caracteristica 2")
plt.legend()
plt.show()
```

Interpretación del Gráfico:

En el gráfico generado:

- **Puntos de datos**: Los puntos están coloreados en función del cluster al que pertenecen. Cada color representa un grupo de puntos que el algoritmo considera similar.
- **Centroides**: Los marcadores X en rojo representan los centros de cada cluster. Estos centroides son puntos medios calculados por el algoritmo y ayudan a visualizar la cohesión y separación de los clusters.

6. Conclusiones

A través de este proyecto, se han logrado los siguientes objetivos:

- **Aplicación práctica de K-Means**: Se ha demostrado cómo aplicar K-Means en un conjunto de datos no supervisado, lo cual es útil para segmentación de clientes, análisis de patrones y otras aplicaciones.
- **Interpretación visual**: La visualización permite interpretar fácilmente los clusters y evaluar la efectividad de K-Means para agrupar los datos.
- **Justificación de K-Means**: Los resultados obtenidos con K-Means son adecuados dado que los datos presentan una estructura de clusters compacta, la cual es capturada eficientemente por este algoritmo.