Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

3.12. Análisis de Clustering con K-medias

Ejercicio 107_01. Clustering simple con K-medias en Python

```python
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Cargar los datos desde el CSV
data = pd.read_csv("tu_archivo.csv")

# Seleccionar las columnas relevantes (ajusta según tu dataset)
X = data[['columna1', 'columna2', ...]]

# Crear un rango de valores para el número de clusters
range_n_clusters = [2, 3, 4, 5, 6]

for n_clusters in range_n_clusters:
    # Crear y ajustar el modelo KMeans
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(X)

    # Calcular la puntuación de silueta
    silhouette_avg = silhouette_score(X, kmeans.labels_)
    print("Para n_clusters =", n_clusters,
          "La puntuación de silueta promedio es :", silhouette_avg)
```

Ejercicio 107_02. Clustering simple con K-medias en MATLAB

```matlab
% Cargar los datos desde el CSV
data = csvread('tu_archivo.csv');

% Seleccionar las columnas relevantes (ajusta según tu dataset)
X = data(:, [1, 2, ...]); % Ajusta los índices de las columnas

% Crear un rango de valores para el número de clusters
range_n_clusters = [2, 3, 4, 5, 6];

for n_clusters = range_n_clusters
    % Crear y ajustar el modelo KMeans
    opts = statset('Display','final');
    [idx, C, sumd, D] = kmeans(X, n_clusters, 'Replicates',5, 'Options',opts);

    % Calcular la puntuación de silueta
    silhouette_avg = mean(silhouette(X,idx));
    fprintf('Para n_clusters = %d, la puntuación de silueta promedio es: %f\n',
n_clusters, silhouette_avg);
end
```

Explicación:

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

1. **Carga de datos:** Se carga el archivo CSV utilizando pandas en Python y csvread en MATLAB.
2. **Selección de características:** Se seleccionan las columnas relevantes para el clustering.
3. **Rango de clusters:** Se define un rango de valores para el número de clusters.
4. **Creación y ajuste del modelo:** Se crea un objeto KMeans y se ajusta al conjunto de datos.
5. **Evaluación:** Se calcula la puntuación de la silueta para evaluar la calidad de cada clustering.

Ejercicio 108_01. Clustering con K-medias con normalización, visualización y métricas en Python

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score,
davies_bouldin_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Cargar los datos y normalizarlos
data = pd.read_csv("tu_archivo.csv")
X = data[['columna1', 'columna2', ...]]  # Seleccionar las columnas relevantes
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Rango de valores para el número de clusters
range_n_clusters = [2, 3, 4, 5, 6]

# Diccionarios para almacenar las métricas
silhouette_avg_scores = {}
calinski_harabasz_scores = {}
davies_bouldin_scores = {}

# Iterar sobre el rango de clusters
for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(X_scaled)

    silhouette_avg = silhouette_score(X_scaled, kmeans.labels_)
    calinski_harabasz = calinski_harabasz_score(X_scaled, kmeans.labels_)
    davies_bouldin = davies_bouldin_score(X_scaled, kmeans.labels_)

    silhouette_avg_scores[n_clusters] = silhouette_avg
    calinski_harabasz_scores[n_clusters] = calinski_harabasz
    davies_bouldin_scores[n_clusters] = davies_bouldin

# Visualizar las métricas
```

```
plt.plot(list(silhouette_avg_scores.keys()), list(silhouette_avg_scores.values()))
plt.xlabel('Número de clusters')
plt.ylabel('Puntuación de silueta')
plt.title('Evolución de la puntuación de silueta')
plt.show()

# Visualizar los clusters (ejemplo con 2 dimensiones)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=kmeans.labels_)
plt.title('Clustering con KMeans')
plt.show()
```

Ejercicio 108_02. Clustering con K-medias con normalización, visualización y métricas en MATLAB

```
% Cargar los datos y normalizar
data = csvread('tu_archivo.csv');
X = data(:, [1, 2, ...]); % Ajusta los índices de las columnas
X_scaled = zscore(X);

% Rango de valores para el número de clusters
range_n_clusters = 2:6;

% Inicializar variables para almacenar métricas
silhouette_scores = zeros(size(range_n_clusters));
calinski_harabasz_scores = zeros(size(range_n_clusters));
davies_bouldin_scores = zeros(size(range_n_clusters));

% Iterar sobre diferentes números de clusters
for i = 1:length(range_n_clusters)
    n_clusters = range_n_clusters(i);

    % Crear y ajustar el modelo KMeans
    opts = statset('Display','final');
    [idx, C, sumd, D] = kmeans(X_scaled, n_clusters, 'Replicates',5,
'Options',opts);

    % Calcular métricas
    silhouette_scores(i) = mean(silhouette(X_scaled,idx));
    calinski_harabasz_scores(i) = evalclusters(X_scaled, idx, 'CalinskiHarabasz');
    davies_bouldin_scores(i) = evalclusters(X_scaled, idx, 'DaviesBouldin');

    % Visualización de los clusters (opcional)
    figure;
    gscatter(X_scaled(:,1), X_scaled(:,2), idx);
    title(['Clustering con K = ', num2str(n_clusters)]);
    xlabel('Característica 1');
    ylabel('Característica 2');
end

% Visualizar las métricas
figure;
plot(range_n_clusters, silhouette_scores, '.-', ...
     range_n_clusters, calinski_harabasz_scores, '.-', ...
     range_n_clusters, davies_bouldin_scores, '.-');
```

```
legend('Silhouette', 'Calinski-Harabasz', 'Davies-Bouldin');
xlabel('Número de clusters');
ylabel('Valor de la métrica');
title('Comparación de métricas');
```

Ejercicio 109_01. Ejemplo de clustering con función-script en Python

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score,
davies_bouldin_score
from scipy.spatial.distance import pdist, squareform
from scipy.stats import entropy

# Función para calcular el índice de Dunn
def dunn_index(X, labels):
    distances = squareform(pdist(X))
    unique_labels = np.unique(labels)
    inter_cluster_distances = []
    intra_cluster_distances = []

    for i in unique_labels:
        cluster_i = X[labels == i]
        other_clusters = X[labels != i]
        inter_cluster_distances.append(np.min(pdist(np.vstack([cluster_i,
other_clusters]))))
        intra_cluster_distances.append(np.max(pdist(cluster_i)))

    dunn = np.min(inter_cluster_distances) / np.max(intra_cluster_distances)
    return dunn

# Función principal para realizar clustering K-means y calcular métricas
def clustering_kmeans(data, range_n_clusters, range_replicates):
    # Normalizar los datos
    X_scaled = (data - np.mean(data, axis=0)) / np.std(data, axis=0)

    # Inicializar matrices para almacenar las métricas
    silhouette_scores = np.zeros((len(range_n_clusters), len(range_replicates)))
    calinski_harabasz_scores = np.zeros((len(range_n_clusters),
len(range_replicates)))
    davies_bouldin_scores = np.zeros((len(range_n_clusters),
len(range_replicates)))
    entropy_scores = np.zeros((len(range_n_clusters), len(range_replicates)))
    dunn_scores = np.zeros((len(range_n_clusters), len(range_replicates)))

    # Iterar sobre diferentes números de clusters y réplicas
    for i, n_clusters in enumerate(range_n_clusters):
        for j, replicates in enumerate(range_replicates):
            # Crear y ajustar el modelo KMeans
```

```
            kmeans = KMeans(n_clusters=n_clusters, n_init=replicates,
random_state=0)
            labels = kmeans.fit_predict(X_scaled)

            # Calcular métricas
            silhouette_scores[i, j] = silhouette_score(X_scaled, labels)
            calinski_harabasz_scores[i, j] = calinski_harabasz_score(X_scaled,
labels)
            davies_bouldin_scores[i, j] = davies_bouldin_score(X_scaled, labels)
            entropy_scores[i, j] = entropy(np.bincount(labels) / len(labels))  #
Normalizar para calcular entropía
            dunn_scores[i, j] = dunn_index(X_scaled, labels)

    return silhouette_scores, calinski_harabasz_scores, davies_bouldin_scores,
entropy_scores, dunn_scores

# SCRIPT
import pandas as pd
import matplotlib.pyplot as plt

# Cargar datos
data = pd.read_csv('mydata.csv').values

# Definir rangos
range_n_clusters = range(2, 7)
range_replicates = [5, 10, 20]

# Aplicar clustering y calcular métricas
silhouette_scores, calinski_harabasz_scores, davies_bouldin_scores, entropy_scores,
dunn_scores = clustering_kmeans(data, range_n_clusters, range_replicates)

# Visualizar resultados (ejemplo)
plt.plot(range_n_clusters, np.mean(silhouette_scores, axis=1), 'o-')
plt.xlabel('Número de clusters')
plt.ylabel('Coeficiente de silueta promedio')
plt.show()
```

Ejercicio 109_02. Ejemplo de clustering función-script en MATLAB

```
function [silhouette_scores, calinski_harabasz_scores, davies_bouldin_scores,
entropy_scores, dunn_scores] = clustering_kmeans(data, range_n_clusters,
range_replicates)
    % clustering_kmeans: Realiza clustering K-means y calcula múltiples métricas
    %
    %   Entrada:
    %       data: Matriz de datos
    %       range_n_clusters: Vector con el rango de número de clusters
    %       range_replicates: Vector con el rango de réplicas
    %
    %   Salida:
    %       silhouette_scores: Matriz con los valores del coeficiente de silueta
    %       calinski_harabasz_scores: Matriz con los valores del índice de
Calinski-Harabasz
```

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

```
    %       davies_bouldin_scores: Matriz con los valores del índice de Davies-
Bouldin
    %       entropy_scores: Matriz con los valores de entropía
    %       dunn_scores: Matriz con los valores del índice de Dunn

    % Normalizar los datos
    X_scaled = zscore(data);

    % Inicializar matrices para almacenar las métricas
    silhouette_scores = zeros(length(range_n_clusters), length(range_replicates));
    calinski_harabasz_scores = zeros(length(range_n_clusters),
length(range_replicates));
    davies_bouldin_scores = zeros(length(range_n_clusters),
length(range_replicates));
    entropy_scores = zeros(length(range_n_clusters), length(range_replicates));
    dunn_scores = zeros(length(range_n_clusters), length(range_replicates));

    % Función para calcular el índice de Dunn
    dunn_index = @(X, idx) min(pdist2(X(idx==i,:), X(idx~=i,:),'euclidean'),[],1);

    % Iterar sobre diferentes números de clusters y réplicas
    for i = 1:length(range_n_clusters)
        for j = 1:length(range_replicates)
            n_clusters = range_n_clusters(i);
            replicates = range_replicates(j);

            % Crear y ajustar el modelo KMeans
            opts = statset('Display','final');
            [idx, C, sumd, D] = kmeans(X_scaled, n_clusters,
'Replicates',replicates, 'Options',opts);

            % Calcular métricas
            silhouette_scores(i,j) = mean(silhouette(X_scaled,idx));
            calinski_harabasz_scores(i,j) = evalclusters(X_scaled, idx,
'CalinskiHarabasz');
            davies_bouldin_scores(i,j) = evalclusters(X_scaled, idx,
'DaviesBouldin');
            entropy_scores(i,j) = entropy(idx);
            dunn_scores(i,j) = max(dunn_index(X_scaled, idx));
        end
    end
end

% Script
% Cargar datos
data = csvread('mydata.csv');

% Definir rangos
range_n_clusters = 2:6;
range_replicates = [5, 10, 20];

% Aplicar clustering y calcular métricas
[silhouette_scores, calinski_harabasz_scores, ...] = clustering_kmeans(data,
range_n_clusters, range_replicates);
```

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

```
% Visualizar resultados (ejemplo)
figure;
plot(range_n_clusters, mean(silhouette_scores, 2), 'o-');
xlabel('Número de clusters');
ylabel('Coeficiente de silueta promedio');
```