CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

APRENDIZAJE INTELIGENTE

6° "A"

PRÁCTICA 5: IMPLEMENTACIÓN DE K-MEDIAS

Profesor: Francisco Javier Luna Rosas

Alumnos:

Espinoza Sánchez Joel Alejandro

Gómez Garza Dariana

González Arenas Fernando Francisco

Fecha de Entrega: Aguascalientes, Ags.**, 24** de febrero de 2021

Práctica 5: Implementación de K-Medias

Objetivo:

Mediante el desarrollo de esta práctica, implementar el método de clustering denominado como K-Means.

Introducción:

Los métodos de clustering se agrupan dentro de las técnicas de machine learning y de aprendizaje no supervisado basados en agrupar o identificar clústeres (subconjuntos similares entre sí) dentro de un conjunto de datos, de acuerdo a una determinada medida de similitud entre las observaciones, pudiendo obtener diferentes clústeres en función de la mediad utilizada. La finalidad pues, es la de particionar los datos en distintos grupos de manera que las observaciones dentro de cada grupo sean bastante similares entre sí, y distintas a otros grupos. El concepto de “similar” dependerá del caso de estudio. Podemos agrupar observaciones en base a las variables disponibles, o agrupar las variables en base a las observaciones.

El método de clustering se relaciona con el análisis de componentes principales en el sentido de que ambos buscan simplificar los datos, aunque el mecanismo de ambos es distinto: mientras que PCA pretende encontrar una representación de los datos en pocas dimensiones que expliquen gran parte de la varianza, el método de clustering se aplica para encontrar subgrupos homogéneos de observaciones.

Siendo un método de data mining bastante popular en muchos campos, existe un gran número de métodos de clustering, siendo dos de los más conocidos:

- K-means clustering: partición de las observaciones en un número predefinido de clústeres.
- Hierarchical clustering: no partimos de un número predefinido de clústeres. Representación de datos en un dendograma (representación en forma de árbol).

En la presente práctica se tratará de implementar en Python el Clustering por medio del cálculo de K-Medias.

Pregunta de Investigación:

¿Cómo se puede implementar método de clustering en Python?

Predicción:

Creemos que la implementación consistirá en buscar funciones equivalentes a las existentes en R para trabajar ahora en Python.

Materiales:

Una computadora con Python y el entorno Anaconda.

Método (Variables):

Dependiente: La predicción que se realizará obtenida por las gráficas.
Independiente: Los datos a tratar con el conjunto de datos de estudiantes.
Controlada: El algoritmo de K-Means a implementar.

Seguridad:

Realmente no se trabajó en campo, por lo que no se corren riesgos con la práctica.

Procedimiento:

1.- El programa inicia al encontrar el archivo con el que se trabajará para hacer el algoritmo K-Medias. De él se imprimen los datos para revisar que se están leyendo correctamente.

```
#Estudiantes
import pandas as pd
os.chdir("C:/Users/Dell/Desktop/KMeans") #Ruta a cambiar donde se ubica tanto el .py como el .csv

datos = pd.read_csv('EjemploEstudiantes.csv', delimiter = ';', decimal = ",", index_col = 0)
print(datos)
print(datos.shape)

#Ejecuta k-medias con 3 clusters
kmedias = KMeans(n_clusters = 3)
kmedias.fit(datos)
print(kmedias.predict(datos))
centros = np.array(kmedias.cluster_centers_)
print(centros)
```

Posteriormente, se declara en el código el uso de 3 clasificaciones, es decir, 3 clústeres. Por medio de la función de predicción los datos se transformarán y también se podrá asignar el centro de cada una con otra función similar.

2.- Luego se graficarán estos resultados en dos formas de visualización distintas.

```
#Interpretación con 3 clústeres - Graficación de barras
plt.figure(1, figsize = (12, 8))
bar_plot(centros, datos.columns)
plt.show()

#Interpretación con 3 clústeres - Graficación radar plot
plt.figure(1, figsize = (10, 10))
radar_plot(centros, datos.columns)
plt.show()
```

3.- Para ello, se debieron programar las visualizaciones correspondientes. A continuación está la programación para la gráfica de barras.

```
#Función para graficar los gráficos de barras para la interpretación de clústeres
def bar_plot(centros, labels, cluster = None, var = None):
    from math import ceil, floor
    from seaborn import color_palette
    colores = color_palette()
    minimo = floor(centros.min()) if floor(centros.min()) < 0 else 0
    def inside_plot(valores, labels, titulo):
        plt.barh(range(len(valores)), valores, 1/1.5, color = colores)
        plt.xlim(minimo,ceil(centros.max()))
        plt.title(titulo)
    if var is not None:
        centros = np.array([n[[x in var for x in labels]] for n in centros])
        colores = [colores[x % len(colores)] for x, i in enumerate(labels) if i in var]
        labels = labels[[x in var for x in labels]]
    if cluster is None:
        for i in range(centros.shape[0]):
            plt.subplot(1, centros.shape[0], i + 1)
            inside_plot(centros[i].tolist(), labels, ('Cluster' + str(i)))
            plt.yticks(range(len(labels)), labels) if i == 0 else plt.yticks([])
    else:
        pos = 1
        for i in cluster:
            plt.subplot(1, len(cluster), pos)
            inside_plot(centros[i].tolist(), labels, ('Cluster' + str(i)))
            plt.yticks(range(len(labels)), labels) if i == 1 else plt.yticks([])
            pos += 1
```

Así también se desarrolló otra gráfica de radar que se muestra su código a continuación:

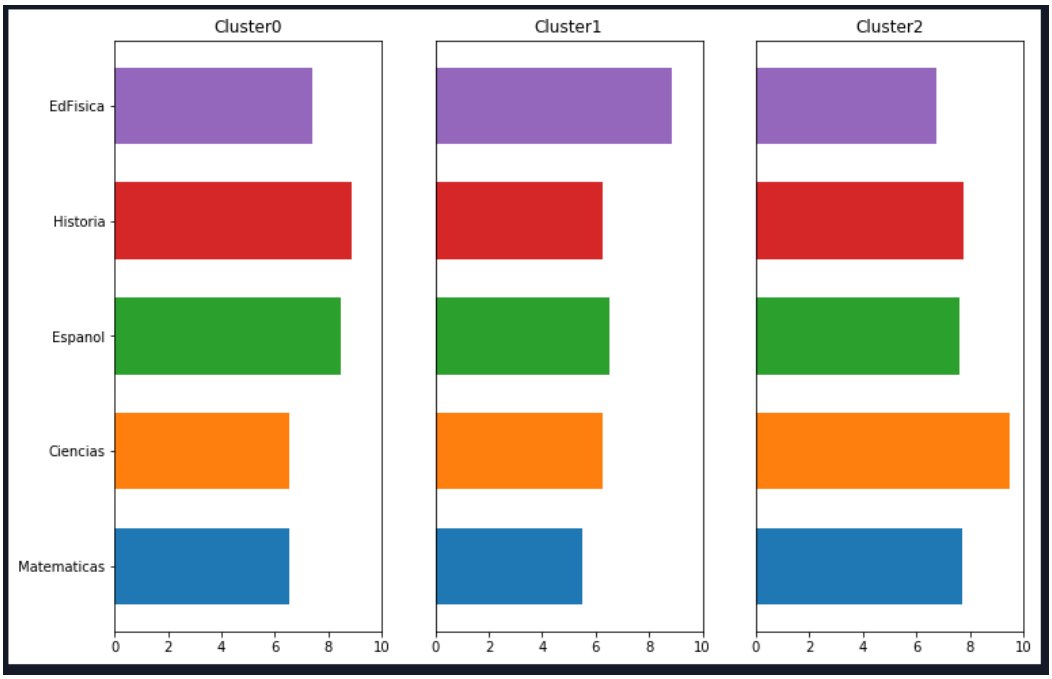
```
#Función para graficar los gráficos tipo radar para la interpretación de clústeres
def radar_plot(centros, labels):
    from math import pi
    centros = np.array([((n - min(n)) / (max(n) - min(n)) * 100) if
                        max(n) != min(n) else (n/n * 50) for n in centros.T])
    angulos = [n / float(len(labels)) * 2 * pi for n in range(len(labels))]
    angulos += angulos[:1]
    ax = plt.subplot(111, polar = True)
    ax.set_theta_offset(pi / 2)
    ax.set_theta_direction(-1)

    plt.xticks(angulos[:-1], labels)
    ax.set_rlabel_position(0)
    plt.yticks([10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
               ["10%", "20%", "30%", "40%", "50%", "60%", "70%", "80%", "90%", "100%"],
               color = "grey", size = 8)
    plt.ylim(-10, 100)
    for i in range(centros.shape[1]):
        valores = centros[:,i].tolist()
        valores += valores[:1]
        ax.plot(angulos, valores, linewidth = 1, linestyle = 'solid',
                label = 'Cluster' + str(i))
        ax.fill(angulos, valores, alpha = 0.3)
        plt.legend(loc = 'upper right', bbox_to_anchor = (0.1, 0.1))
```

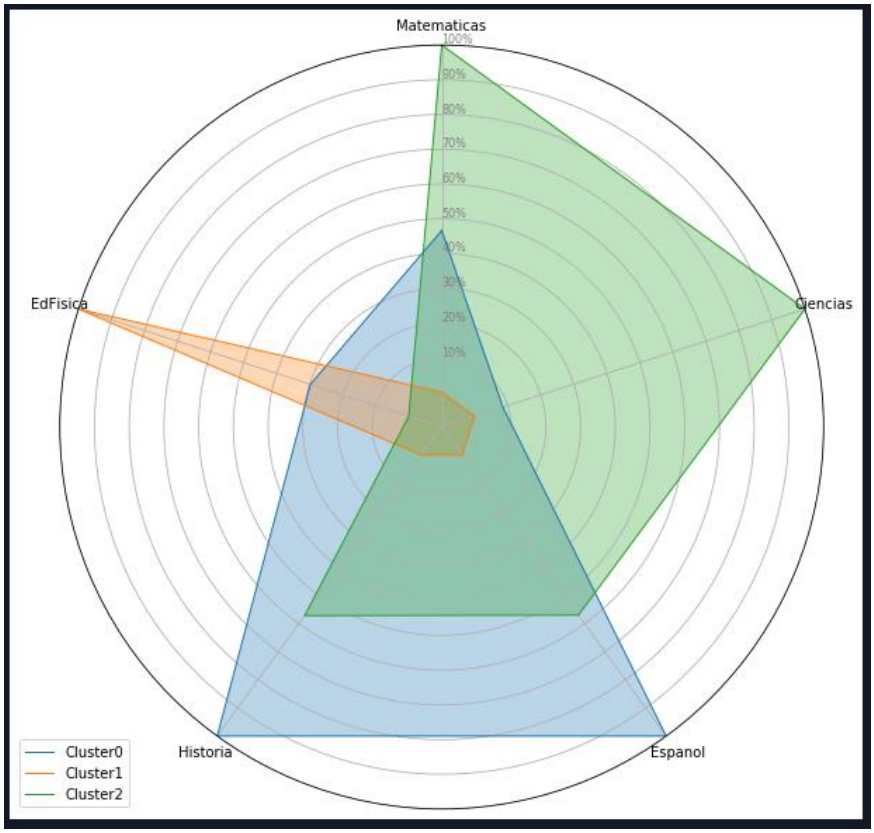
Obtención y Procesamiento de Datos:

La impresión de los primeros valores es la siguiente:

```
        Matematicas  Ciencias  Espanol  Historia  EdFisica
Lucia           7.0       6.5      9.2       8.6       8.0
Pedro           7.5       9.4      7.3       7.0       7.0
Ines            7.6       9.2      8.0       8.0       7.5
Luis            5.0       6.5      6.5       7.0       9.0
Andres          6.0       6.0      7.8       8.9       7.3
Ana             7.8       9.6      7.7       8.0       6.5
Carlos          6.3       6.4      8.2       9.0       7.2
Jose            7.9       9.7      7.5       8.0       6.0
Sonia           6.0       6.0      6.5       5.5       8.7
Maria           6.8       7.2      8.7       9.0       7.0
(10, 5)
[0 2 2 1 0 2 0 2 1 0]
[[6.525 6.525 8.475 8.875 7.375]
 [5.5   6.25  6.5   6.25  8.85 ]
 [7.7   9.475 7.625 7.75  6.75 ]]
```

Se observa que los valores están leyéndose correctamente, incluso se perciben los valores de los centros y clasificadores en la imagen anterior, pero para percibirla de mejor manera, se desarrollaron los gráficos de barras y radar.





Conclusiones:

Joel Alejandro Espinoza Sánchez: El desarrollo de este programa nos permite observar la implementación de un aprendizaje no supervisado por medio de métodos que a la computadora le ayudan, tales como la separación correcta de la información, por ello se trata de minimizar la distancia que existe entre los elementos de una clasificación (distancia intraclúster) pero maximizar las distancias de las clasificaciones como conjunto en sí (distancia interclúster), de este modo, se puede procesar la información sin necesidad de supervisarse y fue un buen método de aplicación para librerías y procedimientos que no habíamos probado.

Dariana Gómez Garza: En este trabajo pudimos aplicar y visualizar este algoritmo de clasificación no supervisada mucho mejor completando nuestros apuntes de las clases y de la práctica que habíamos realizado en R, ahora en otro lenguaje diferente. Nos ayuda a realizar y conseguir predicciones un poco más precisas a partir de los datos previos que tenemos

Fernando Francisco González Arenas: Con esta práctica implementamos el algoritmo de K medias en Python, el cual previamente se nos fue facilitado en el lenguaje R. con este algoritmo podemos agrupar datos en diferentes grupos tomando en cuenta la cercanía de los datos con los clústeres elegidos al azar.

Así podemos implementar el aprendizaje no supervisado, el cual nos permite que nuestra inteligencia artificial aprenda solo a partir de los datos de entrada que se le proporcionan. esto es muy útil ya que no es necesario decirle a la IA lo que queremos que sea la salida, sino que será capas por si sola identificar las características que tengan en común los datos y a partir de eso generar información y por consiguiente conocimiento.

Referencias:

Martínez, C. (2018). *Métodos de Clustering.* Marzo 21, 2021, de RPubs Sitio web: https://rpubs.com/Cristina_Gil/Clustering#:~:text=El%20m%C3%A9todo%20de%20K%2Dmeans,a%20m%C3%A1s%20de%20un%20cl%C3%BAster.

Anexos:

Anexo 1: Código del programa en lenguaje Python:

```
#%% Presentación
'''
   Universidad Autónoma de Aguascalientes

         Centro de Ciencias Básicas
 Departamento de Ciencias de la Computación
          Aprendizaje Inteligente
                   6° "A"

                 Práctica 3

   Profesor: Francisco Javier Luna Rosas
  Alumnos:
      Espinoza Sánchez Joel Alejandro
      Gómez Garza Dariana
      González Arenas Fernando Francisco

  Fecha de Entrega: 15 de marzo del 2021

Descripción: Red neuroonal con Backtracking
'''


#Configuraciones de Python
import pandas as pd
pd.options.display.max_rows = 10

#Paquetes usados en la presentación
import os
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np

#Función para graficar los gráficos de barras para la interpretación de
clústeres
def bar_plot(centros, labels, cluster = None, var = None):
    from math import ceil, floor
    from seaborn import color_palette
    colores = color_palette()
    minimo = floor(centros.min()) if floor(centros.min()) < 0 else 0
    def inside_plot(valores, labels, titulo):
        plt.barh(range(len(valores)), valores, 1/1.5, color = colores)
        plt.xlim(minimo,ceil(centros.max()))
        plt.title(titulo)
    if var is not None:
        centros = np.array([n[[x in var for x in labels]] for n in centros])
```

```
        colores = [colores[x % len(colores)] for x, i in enumerate(labels) if i
in var]
        labels = labels[[x in var for x in labels]]
    if cluster is None:
        for i in range(centros.shape[0]):
            plt.subplot(1, centros.shape[0], i + 1)
            inside_plot(centros[i].tolist(), labels, ('Cluster' + str(i)))
            plt.yticks(range(len(labels)), labels) if i == 0 else plt.yticks([])
    else:
        pos = 1
        for i in cluster:
            plt.subplot(1, len(cluster), pos)
            inside_plot(centros[i].tolist(), labels, ('Cluster' + str(i)))
            plt.yticks(range(len(labels)), labels) if i == 1 else plt.yticks([])
            pos += 1

#Función para graficar los gráficos tipo radar para la interpretación de
clústeres
def radar_plot(centros, labels):
    from math import pi
    centros = np.array([((n - min(n)) / (max(n) - min(n)) * 100) if
                        max(n) != min(n) else (n/n * 50) for n in centros.T])
    angulos = [n / float(len(labels)) * 2 * pi for n in range(len(labels))]
    angulos += angulos[:1]
    ax = plt.subplot(111, polar = True)
    ax.set_theta_offset(pi / 2)
    ax.set_theta_direction(-1)

    plt.xticks(angulos[:-1], labels)
    ax.set_rlabel_position(0)
    plt.yticks([10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
               ["10%", "20%", "30%", "40%", "50%", "60%", "70%", "80%", "90%",
"100%"],
               color = "grey", size = 8)
    plt.ylim(-10, 100)
    for i in range(centros.shape[1]):
        valores = centros[:,i].tolist()
        valores += valores[:1]
        ax.plot(angulos, valores, linewidth = 1, linestyle = 'solid',
                label = 'Cluster' + str(i))
        ax.fill(angulos, valores, alpha = 0.3)
        plt.legend(loc = 'upper right', bbox_to_anchor = (0.1, 0.1))

#Estudiantes
import pandas as pd
os.chdir("C:/Users/Dell/Desktop/KMeans") #Ruta a cambiar donde se ubica tanto el
.py como el .csv

datos = pd.read_csv('EjemploEstudiantes.csv', delimiter = ';', decimal = ",",
index_col = 0)
print(datos)
```

```
print(datos.shape)

#Ejecuta k-medias con 3 clusters
kmedias = KMeans(n_clusters = 3)
kmedias.fit(datos)
print(kmedias.predict(datos))
centros = np.array(kmedias.cluster_centers_)
print(centros)

#%% Gráfica Bar plot para la interpretación

#Interpretación con 3 clústeres - Graficación de barras
plt.figure(1, figsize = (12, 8))
bar_plot(centros, datos.columns)
plt.show()

#Interpretación con 3 clústeres - Graficación radar plot
plt.figure(1, figsize = (10, 10))
radar_plot(centros, datos.columns)
plt.show()
```