Universidad Nacional Autónoma de México
Facultad de Ingeniería
División de Ingeniería Eléctrica
Ingeniería en Computación

Práctica 3.1
“Cluster”

Integrantes:
- Castelan Ramos Carlos

Materia: Minería de Datos
Grupo: 03
Semestre: 2024-1

Fecha de entrega: 28 de septiembre 2023

Práctica 3.1 “Cluster”

Instrucciones:

Realizar la practica 3.1 Cluster con las pruebas de 2 algoritmos de cluster uno debe ser K-means y el otro el que deseen, variar los parámetros y dar sus observaciones de como impactan y también comentar la diferencia entre los 2 algoritmos, tomar los datos que se anexa de Transacciones de Ventas para la prueba.

Introducción

En el mundo de la minería de datos y el análisis de datos, el clustering es una técnica fundamental que se utiliza para agrupar datos similares y descubrir patrones ocultos en conjuntos de datos. Dos de los algoritmos de clustering más populares y ampliamente utilizados son K-Means y DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

Esta práctica tiene como objetivo explorar y aplicar estos dos algoritmos de clustering, K-Means y DBSCAN, para realizar análisis de agrupamiento en un conjunto de datos específico. A lo largo de este proyecto, examinaremos tanto la teoría detrás de estos algoritmos como su implementación práctica en Python. Además, evaluaremos su desempeño en un contexto del mundo real y exploraremos cómo pueden ser útiles en la toma de decisiones y la extracción de conocimiento a partir de datos.

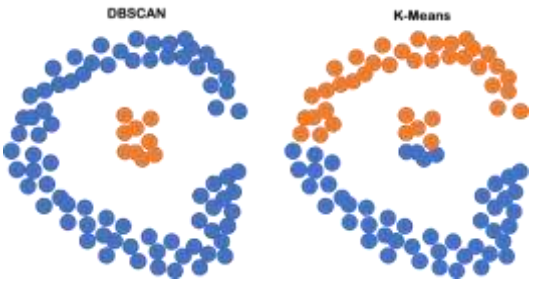
Objetivos del Proyecto

- Comprender los conceptos fundamentales de clustering y las diferencias entre K-Means y DBSCAN.
- Implementar los algoritmos de K-Means y DBSCAN en Python utilizando bibliotecas como scikit-learn.
- Realizar una exploración de datos previa para comprender mejor el conjunto de datos de entrada.
- Aplicar K-Means y DBSCAN al conjunto de datos para descubrir clústeres o grupos de datos similares.
- Evaluar y comparar el rendimiento de ambos algoritmos utilizando métricas de evaluación de clustering.
- Visualizar los resultados de clustering de manera efectiva para comprender la estructura de los datos.
- Interpretar y discutir las implicaciones y aplicaciones de los resultados de clustering en un contexto del mundo real.

Revisión teórica:

K-Means:

K-Means es un algoritmo de clustering ampliamente utilizado en aprendizaje no supervisado que busca agrupar datos en clústeres basados en similitudes. Funciona asignando puntos de datos a clústeres de tal manera que minimiza la distancia euclidiana entre los puntos y el centroide (punto promedio) de su clúster asignado. El algoritmo comienza eligiendo aleatoriamente k centroides y luego itera para mejorar la asignación de puntos a clústeres y los centroides hasta que converja. K-Means es eficiente y fácil de implementar, pero requiere que se especifique el número de clústeres de antemano y es sensible a la inicialización de centroides.



DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

DBSCAN es otro algoritmo de clustering que se basa en la densidad de los datos en lugar de la distancia euclidiana. Este algoritmo identifica clústeres encontrando regiones densas de puntos en el espacio de datos, separadas por regiones menos densas. No es necesario especificar el número de clústeres de antemano, y DBSCAN puede identificar clústeres de diferentes formas y tamaños, lo que lo hace robusto frente a datos con ruido y clústeres de densidad variable. DBSCAN también puede identificar puntos atípicos o de ruido. Sin embargo, la elección de los parámetros, como el radio de la vecindad (eps) y el número mínimo de puntos (min_samples), puede afectar los resultados.

Realización de la práctica:

K-Means:

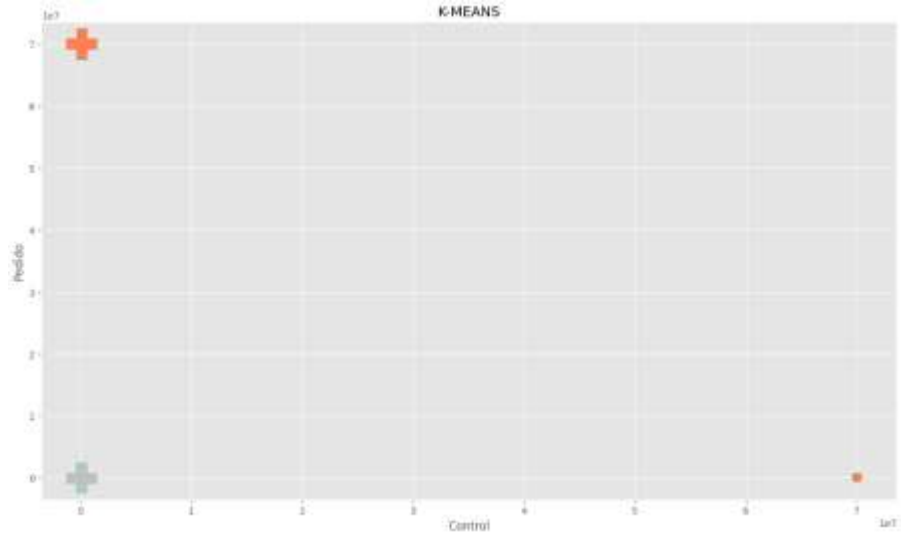
URL:
https://colab.research.google.com/drive/1k7D9vLmqJengLQmdp_BpS9MEDNf2C52j#scrollTo=Tzt3T7S8vJGU&uniqifier=1

Código:

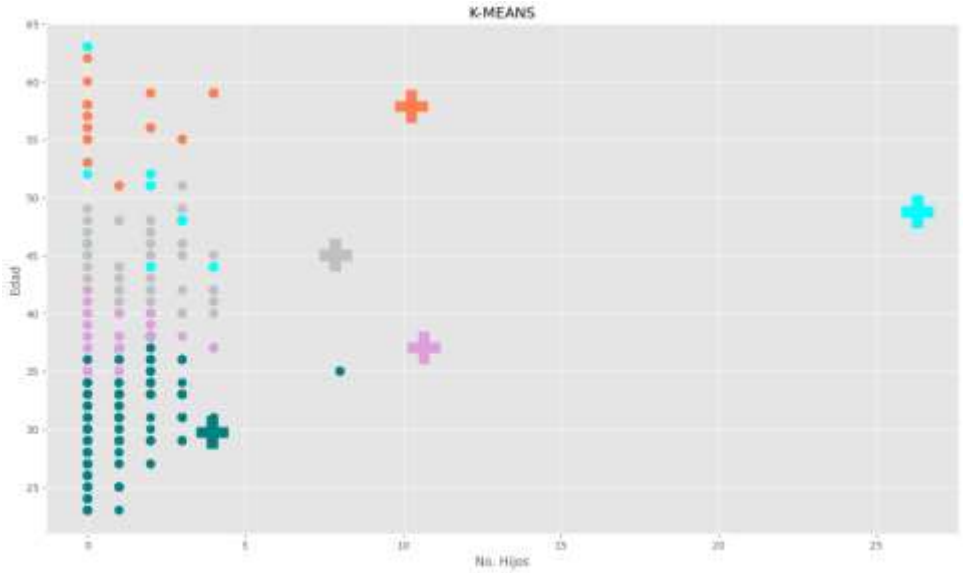
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min

%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

transaccion = pd.read_csv("/content/transacciones.csv", encoding='latin1')
transaccion

# Primer Análisis
X = np.array(transaccion[["Pedido","Control","Producto"]])
y = np.array(transaccion['Quarter'])
X.shape

kmeans = KMeans(n_clusters=5).fit(X)
centroids = kmeans.cluster_centers_
print(centroids)

# Predicting the clusters
labels = kmeans.predict(X)
# Getting the cluster centers
C = kmeans.cluster_centers_
color=['plum','teal','coral','cyan','silver']
asignar=[]
for row in labels:
    asignar.append(color[row])

# Getting the values and plotting it
ejeX1 = transaccion['Producto'].values
ejeY1 = transaccion['Pedido'].values

plt.title("K-MEANS")
plt.xlabel("Control")
plt.ylabel("Pedido")
plt.scatter(ejeX1, ejeY1, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='P', c=color, s=1000)
plt.show()
```

```
# Segundo Análisis
X1 = np.array(transaccion[["Antiguedad","Edad","No._Hijos"]])
y1 = np.array(transaccion['Quarter'])
X1.shape

kmeans = KMeans(n_clusters=5).fit(X1)
centroids = kmeans.cluster_centers_
print(centroids)

# Predicting the clusters
labels = kmeans.predict(X1)
# Getting the cluster centers
C = kmeans.cluster_centers_
color=['plum','teal','coral','cyan','silver']
asignar=[]
for row in labels:
    asignar.append(color[row])

# Getting the values and plotting it
ejeX = transaccion['No._Hijos'].values
ejeY = transaccion['Edad'].values

plt.title("K-MEANS")
plt.xlabel("No. Hijos")
plt.ylabel("Edad")
plt.scatter(ejeX, ejeY, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='P', c=color, s=1000)
plt.show()

# Getting the values and plotting it
ejeX = transaccion['Antiguedad'].values
ejeY = transaccion['Edad'].values

plt.title("K-MEANS")
plt.xlabel("No. Hijos")
plt.ylabel("Edad")
plt.scatter(ejeX, ejeY, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='P', c=color, s=1000)
plt.show()
```

Clustering Producto-Pedido:



Clustering Edad-No. Hijos



Clustering Edad-Antigüedad

Podemos observar que para K-medias, la dispersión es demasiado grande dentro de los valores de las columnas, donde las dos únicas decentes son las de Edad-No. Hijos y Edad-Antigüedad.

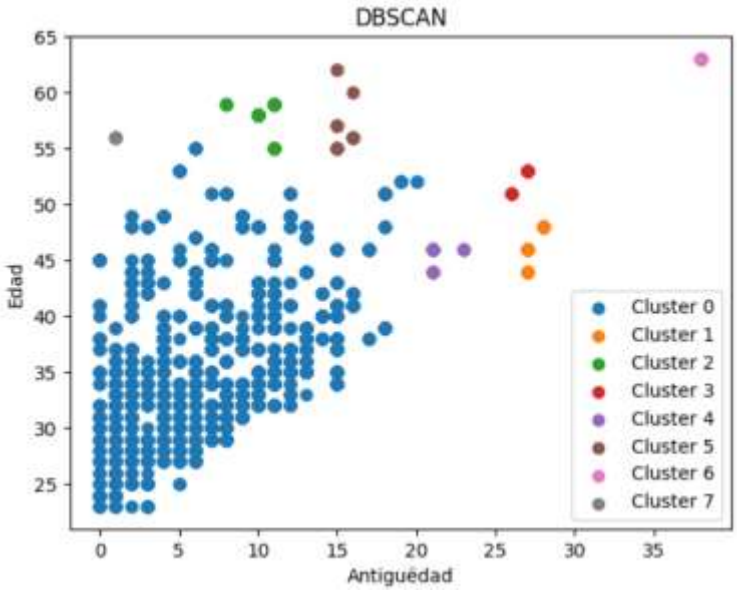
DBSCAN:

URL: https://colab.research.google.com/drive/1m3aM7-eAta48NeejGqpuQn_RPUhNMv2h#scrollTo=Wm-zjAqudaSO

Código:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

data = pd.read_csv("transacciones.csv", encoding='latin1')

# Supongamos que tienes dos columnas 'X' y 'Y' que deseas utilizar para clustering
X = data[['Antiguedad', 'Edad']]

# Escalar los datos para que tengan media 0 y varianza 1 (importante para DBSCAN)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Crear y ajustar el modelo DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(X_scaled)

# Agregar las etiquetas de clúster al DataFrame original
data['Cluster'] = dbscan.labels_

# Visualizar los resultados en un gráfico de dispersión
plt.figure(figsize=(8, 6))

# Los puntos que no pertenecen a ningún clúster se etiquetan como ruido (-1)
plt.scatter(data[data['Cluster'] == -1]['Antiguedad'], data[data['Cluster'] == -
1]['Edad'], color='gray', label='Ruido')

# Visualizar los clústeres
unique_clusters = set(dbscan.labels_)
for cluster in unique_clusters:
    if cluster != -1:  # Excluye el ruido
        cluster_data = data[data['Cluster'] == cluster]
        plt.scatter(cluster_data['Antiguedad'], cluster_data['Edad'],
label=f'Cluster {cluster}')
        plt.xlabel('X')
plt.ylabel('Y')
plt.title('Resultados del clustering DBSCAN')
plt.legend()
plt.show()
```

Clustering Edad-Antigüedad



Debido a que con el anterior algoritmo de Clustering nos dimos cuenta de que los valores que arrojan una agrupación no tan dispersa es la edad con antigüedad.

Conclusiones.

Tanto K-Means como DBSCAN son algoritmos de clustering ampliamente utilizados en el campo de la minería de datos, pero difieren en enfoque y funcionamiento. Una similitud importante es que ambos buscan agrupar datos en clústeres, pero las diferencias radican en cómo lo hacen. K-Means utiliza la distancia euclidiana para asignar puntos de datos a clústeres, mientras que DBSCAN se basa en la densidad de los datos y busca regiones densas en el espacio de datos. Una ventaja de DBSCAN es que no requiere que se especifique el número de clústeres de antemano, lo que lo hace adecuado para datos con clústeres de formas y tamaños variados. Además, DBSCAN puede identificar puntos de datos como ruido o atípicos. Por otro lado, K-Means es más simple de implementar y generalmente es más eficiente computacionalmente, pero puede ser sensible a la inicialización de centroides y requiere conocer el número de clústeres previamente. En resumen, la elección entre K-Means y DBSCAN dependerá de la naturaleza de los datos y los objetivos de la tarea de clustering, ya que cada uno tiene sus fortalezas y limitaciones.

Notas.

- Github: https://github.com/CarlosCR07/Miner-a-de-Datos
- Drive: https://drive.google.com/drive/folders/1YgQPkkuGlWwOwjlvrdYEOHd_GrynFDUB?usp=drive_link

Referencias:

- Na, & Na. (2020). K-Means con Python paso a paso | Aprende Machine Learning. *Aprende Machine Learning*. https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/
- Omran, M. G., Engelbrecht, A. P., & Salman, A. (2007). An overview of clustering methods. *Intelligent Data Analysis*, *11*(6), 583-605.
- Milligan, G. W., & Cooper, M. C. (1987). Methodology review: Clustering methods. *Applied psychological measurement*, *11*(4), 329-354.
- Rocio Chavez Ciencia de Datos. (2020, 6 abril). *Clustering Método K-Means en Python (ENGLISH SUBTITLES)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=s6PSSzeUMFk
- Matematica II B. (2020, 5 julio). *Clase práctica | ejemplo donde K-Medias no fucniona y DBSCAN sí - Cómo elegir el Epsilon en DBSCAN?* [Vídeo]. YouTube. https://www.youtube.com/watch?v=l_RlzJ9UY3Q