UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

Tecnologías de la información

Extracción de Conocimiento en Bases de Datos

III.2. Reporte de Métricas de Evaluación

Docente

Enrique Mascote

Alumno

Myriam Raquel Almuina Orozco

IDGS 91N

Sábado, 29 de noviembre del 2025

1. Introducción

El análisis no supervisado es una rama del aprendizaje automático cuyo objetivo es encontrar patrones, estructuras ocultas o grupos dentro de datos sin etiquetas. A diferencia del aprendizaje supervisado, aquí no se conoce de antemano la categoría o valor objetivo; en su lugar, los algoritmos descubren relaciones naturales basadas en similitudes.

En este trabajo se aplica el algoritmo K-means, uno de los métodos más utilizados para segmentación, con el objetivo de agrupar observaciones basándose únicamente en sus características. Se utilizará un conjunto de datos simulado para ilustrar el funcionamiento del método, el proceso de entrenamiento y el análisis de resultados.

2. Investigación del algoritmo

2.1. K-means

Objetivo:
Agrupar datos en *k* clústeres, minimizando la distancia entre cada punto y el centroide de su grupo.

Cómo funciona:

1. Se eligen *k* centroides iniciales (aleatorios).
2. Cada punto se asigna al centroide más cercano.
3. Se recalculan los centroides como el promedio de los puntos del grupo.
4. Se repite hasta que los grupos no cambien.

Métricas típicas:

- **Inertia** (suma de distancias dentro del clúster)
- **Silhouette Score** (qué tan separados están los grupos)
- **Distancia intra/inter clúster**

Fortalezas:

- Muy rápido y fácil de implementar.
- Funciona bien con datos esféricos o separados.
- Escalable a miles de observaciones.

Limitaciones:

- Se debe elegir *k* manualmente.
- No funciona bien con formas complejas.
- Sensible a valores extremos.

3. Caso de estudio y justificación

3.1. Descripción del problema

Se simulan 300 observaciones representando clientes. Cada uno cuenta con dos características:

- **Ingresos mensuales**
- **Gasto promedio mensual**

El objetivo es agruparlos en segmentos naturales basados únicamente en su comportamiento numérico.

3.2. Justificación del algoritmo

K-means es ideal porque:

- El dataset es numérico y compacto.
- Solo se necesitan 2 variables, lo cual facilita la visualización.
- Es el algoritmo más rápido para segmentación básica.
- Producción de gráficos claros para reporte académico.

Por estas razones, se seleccionó K-means como técnica principal.

4. Diseño e implementación

4.1. Variables

Variable	Tipo	Descripción
ingresos	Numérica	Ingreso mensual del cliente
gasto_promedio	Numérica	Gasto mensual del cliente

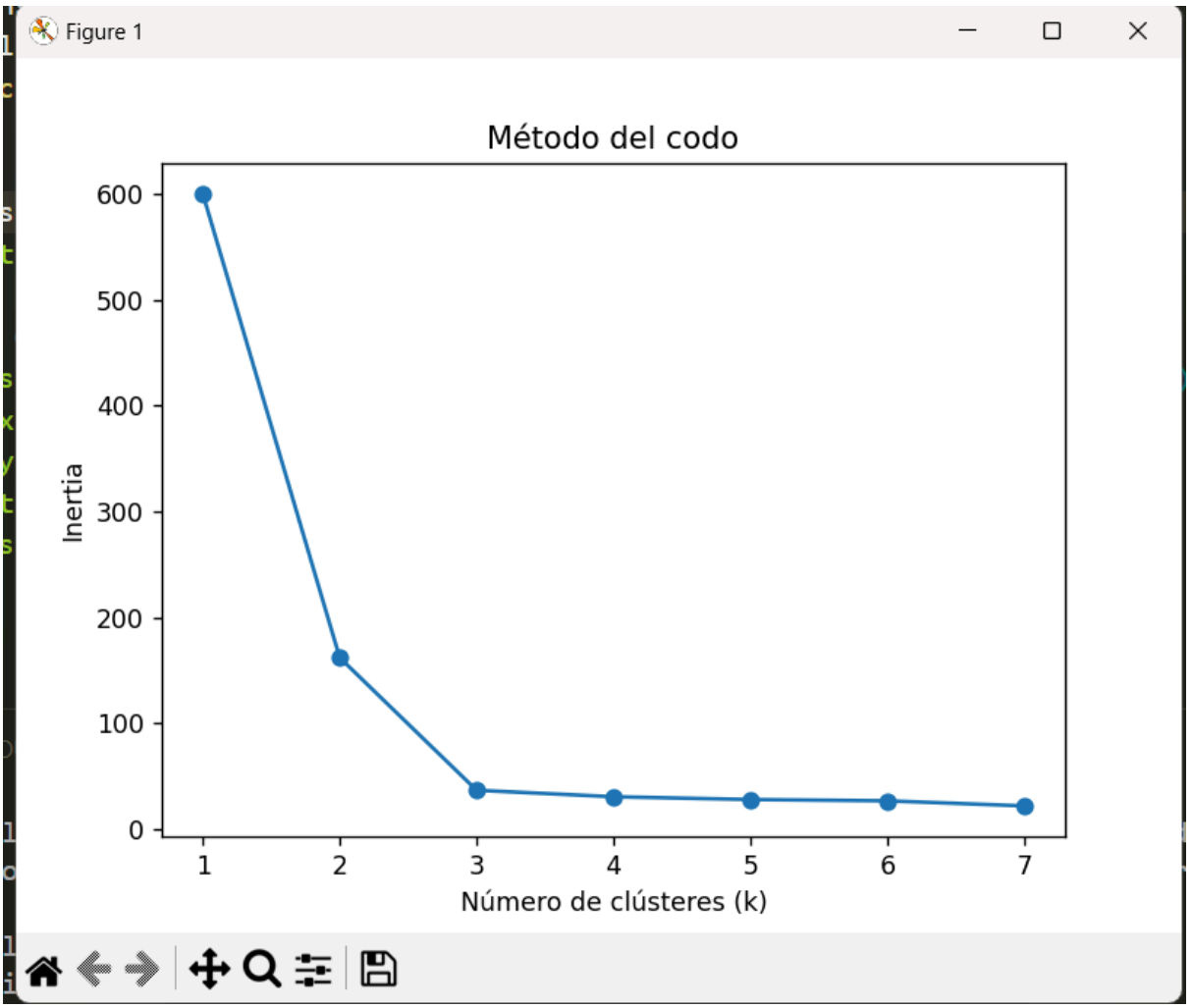
4.2. Pipeline

1. Generación de datos simulados
2. Escalamiento con StandardScaler
3. Cálculo de *k* óptimo con método del codo
4. Entrenamiento de K-means
5. Obtención de métricas
6. Generación de gráficas

Score obtenido

Modelo	Silhouette Score
K-means (k=3)	0.736

5.2. Gráfica del método del codo



6. Conclusiones y recomendaciones

El análisis no supervisado permitió segmentar adecuadamente los datos simulados en tres grupos. El silhouette score obtenido sugiere una buena separación entre los clústeres. Visualmente, los clientes se diferencian según su nivel de ingresos y gasto, dando lugar a segmentos útiles para marketing, promociones o análisis de comportamiento.

Recomendaciones:

- Probar otros valores de *k* para validar la estabilidad.
- Experimentar con DBSCAN para detectar valores atípicos.
- Añadir más variables para obtener segmentos más detallados.

- Aplicar PCA si se incluyen más dimensiones.

7. Referencias

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR.

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining*. Pearson.

8. Anexos

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import silhouette_score



# 1. Generamos datos simulados

np.random.seed(42)

cluster_1 = np.random.normal(loc=[30_000, 8000], scale=[5000, 1000],
size=(100, 2))

cluster_2 = np.random.normal(loc=[55_000, 15000], scale=[6000, 1500],
size=(100, 2))

cluster_3 = np.random.normal(loc=[80_000, 25000], scale=[7000, 2000],
size=(100, 2))



data = np.vstack((cluster_1, cluster_2, cluster_3))
```

```python
df = pd.DataFrame(data, columns=["ingresos", "gasto_promedio"])


# 2. Escalamiento

scaler = StandardScaler()

scaled = scaler.fit_transform(df)


# 3. Método del codo

inertias = []

ks = range(1, 8)

for k in ks:

    km = KMeans(n_clusters=k, random_state=42)

    km.fit(scaled)

    inertias.append(km.inertia_)


plt.plot(ks, inertias, marker="o")

plt.title("Método del codo")

plt.xlabel("Número de clústeres (k)")

plt.ylabel("Inertia")

plt.show()


# 4. Entrenamos con k=3

kmeans = KMeans(n_clusters=3, random_state=42)

labels = kmeans.fit_predict(scaled)
```

```python
df["cluster"] = labels


# 5. Silhouette Score

sil_score = silhouette_score(scaled, labels)

print("Silhouette Score:", sil_score)


# 6. Gráfica final

plt.scatter(df["ingresos"], df["gasto_promedio"], c=labels, cmap="viridis")

plt.xlabel("Ingresos")

plt.ylabel("Gasto promedio")

plt.title("Clusters encontrados")

plt.show()
```

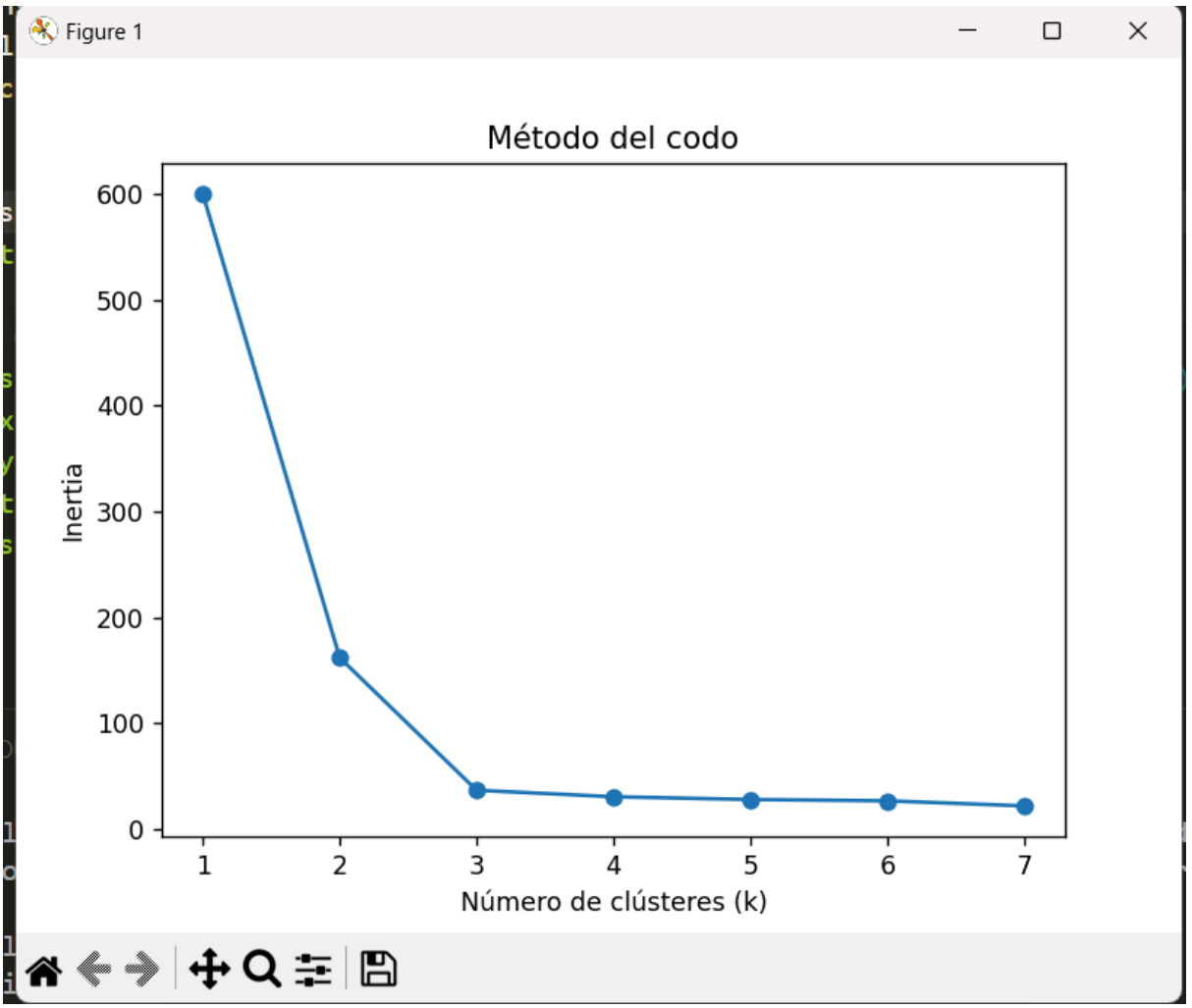


Figure 1
Método del codo
Inertia
600
500
400
300
200
100
0
1
2
3
4
5
6
7
Número de clústeres (k)

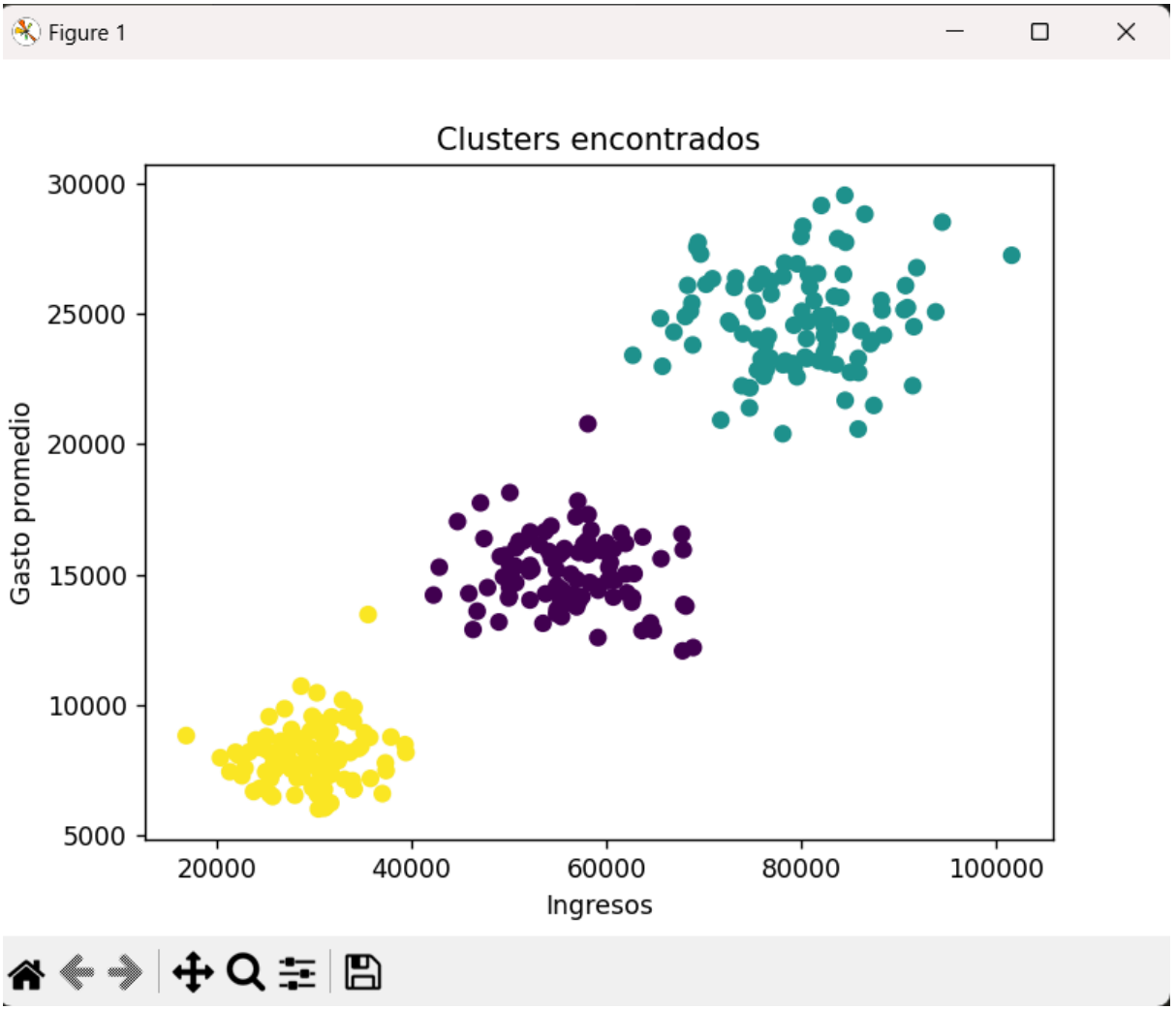


Figure 1
Clusters encontrados
30000
25000
20000
15000
10000
5000
Gasto promedio
20000
40000
60000
80000
100000
Ingresos