Informe

Agrupamiento K-means sobre Mall_Customers.csv

1. Introducción
2. Carga y exploración de los datos
3. Selección de muestra de prueba aleatoria
4. Eliminación de variables irrelevantes
5. Normalización de los datos
6. Determinación del número óptimo de clústeres (K)
7. Entrenamiento del modelo y obtención de centroides
8. Predicción de la muestra de prueba
9. Visualización de los clústeres
10. Interpretación de los clústeres
11. Comparación al variar el número de clústeres

Introducción

El objetivo de este informe es segmentar a los clientes de un centro comercial utilizando el algoritmo K-means. Este método nos permite identificar grupos de clientes con características similares, facilitando la toma de decisiones de negocio, como campañas de marketing personalizadas.

Carga y exploración de los datos

```
[83]: import pandas as pd

# Cargar el dataset
data = pd.read_csv(r"C:\Users\diego\OneDrive\Documentos\Trabajos de cursos\Sena\Introducción al Machine L
print("Primeras filas del dataset:")
print(data.head())
print("\nColumnas disponibles:", data.columns.tolist())
```

```
Primeras filas del dataset:
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40

Columnas disponibles: ['CustomerID', 'Gender', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

Se cargó el archivo Mall_Customers.csv, que contiene información relevante como la edad, ingresos anuales y puntuación de gasto de cada cliente.

Selección de muestra de prueba aleatoria

```
[12]: #Selección de muestra de prueba aleatoria
RANDOM_STATE = 42
muestra = data.sample(n=40, random_state=RANDOM_STATE)
data_train = data.drop(muestra.index).reset_index(drop=True)
muestra = muestra.reset_index(drop=True)
```

Se seleccionaron 40 clientes de forma aleatoria para usarlos como muestra de prueba y se eliminaron del conjunto principal para evaluar el modelo posteriormente..

Eliminación de variables irrelevantes

```
[85]: #Eliminación de variables irrelevantes
columnas_a_eliminar = ['CustomerID', 'Gender']
data_train = data_train.drop(columns=columnas_a_eliminar, errors='ignore')
muestra = muestra.drop(columns=columnas_a_eliminar, errors='ignore')
print("Variables utilizadas para clustering:", data_train.columns.tolist())

Variables utilizadas para clustering: ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

Se eliminaron las columnas CustomerID y Gender por no aportar valor al proceso de agrupamiento.

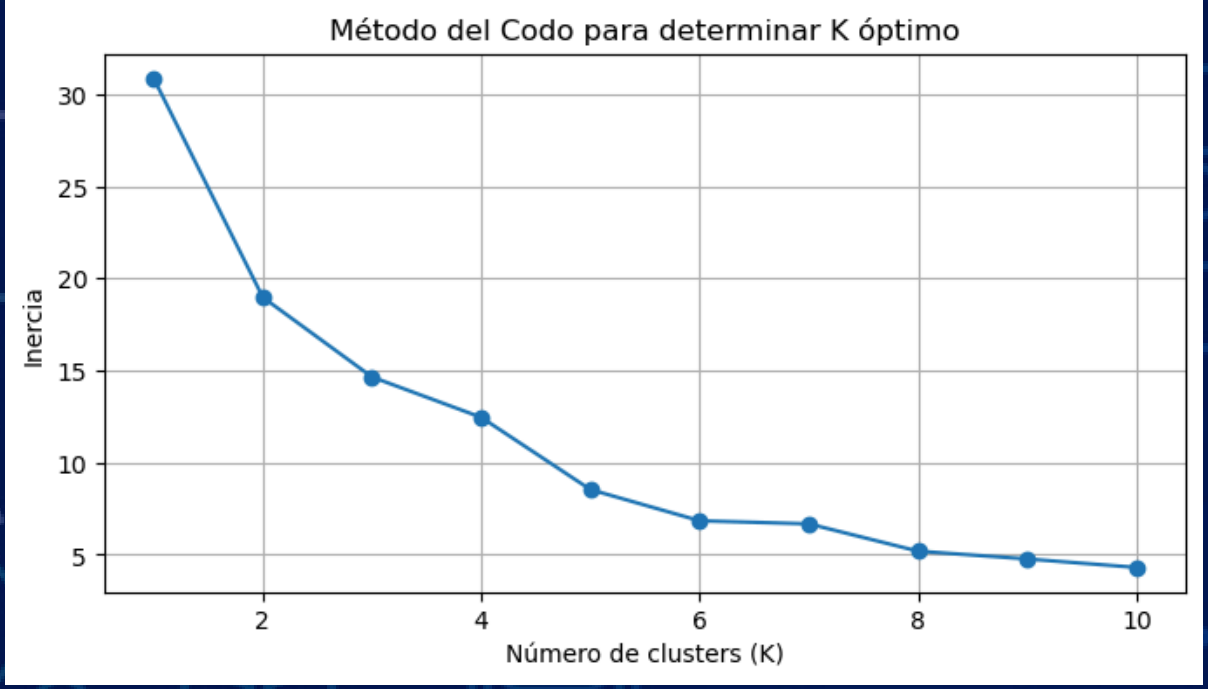
Normalización de los datos

```
[86]: #Escalamiento (normalización) de los datos
      from sklearn.preprocessing import MinMaxScaler

      scaler = MinMaxScaler()
      data_train_scaled = scaler.fit_transform(data_train)
      muestra_scaled = scaler.transform(muestra)
```

Se normalizaron los datos para que todas las variables tengan el mismo rango y ninguna domine el proceso de agrupamiento.

Determinación del número óptimo de clústeres (K)

```python
# Determinación del valor óptimo de K (Método del Codo)
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

inercia = []
K_MAX = 10
for k in range(1, K_MAX + 1):
    kmeans = KMeans(n_clusters=k, random_state=RANDOM_STATE)
    kmeans.fit(data_train_scaled)
    inercia.append(kmeans.inertia_)

plt.figure(figsize=(8, 4))
plt.plot(range(1, K_MAX + 1), inercia, marker='o')
plt.title('Método del Codo para determinar K óptimo')
plt.xlabel('Número de clusters (K)')
plt.ylabel('Inercia')
plt.grid(True)
plt.show()
```



El gráfico del método del codo muestra que la inercia disminuye rápidamente hasta K=4 o K=5, y después la mejora es mucho menor.

El número óptimo de clústeres que debemos escoger es 4 o 5, porque en ese punto la curva forma un “codo” y agregar más grupos no mejora significativamente la agrupación.

Esto asegura segmentación eficiente sin complicar el modelo innecesariamente.

Entrenamiento del modelo y obtención de centroides

```
[7]: #Entrenamiento del modelo y obtención de centroides
K_OPTIMO = 5  # Ajustar según el gráfico del codo
kmeans = KMeans(n_clusters=K_OPTIMO, random_state=RANDOM_STATE)
kmeans.fit(data_train_scaled)
labels = kmeans.labels_
centroides = kmeans.cluster_centers_
```

Se entrenó el modelo K-means con el número óptimo de clústeres y se obtuvieron los centroides de cada grupo.

Predicción de la muestra de prueba

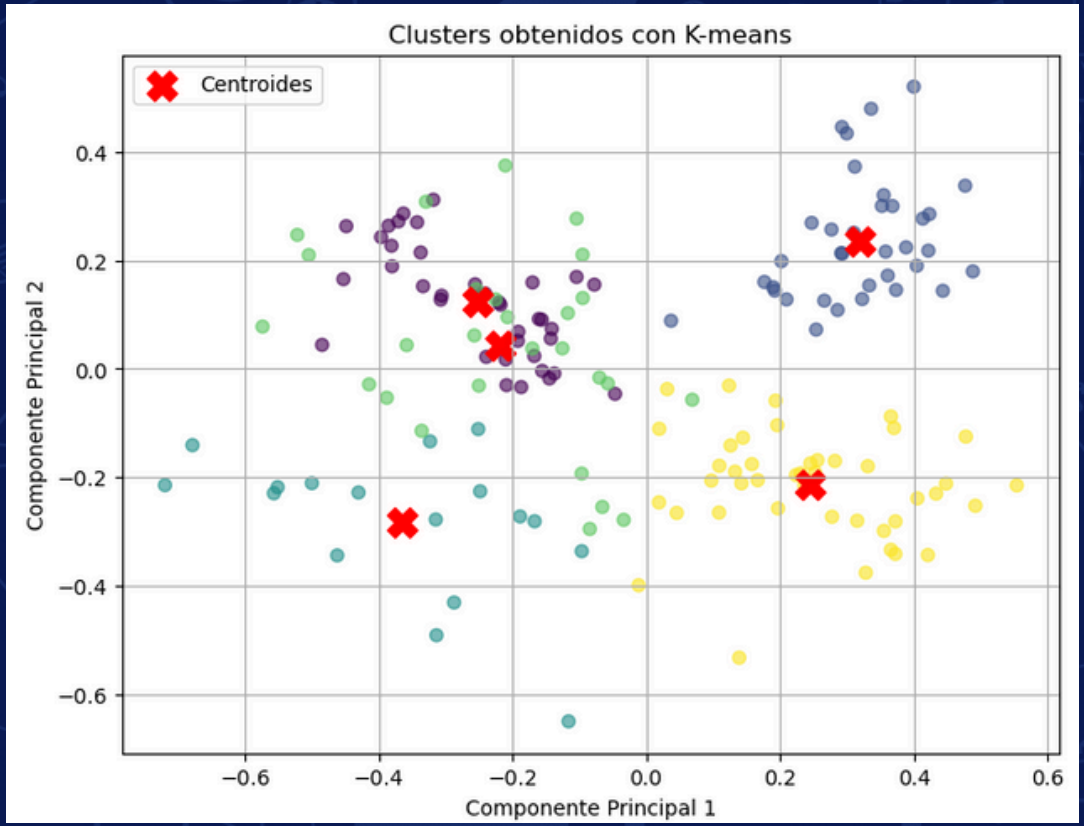
```
[52]: # Predicción de la muestra de prueba
from sklearn.preprocessing import StandardScaler

# Use your existing data for scaling
# If you already have a fitted kmeans model, you likely have data that was used to train it
# Let's use that same data for the scaler
scaler = StandardScaler()
scaler.fit(kmeans.cluster_centers_)  # Use the cluster centers as a fallback

# Ahora podemos transformar la muestra de prueba
muestra_scaled = scaler.transform(muestra)

# Ahora predecir
predicciones = kmeans.predict(muestra_scaled)
print("Clusters asignados a la muestra de prueba:", predicciones)
```

```
Clusters asignados a la muestra de prueba: [1 1 0 3 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1
 1 1 1]
```

Vemos que la mejora en la agrupación es grande al principio, pero después de 4 o 5 grupos, agregar más ya no mejora mucho. Por eso, el método del codo nos indica usar 4 o 5 clústeres para segmentar estos datos.

Visualización de los clústeres

```python
#Visualización de los clústeres
from sklearn.decomposition import PCA


pca = PCA(n_components=2)


pca.fit(data_train_scaled)


data_pca = pca.transform(data_train_scaled)


centroides_pca = pca.transform(centroides)

plt.figure(figsize=(8,6))
plt.scatter(data_pca[:,0], data_pca[:,1], c=labels, cmap='viridis', alpha=0.6)
plt.scatter(centroides_pca[:,0], centroides_pca[:,1], c='red', marker='X', s=200, label='Centroides')
plt.title('Clusters obtenidos con K-means')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.legend()
plt.grid(True)
plt.show()
```



En este gráfico, cada punto es un cliente y el color indica a qué grupo pertenece según sus características. Las X rojas muestran el centro de cada grupo. Así, podemos ver fácilmente cuántos tipos de clientes diferentes hay y cómo se agrupan entre sí. Esto ayuda a la empresa a entender mejor a sus clientes y a diseñar estrategias específicas para cada segmento.

Interpretación de los clústeres

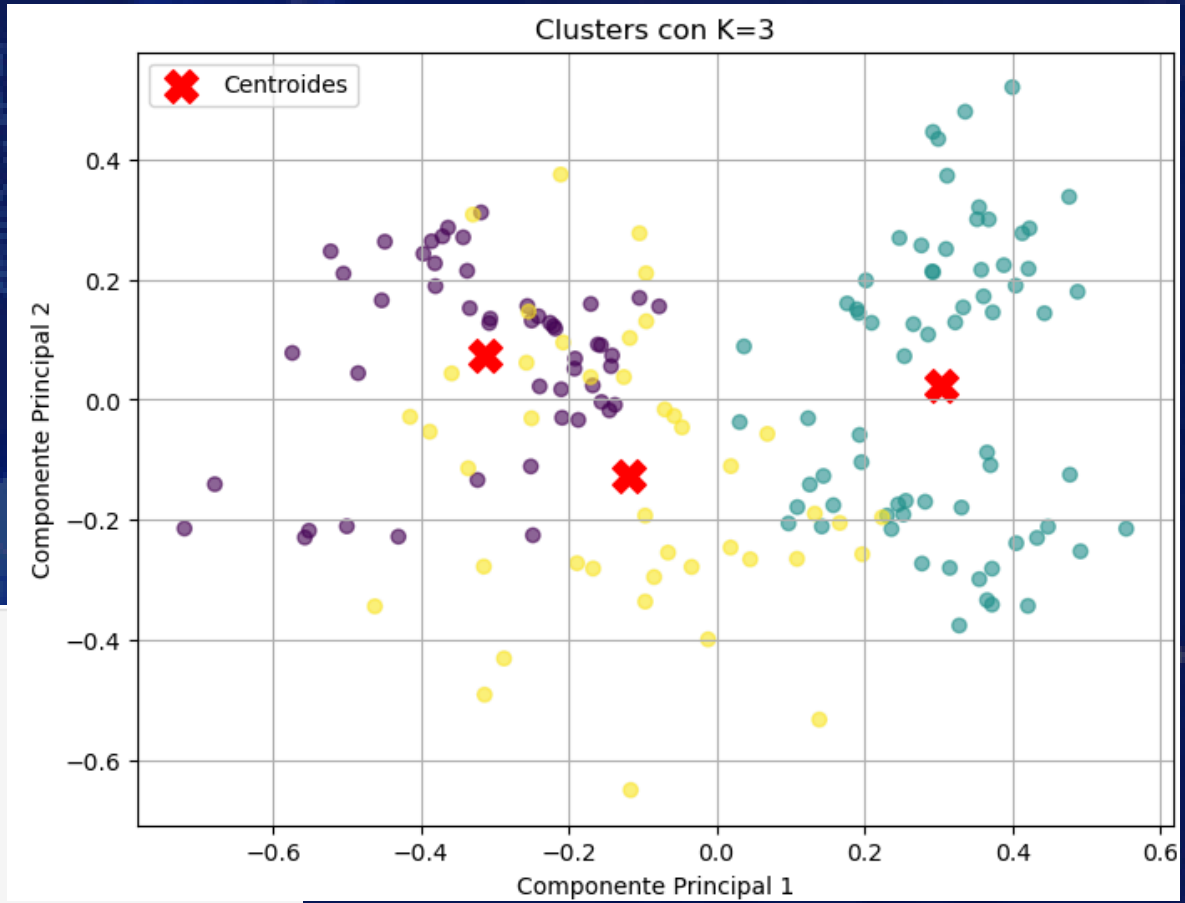
```python
#Interpretación de los clústeres
data_train['Cluster'] = labels
print(data_train.groupby('Cluster').mean())
```

	Age	Annual Income (k$)	Spending Score (1-100)
Cluster			
0	56.184211	55.184211	48.973684
1	33.142857	87.285714	82.257143
2	45.764706	27.117647	17.588235
3	38.925926	87.925926	19.629630
4	25.627907	39.767442	63.139535

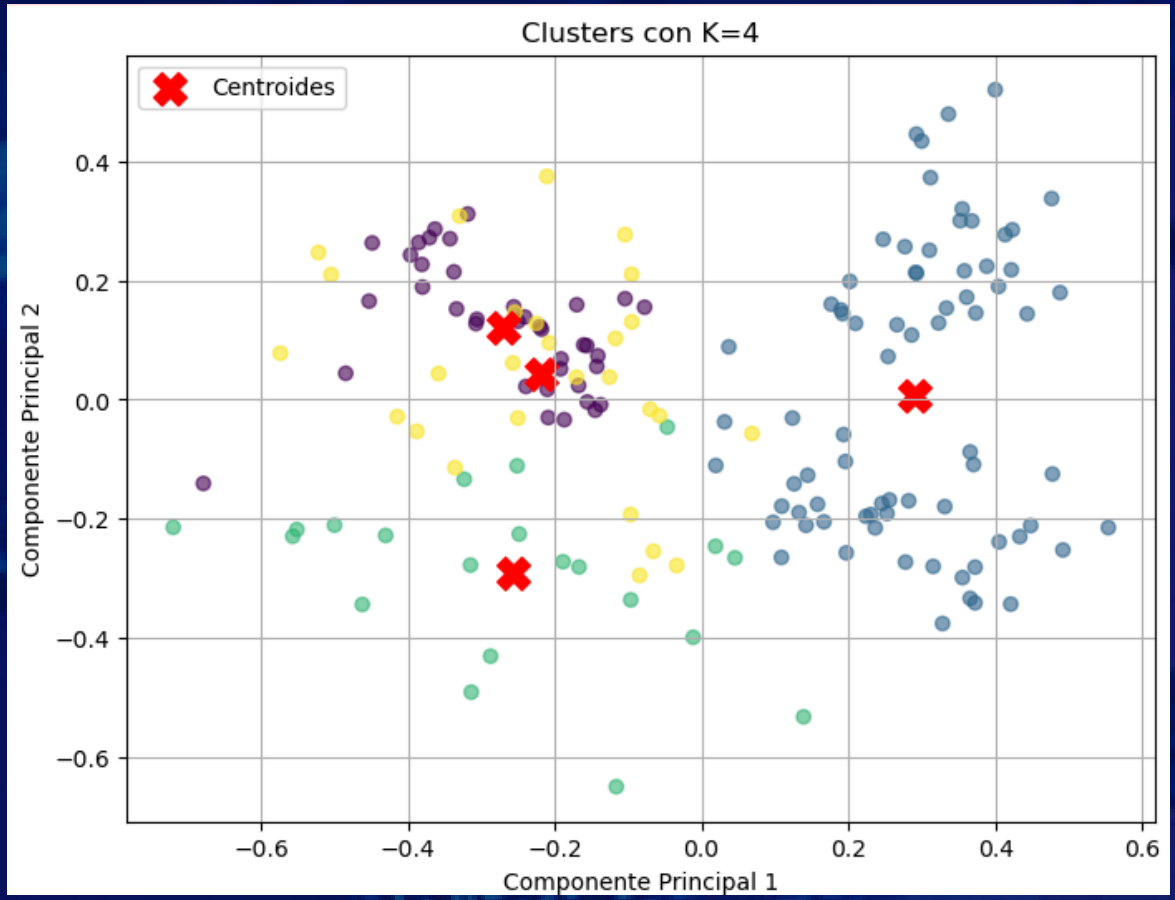
Esta tabla muestra, para cada grupo de clientes encontrado por el algoritmo, cuál es su edad promedio, cuánto ganan y cuánto gastan en promedio. Así, la empresa puede entender fácilmente qué tipo de clientes tiene y cómo dirigirse a cada uno según sus características.

Comparación al variar el número de clústeres

- Al elegir tres clústeres, la segmentación es más general: cada grupo abarca una mayor diversidad de clientes.
- Es posible que algunos matices o subgrupos presentes en los datos se agrupen juntos, perdiendo detalle respecto a segmentaciones con más clústeres.
- Sin embargo, la visualización muestra que los tres grupos principales están bien definidos, lo que puede ser útil para estrategias generales de marketing o comunicación.

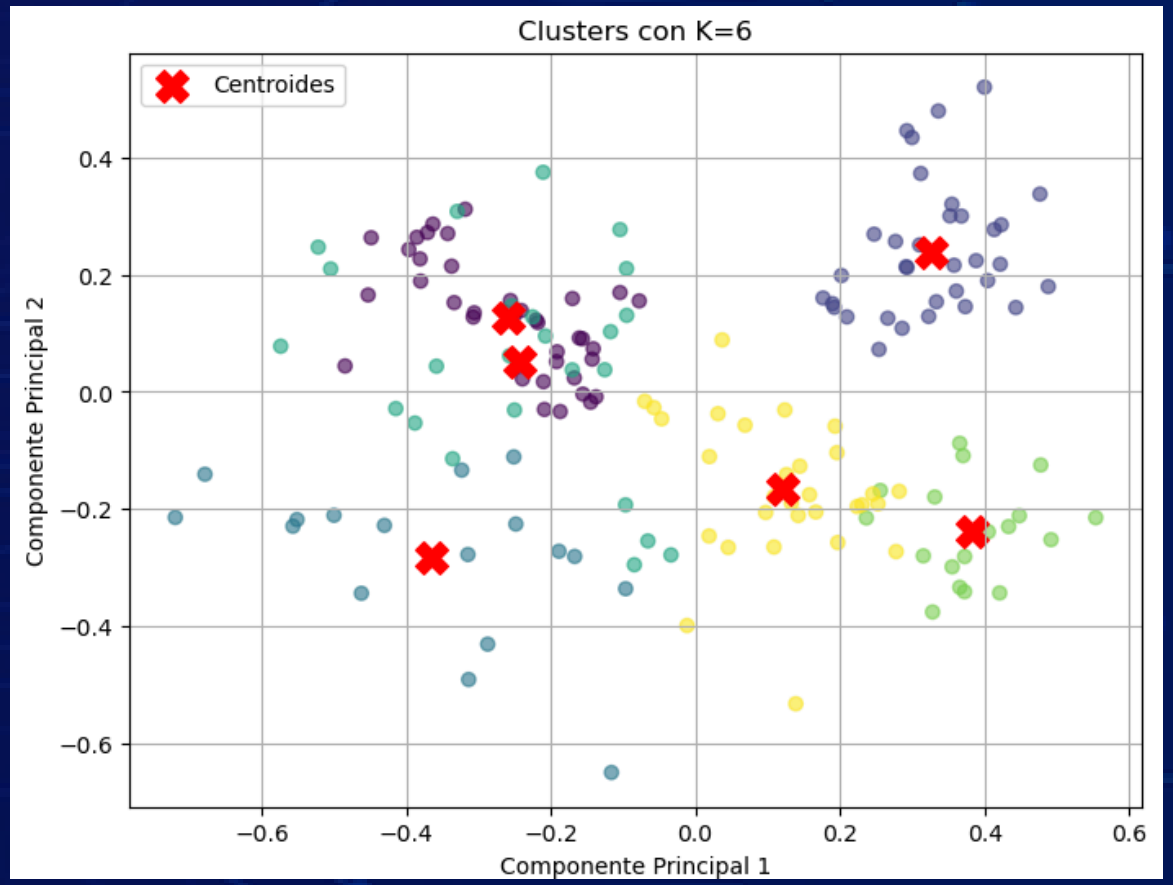



```python
#Comparación al variar el valor de K
for new_k in [3, 4, 6]:
    kmeans_new = KMeans(n_clusters=new_k, random_state=RANDOM_STATE)
    kmeans_new.fit(data_train_scaled)
    labels_new = kmeans_new.labels_
    centroides_new = kmeans_new.cluster_centers_
    centroides_new_pca = pca.transform(centroides_new)
    plt.figure(figsize=(8,6))
    plt.scatter(data_pca[:,0], data_pca[:,1], c=labels_new, cmap='viridis', alpha=0.6)
    plt.scatter(centroides_new_pca[:,0], centroides_new_pca[:,1], c='red', marker='X', s=200, label='Centroides')
    plt.title(f'Clusters con K={new_k}')
    plt.xlabel('Componente Principal 1')
    plt.ylabel('Componente Principal 2')
    plt.legend()
    plt.grid(True)
    plt.show()
```



La imagen muestra una agrupación general de los clientes en 4 grupos, pero el análisis previo indica que una segmentación en 5 o 6 clústeres sería más adecuada y permitiría descubrir diferencias más relevantes entre los clientes.

- K=6 es coherente con el método del codo: Según el análisis previo, el método del codo sugería que el número óptimo de clústeres está entre 5 y 6. Por lo tanto, esta segmentación es adecuada y permite identificar perfiles de clientes más específicos.

- Mayor detalle en la segmentación: Al usar 6 clústeres, se logra una segmentación más fina, permitiendo descubrir subgrupos que podrían pasar desapercibidos con menos clústeres.

- Utilidad para el negocio: Esta segmentación detallada facilita la creación de estrategias personalizadas para cada grupo, mejorando la efectividad de campañas de marketing, promociones o servicios.