

✓ Resumen del proyecto: Customer Personality Clustering

Objetivo

Segmentar clientes en grupos homogéneos según su perfil económico y comportamiento de compra, con el fin de:

- Diseñar estrategias de marketing personalizadas
 - Mejorar la fidelización
 - Optimizar la asignación de recursos comerciales
-

Proceso realizado

1. Carga y exploración de datos

- Dataset original de 2.240 clientes
- Limpieza y depuración:
 - Eliminación de duplicados
 - Agrupación de categorías raras en `Marital_Status`
 - Imputación de valores nulos en `Income`

2. Ingeniería de características

- Creación de variables:
 - `Age` : Edad actual
 - `Children` : Total de hijos en el hogar
 - `Total_Spent` : Gasto acumulado en los últimos 2 años
 - `Total_Purchases` : Total de compras en todos los canales
 - `Campaigns_Accepted` : Número de campañas aceptadas
 - `Customer_Since_Days` : Antigüedad del cliente en días
- Eliminación de columnas redundantes e identificadores

3. Codificación y escalado

- One-Hot Encoding en:
 - `Education`
 - `Marital_Status`
- Escalado estándar (`StandardScaler`) en todas las variables

4. Clustering

- Modelo utilizado: **KMeans**
- Número de clusters: **4**
- Asignación de segmento a cada cliente
- Evaluación de la calidad:
 - Silhouette Score: **0.119**
 - Distribución equilibrada de clientes por segmento

5. Visualización

- Reducción dimensional con PCA (2 componentes) y t-SNE (2 componentes)
- Gráficos de dispersión coloreados por segmento
- t-SNE permitió identificar mejor la distribución de grupos

6. Guardado de artefactos

- Modelo entrenado serializado (`kmeans_customer_segments.pkl`)
 - Dataset final con segmentos (`customer_segments.csv`)
-

Resultados y conclusiones

- El clustering logró identificar **4 segmentos diferenciados** de clientes.
- Cada grupo presenta patrones distintos en gasto total, antigüedad, ingresos y campañas aceptadas.
- Las visualizaciones de t-SNE mostraron una separación más clara que PCA.
- Este modelo permite:
 - Clasificar nuevos clientes en segmentos
 - Analizar la rentabilidad y perfil de cada grupo

- Aplicar estrategias de marketing específicas
-

✓ Paso 0 – Importación de librerías

```
#@title Paso 0 – Importación de librerías
```

```
# Manipulación de datos
```

```
import pandas as pd
```

```
import numpy as np
```

```
# Visualización
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
# Preprocesamiento
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.decomposition import PCA
```

```
from sklearn.pipeline import Pipeline
```

```
from sklearn.base import BaseEstimator, TransformerMixin
```

```
# Clustering
```

```
from sklearn.cluster import KMeans
```

```
from sklearn.metrics import silhouette_score
```

```
from sklearn.manifold import TSNE
```

```
# Guardar y cargar modelos
```

```
import joblib
```

```
print("Librerías cargadas correctamente.")
```

```
↔ Librerías cargadas correctamente.
```

✓ Paso 1 Cargar datos Customer Personality Analysis

```
#@title Paso 1 Cargar datos Customer Personality Analysis
```

```
df = pd.read_csv("Customer_Personality.csv", sep='\t')
```

✓ Paso 2: Limpieza y Feature Engineering

```
#@title Paso 2: Limpieza y Feature Engineering
```

```
# Copia de seguridad por si se quiere conservar el original
```

```
df = df.copy()
```

```
# --- 1. Conversión de fechas y cálculo de antigüedad ---
```

```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], dayfirst=True)
```

```
df['Customer_Since_Days'] = (df['Dt_Customer'].max() - df['Dt_Customer']).dt.days
```

```
# --- 2. Nuevas variables derivadas ---
```

```
df['Age'] = 2025 - df['Year_Birth']
```

```
df['Children'] = df['Kidhome'] + df['Teenhome']
```

```
# Gasto total
```

```
df['Total_Spent'] = df[['MntWines', 'MntFruits', 'MntMeatProducts',  
                      'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']].sum(axis=1)
```

```
# Compras totales
```

```
df['Total_Purchases'] = df[['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases']].sum(axis=1)
```

```
# Total campañas aceptadas
```

```
df['Campaigns_Accepted'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']].sum(axis=1)
```

```
# --- 3. Limpieza de columnas ---
```

```
df.drop(columns=[  
    'ID', 'Year_Birth', 'Dt_Customer',  
    'Kidhome', 'Teenhome',  
    'Z_CostContact', 'Z_Revenue',  
    'AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',  
    'Response',  
    'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',  
    'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases'  
], inplace=True)
```

```
# --- 4. Limpieza de datos ---
```

```
df.drop_duplicates(inplace=True)
```

```
df['Income'] = df['Income'].fillna(df['Income'].median())
```

```
# Agrupar valores extraños de estado civil
df['Marital_Status'] = df['Marital_Status'].replace({
    'Alone': 'Other', 'Absurd': 'Other', 'YOLO': 'Other'
})

print("Limpieza y feature engineering completados.")
```

→ Limpieza y feature engineering completados.

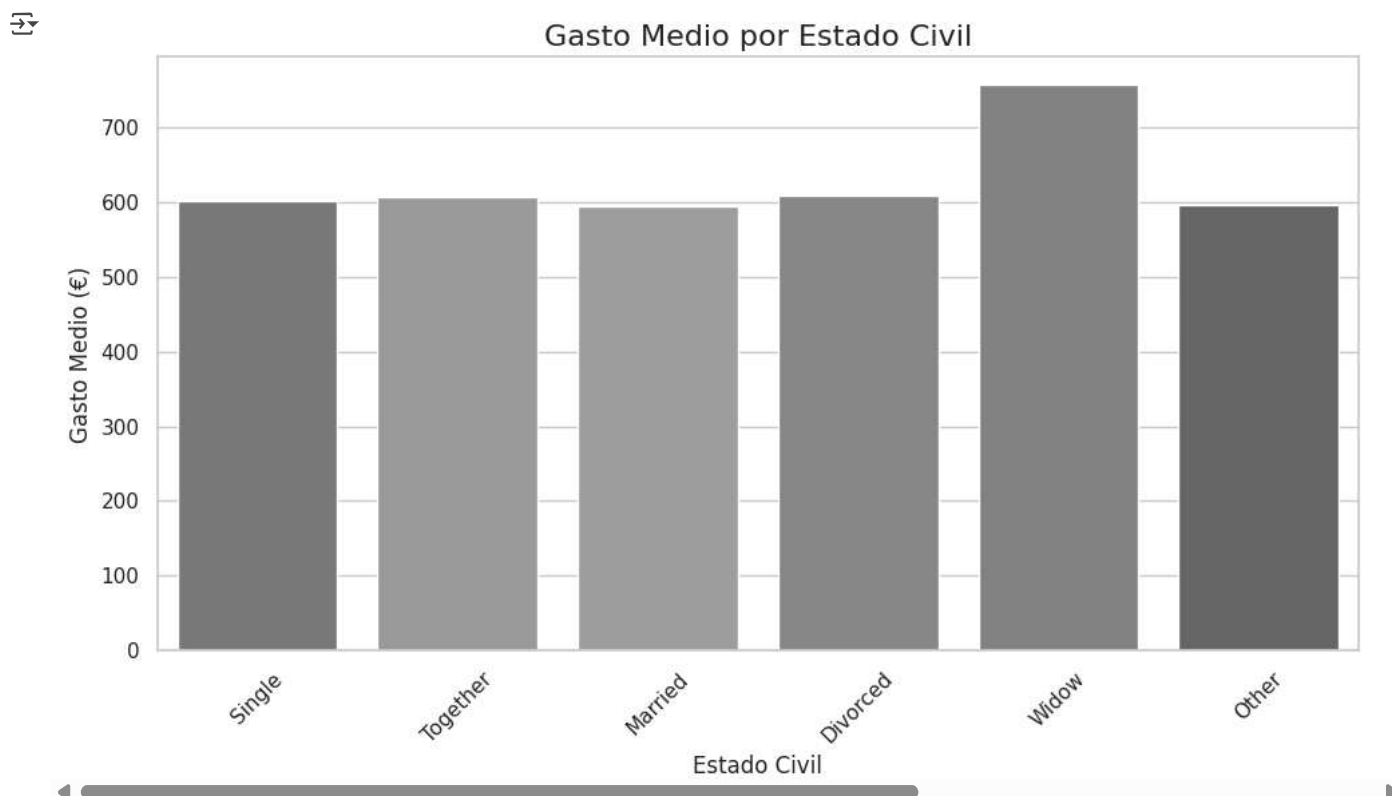
▼ Gráfico de gasto medio por estado civil

```
# @title Gráfico de gasto medio por estado civil

sns.set(style="whitegrid")

plt.figure(figsize=(10, 6))
sns.barplot(
    data=df,
    x='Marital_Status',
    y='Total_Spent',
    estimator='mean',
    errorbar=None,
    hue='Marital_Status', # evita warning de `palette`
    palette='muted',
    legend=False # oculta la leyenda innecesaria
)

plt.title('Gasto Medio por Estado Civil', fontsize=16)
plt.xlabel('Estado Civil', fontsize=12)
plt.ylabel('Gasto Medio (€)', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



▼ Visualización Tabla

```
# @title Visualización Tabla
# Con la función head se muestra las 5 primeras filas del archivo.
df.head()
```

	Education	Marital_Status	Income	Recency	NumDealsPurchases	NumWebVisitsMonth	Complain	Customer_Since_Days	Age	Children	T
0	Graduation	Single	58138.0	58	3	7	0	663	68	0	
1	Graduation	Single	46344.0	38	2	5	0	113	71	2	
2	Graduation	Together	71613.0	26	1	4	0	312	60	0	
3	Graduation	Together	26646.0	26	2	6	0	139	41	1	
4	PhD	Married	58293.0	94	5	5	0	161	44	1	

Resumen estadístico general

#@title Resumen estadístico general

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Income	2039.0	52352.305542	25429.762373	1730.0	35702.5	51537.0	68298.5	666666.0
Recency	2039.0	49.096616	28.974507	0.0	24.0	49.0	74.0	99.0
NumDealsPurchases	2039.0	2.329083	1.934170	0.0	1.0	2.0	3.0	15.0
NumWebVisitsMonth	2039.0	5.310446	2.438496	0.0	3.0	6.0	7.0	20.0
Complain	2039.0	0.009809	0.098576	0.0	0.0	0.0	0.0	1.0
Customer_Since_Days	2039.0	351.848455	201.931177	0.0	178.5	351.0	527.0	699.0
Age	2039.0	56.231976	11.983086	29.0	48.0	55.0	66.0	132.0
Children	2039.0	0.952428	0.748694	0.0	0.0	1.0	1.0	3.0
Total_Spent	2039.0	605.871015	602.092007	5.0	68.5	396.0	1044.5	2525.0
Total_Purchases	2039.0	12.529671	7.188406	0.0	6.0	12.0	18.0	32.0
Campaigns_Accepted	2039.0	0.299166	0.678966	0.0	0.0	0.0	0.0	4.0

Info de tipo de datos, nulos, etc.

#@title Info de tipo de datos, nulos, etc.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2039 entries, 0 to 2239
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Education              2039 non-null  object
1   Marital_Status         2039 non-null  object
2   Income                 2039 non-null  float64
3   Recency                2039 non-null  int64
4   NumDealsPurchases      2039 non-null  int64
5   NumWebVisitsMonth      2039 non-null  int64
6   Complain               2039 non-null  int64
7   Customer_Since_Days    2039 non-null  int64
8   Age                   2039 non-null  int64
9   Children               2039 non-null  int64
10  Total_Spent            2039 non-null  int64
11  Total_Purchases        2039 non-null  int64
12  Campaigns_Accepted     2039 non-null  int64
dtypes: float64(1), int64(10), object(2)
memory usage: 223.0+ KB
```

Leyenda con nuevas variables

Columna	Tipo	Descripción
Education	Categoría	Nivel educativo del cliente (e.g., Graduation, PhD, Master, etc.)
Marital_Status	Categoría	Estado civil del cliente
Income	Númerica	Ingreso anual familiar del cliente (en euros)
Recency	Númerica	Días desde la última compra del cliente
NumDealsPurchases	Númerica	Número de compras realizadas con descuento
NumWebVisitsMonth	Númerica	Número de visitas al sitio web en el último mes
Complain	Binaria	1 = se quejó en los últimos 2 años, 0 = no se quejó
Customer_Since_Days	Númerica	Antigüedad del cliente en días desde su alta en la empresa
Age	Númerica	Edad del cliente (calculada como 2025 - Year_Birth)

Columna	Tipo	Descripción
Children	Númerica	Número total de hijos (Kidhome + Teenhome)
Total_Spent	Númerica	Gasto total acumulado en productos (vino, carne, frutas, oro, etc.)
Total_Purchases	Númerica	Número total de compras por cualquier canal
Campaigns_Accepted	Númerica	Número total de campañas promocionales aceptadas (de 0 a 5)

✓ Paso 3: Codificación de variables categóricas + Escalado

#@title Paso 3: Codificación de variables categóricas + Escalado

1 Codificación One-Hot de variables categóricas

```
df_encoded = pd.get_dummies(
    df,
    columns=['Education', 'Marital_Status'],
    drop_first=True
)
```

```
print("Codificación completada.")
print(f"Columnas resultantes: {df_encoded.columns.tolist()}")
```

2 Escalado de todas las variables

```
scaler = StandardScaler()
df_scaled = pd.DataFrame(
    scaler.fit_transform(df_encoded),
    columns=df_encoded.columns
)
```

```
print("Escalado completado.")
```

3 Verificación de la forma del dataset final

```
print(f"Dimensiones finales del dataset: {df_scaled.shape}")
```

4 Vista previa

```
df_scaled.head()
```

→ ☒ Codificación completada.
Columnas resultantes: ['Income', 'Recency', 'NumDealsPurchases', 'NumWebVisitsMonth', 'Complain', 'Customer_Since_Days', 'Age', 'Children', 'Total_Spent', 'Total_Purchases', 'Campaigns_Accepted']
☒ Escalado completado.
Dimensiones finales del dataset: (2039, 20)

	Income	Recency	NumDealsPurchases	NumWebVisitsMonth	Complain	Customer_Since_Days	Age	Children	Total_Spent	Total_Purchases	Campaigns_Accepted
0	0.227572	0.307359	0.346961	0.693037	-0.099528	1.541257	0.982294	-1.272431	1.679772	0.063556	0.063556
1	-0.236329	-0.383073	-0.170183	-0.127342	-0.099528	-1.183111	1.232708	1.399543	-0.961669	0.063556	0.063556
2	0.757593	-0.797331	-0.687328	-0.537531	-0.099528	-0.197385	0.314522	-1.272431	0.282632	0.063556	0.063556
3	-1.011123	-0.797331	-0.170183	0.282848	-0.099528	-1.054323	-1.271435	0.063556	-0.918475	0.063556	0.063556
4	0.233669	1.550135	1.381250	-0.127342	-0.099528	-0.945348	-1.021021	0.063556	-0.305462	0.063556	0.063556

✓ Paso 4: Entrenamiento del clustering y análisis de segmentos (corregido)

#@title Paso 4: Entrenamiento del clustering y análisis de segmentos (corregido)

1 Entrenar el modelo

```
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(df_scaled)
```

2 Asignar segmentos

```
df["Segment"] = kmeans.labels_
```

```
print(" Clustering completado.")
```

3 Silhouette Score

```
sil_score = silhouette_score(df_scaled, kmeans.labels_)
print(f" Silhouette Score: {sil_score:.3f}")
```


4 Resumen de variables numéricas solamente


```
numerical_cols = ['Income', 'Recency', 'NumDealsPurchases', 'NumWebVisitsMonth',
                  'Complain', 'Customer_Since_Days', 'Age', 'Children',
                  'Total_Spent', 'Total_Purchases', 'Campaigns_Accepted']
```


```
segment_summary = df.groupby("Segment")[numerical_cols].mean().round(1)
print("\n Resumen de cada segmento (variables numéricas):")
display(segment_summary)
```

```
# 5 Distribución de clientes
counts = df["Segment"].value_counts().sort_index()
print("\n Número de clientes por segmento:")
print(counts)


# 6 Vista previa
df.head()
```

 Clustering completado.

 Silhouette Score: 0.119

 Resumen de cada segmento (variables numéricas):

	Income	Recency	NumDealsPurchases	NumWebVisitsMonth	Complain	Customer_Since_Days	Age	Children	Total_Spent	Total_Pu
Segment										
0	48208.7	47.6	2.6	5.6	0.0	323.7	58.2	1.1	412.9	
1	39053.8	49.7	2.7	6.5	0.0	351.3	55.1	1.3	211.2	
2	74749.6	49.0	1.7	3.4	0.0	366.1	58.0	0.4	1288.1	
3	40601.0	49.2	2.6	6.3	0.0	347.4	53.3	1.1	285.7	

 Número de clientes por segmento:

Segment

0 283

1 803

2 675

3 278

Name: count, dtype: int64

	Education	Marital_Status	Income	Recency	NumDealsPurchases	NumWebVisitsMonth	Complain	Customer_Since_Days	Age	Children	T
0	Graduation	Single	58138.0	58	3	7	0	663	68	0	
1	Graduation	Single	46344.0	38	2	5	0	113	71	2	
2	Graduation	Together	71613.0	26	1	4	0	312	60	0	
3	Graduation	Together	26646.0	26	2	6	0	139	41	1	
4	PhD	Married	58293.0	94	5	5	0	161	44	1	

▼ Paso 5: Reducción dimensional con PCA y visualización de clusters

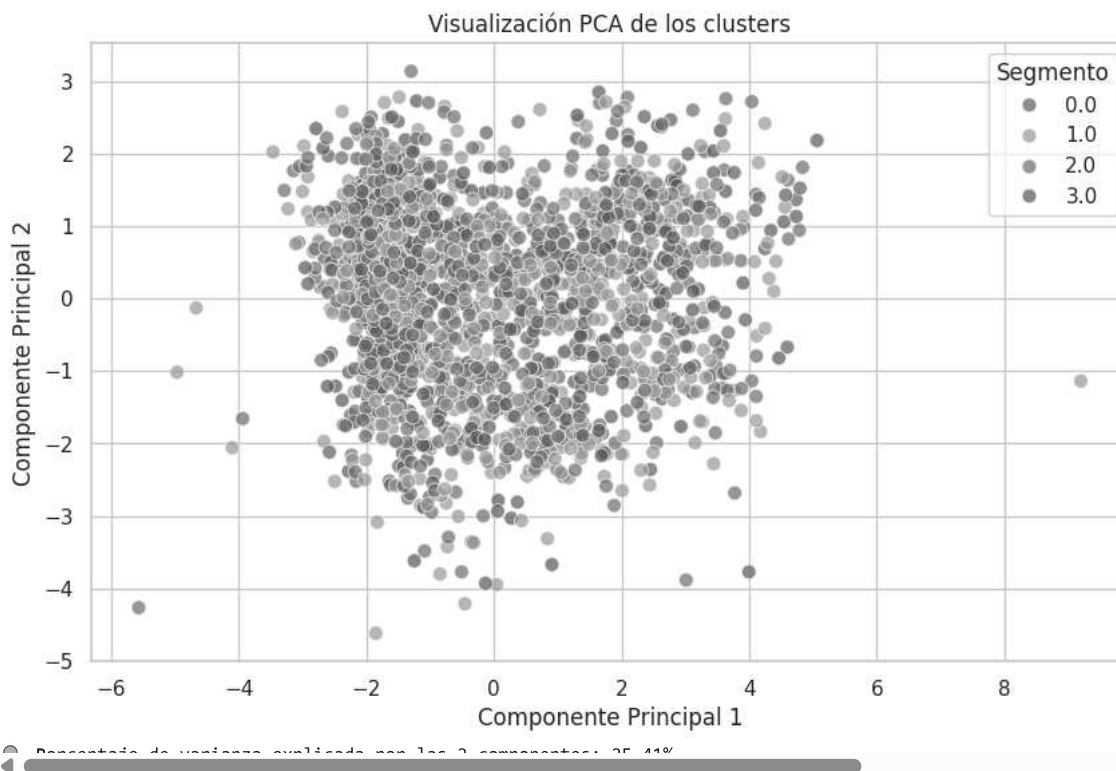
```
#title Paso 5: Reducción dimensional con PCA y visualización de clusters

# 1 Reducir a 2 dimensiones
pca = PCA(n_components=2, random_state=42)
pca_components = pca.fit_transform(df_scaled)

# 2 Crear DataFrame con componentes y segmentos
pca_df = pd.DataFrame(pca_components, columns=["PC1", "PC2"])
pca_df["Segment"] = df["Segment"]

# 3 Visualización
plt.figure(figsize=(10,6))
sns.scatterplot(
    x="PC1",
    y="PC2",
    hue="Segment",
    palette="tab10",
    data=pca_df,
    alpha=0.7,
    s=60
)
plt.title("Visualización PCA de los clusters")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.legend(title="Segmento")
plt.show()

# 4 Varianza explicada
explained_variance = pca.explained_variance_ratio_.sum()
print(f" Porcentaje de varianza explicada por las 2 componentes: {explained_variance:.2%}")
```



✓ Paso 6: Guardar modelo y dataset con segmentos

```
#@title Paso 6: Guardar modelo y dataset con segmentos
# Guardar el modelo KMeans
#import joblib

joblib.dump(kmeans, "kmeans_customer_segments.pkl")
print(" Modelo guardado como 'kmeans_customer_segments.pkl'.")

# Guardar el dataset con segmentos
df.to_csv("customer_segments.csv", index=False)
print(" Dataset guardado como 'customer_segments.csv'.")
```



- ☒ Modelo guardado como 'kmeans_customer_segments.pkl'.
- ☒ Dataset guardado como 'customer_segments.csv'.

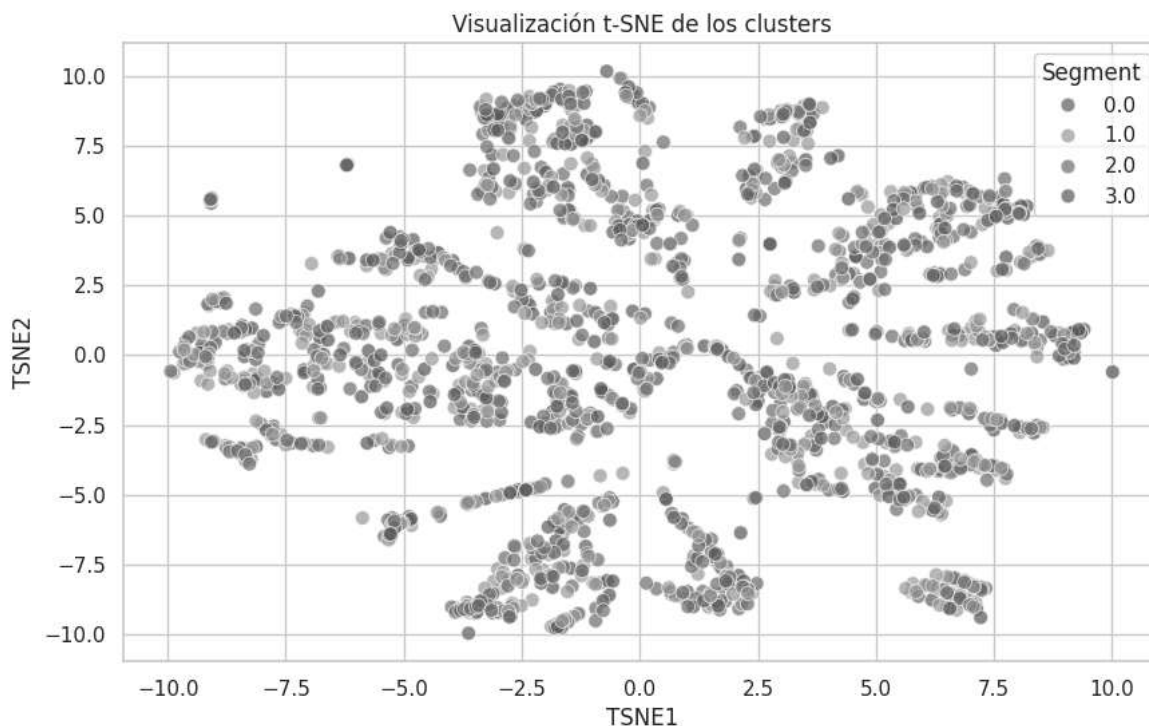
✓ Paso 7: Visualización con t-SNE

```
#@title Paso 7: Visualización con t-SNE

tsne = TSNE(n_components=2, random_state=42, perplexity=40, max_iter=300)
tsne_components = tsne.fit_transform(df_scaled)

tsne_df = pd.DataFrame(tsne_components, columns=["TSNE1", "TSNE2"])
tsne_df["Segment"] = df["Segment"]

plt.figure(figsize=(10,6))
sns.scatterplot(
    x="TSNE1",
    y="TSNE2",
    hue="Segment",
    palette="tab10",
    data=tsne_df,
    alpha=0.7,
    s=60
)
plt.title("Visualización t-SNE de los clusters")
plt.show()
```



▼ Paso 8: Preparar predicciones en nuevos datos

```
#@title Paso 8: Preparar predicciones en nuevos datos
```

```
# Ejemplo: Predecir el segmento de un nuevo cliente
nuevo_cliente = df_scaled.iloc[[0]]
segmento_predicho = kmeans.predict(nuevo_cliente)[0]
print(f"Segmento asignado al nuevo cliente: {segmento_predicho}")
```



Segmento asignado al nuevo cliente: 2

▼ Paso 9: Tabla resumen de medias por segmento

```
#@title Paso 9: Tabla resumen de medias por segmento
```

```
vars_bar = [
    'Income',
    'Age',
    'Total_Spent',
    'Total_Purchases',
    'Customer_Since_Days',
    'Children',
    'Recency'
]

# Crear tabla resumen
tabla_resumen = df.groupby('Cluster_Nombre_Abreve')[vars_bar].mean().round(1)

print(" Tabla de medias por segmento:")
display(tabla_resumen)
```



☒ Tabla de medias por segmento:

	Income	Age	Total_Spent	Total_Purchases	Customer_Since_Days	Children	Recency
Cluster_Nombre_Abreve							
Alto valor	74749.6	58.0	1288.1	19.8	366.1	0.4	49.0
Desconectados	39053.8	55.1	211.2	8.1	351.3	1.3	49.7
Estrella	40601.0	53.3	285.7	9.2	347.4	1.1	49.2
Tradicionales	48208.7	58.2	412.9	11.0	323.7	1.1	47.6

▼ Paso 10: Gráficos de barras con medias reales

```
#@title Paso 10: Gráficos de barras con medias reales
```

```
# Reset index para graficar
```



```

cluster_means = tabla_resumen.reset_index()

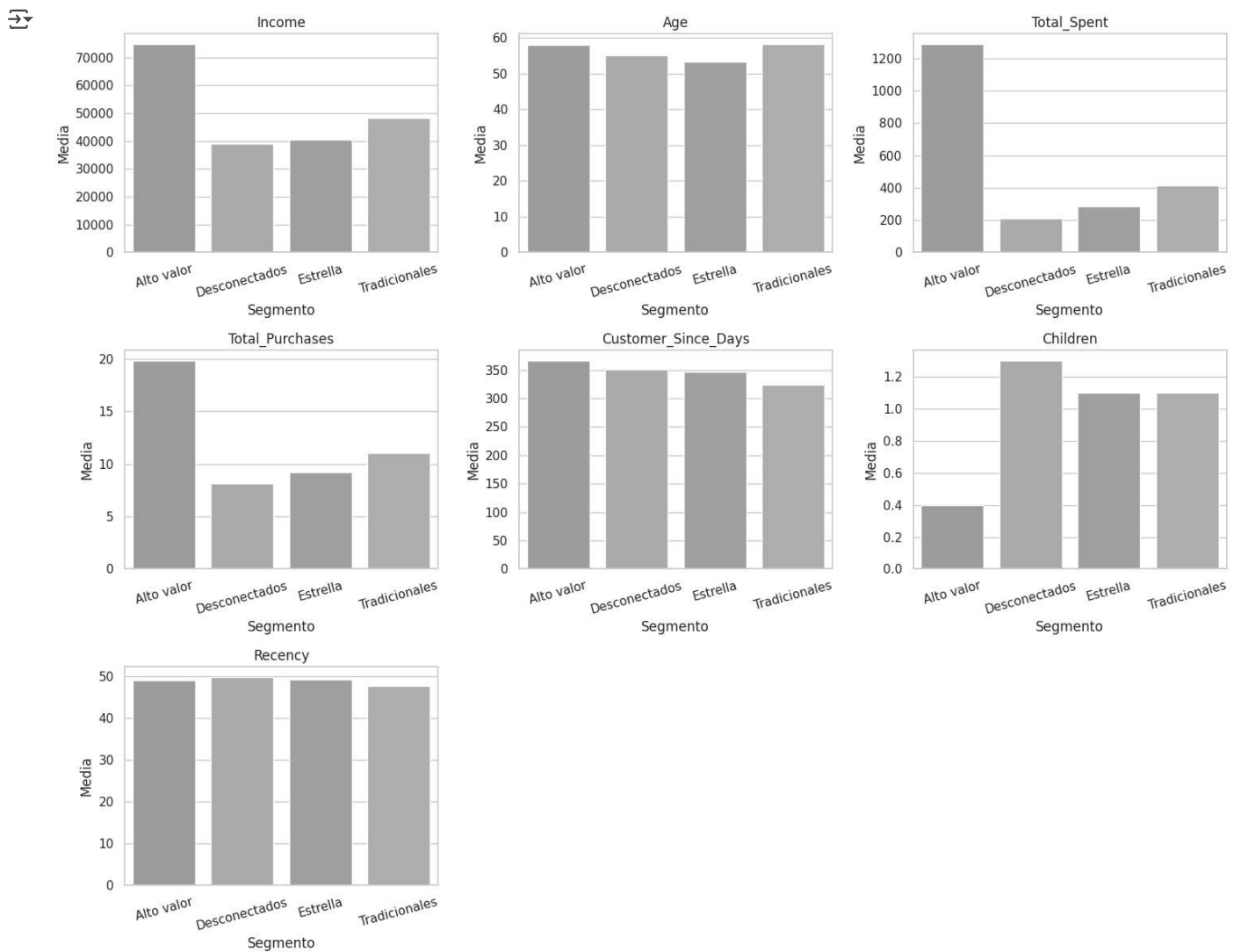
cols = 3
rows = -(-len(vars_bar) // cols)
fig, axes = plt.subplots(rows, cols, figsize=(cols * 5, rows * 4))
axes = axes.flatten()

for i, var in enumerate(vars_bar):
    sns.barplot(
        data=cluster_means,
        x='Cluster_Nombre_Abrev',
        y=var,
        hue='Cluster_Nombre_Abrev',
        palette='Set2',
        legend=False,
        ax=axes[i]
    )
    axes[i].set_title(var, fontsize=12)
    axes[i].set_ylabel('Media')
    axes[i].set_xlabel('Segmento')
    axes[i].tick_params(axis='x', rotation=15)
    axes[i].grid(True, axis='y')

# Quitar subgráficos vacíos si sobran
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()

```



Paso 11 Radar Plot comparativo de segmentos

```

#@title Paso 11 Radar Plot comparativo de segmentos

```

Figura 100: 22 Radar plot comparativo de segmentos

```
# Normalizar cada variable 0-1 para radar
radar_data = tabla_resumen.copy()
for col in radar_data.columns:
    min_ = radar_data[col].min()
    max_ = radar_data[col].max()
    radar_data[col] = (radar_data[col] - min_) / (max_ - min_)

# Variables
labels = radar_data.columns
n_vars = len(labels)

# Ángulos
angles = np.linspace(0, 2 * np.pi, n_vars, endpoint=False).tolist()
angles += angles[:1]

# Crear figura
fig, ax = plt.subplots(figsize=(8,8), subplot_kw=dict(polar=True))

# Plotear cada segmento
for idx, row in radar_data.iterrows():
    values = row.tolist()
    values += values[:1]
    ax.plot(angles, values, label=idx)
    ax.fill(angles, values, alpha=0.25)

# Ajustar etiquetas
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels, fontsize=11)
ax.set_yticks([0.2,0.4,0.6,0.8])
ax.set_yticklabels(['20%', '40%', '60%', '80%'])
ax.set_title('Comparación de segmentos (Radar plot)', size=12, pad=40)
ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))
plt.show()
```

