# A CONTINUACIÓN SE PRESENTA EL CONSOLIDADO DE LAS ACTIVIDADES

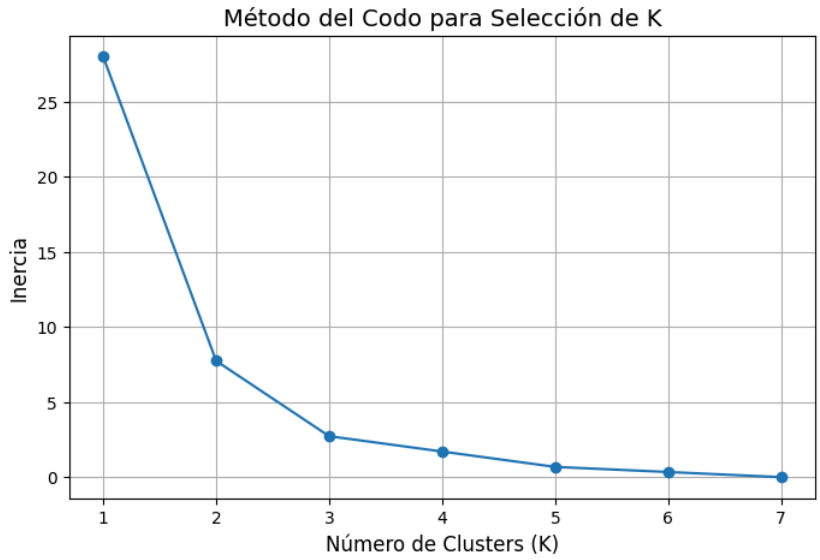
## Actividad 2 - Búsqueda y sistemas basados en reglas

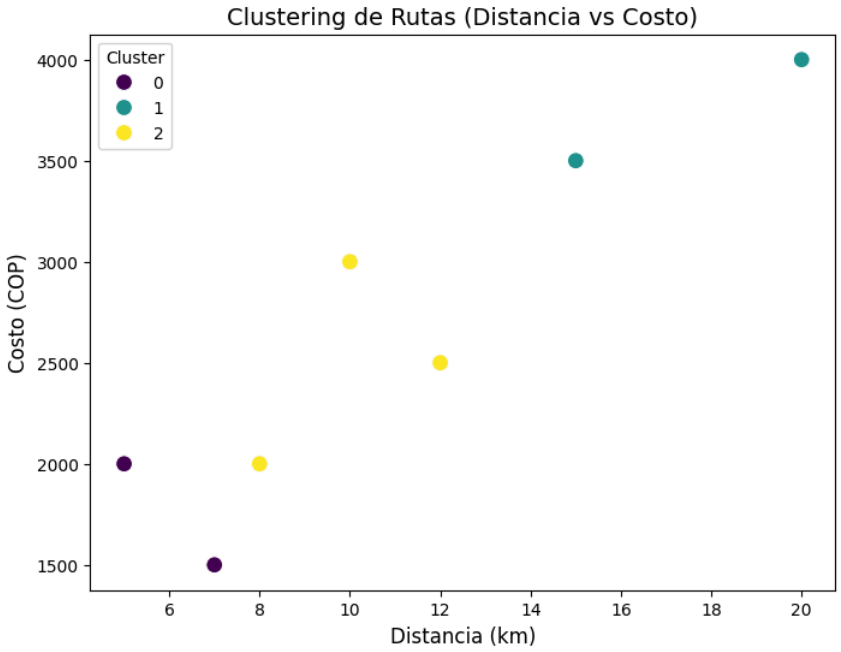
[ ] ↳ *32 celdas ocultas*

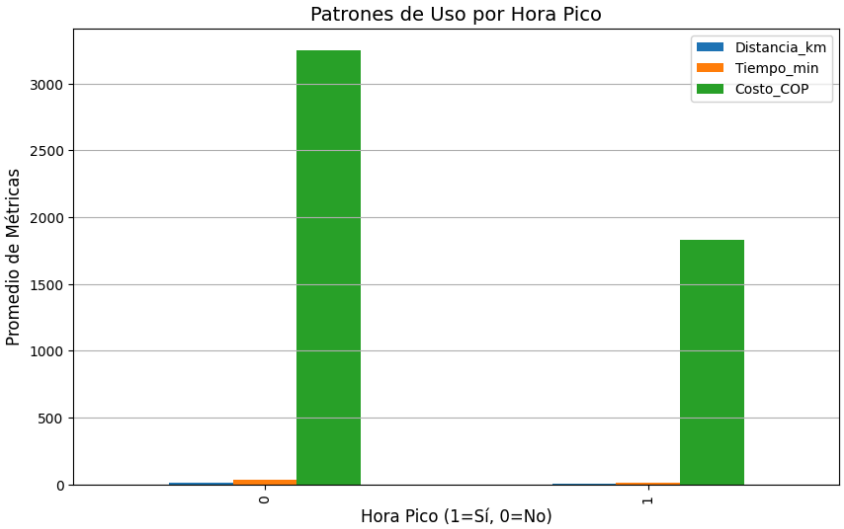
## Actividad 4 - Métodos de aprendizaje no supervisado

Análisis no supervisado para clustering de rutas y detección de patrones de uso horario

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Simular datos de rutas para el análisis
data = {
    'Distancia_km': [5, 10, 7, 12, 8, 15, 20],
    'Tiempo_min': [10, 25, 15, 30, 20, 40, 50],
    'Costo_COP': [2000, 3000, 1500, 2500, 2000, 3500, 4000],
    'Transbordos': [0, 1, 0, 1, 1, 2, 2],
    'Disponible': [1, 1, 1, 1, 0, 1, 1]  # 1=Disponible, 0=No Disponible
}
rutas_df = pd.DataFrame(data)

# Escalar los datos para clustering
scaler = StandardScaler()
scaled_data = scaler.fit_transform(rutas_df[['Distancia_km', 'Tiempo_min', 'Costo_COP', 'Transbordo

# Determinar el número óptimo de clusters usando el método del codo
inertia = []
k_range = range(1, min(len(rutas_df) + 1, 10))
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)

# Graficar el método del codo
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.title('Método del Codo para Selección de K', fontsize=14)
plt.xlabel('Número de Clusters (K)', fontsize=12)
plt.ylabel('Inercia', fontsize=12)
plt.grid()
plt.show()

# Aplicar KMeans con el número óptimo de clusters (asumimos K=3 para este ejemplo)
kmeans = KMeans(n_clusters=3, random_state=42)
rutas_df['Cluster'] = kmeans.fit_predict(scaled_data)

# Visualizar los clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=rutas_df, x='Distancia_km', y='Costo_COP', hue='Cluster', palette='viridis', s
plt.title('Clustering de Rutas (Distancia vs Costo)', fontsize=14)
plt.xlabel('Distancia (km)', fontsize=12)
plt.ylabel('Costo (COP)', fontsize=12)
plt.legend(title='Cluster')
plt.show()

# Análisis de patrones de uso horario o estacionalidad
# Simular datos de horarios para análisis
rutas_df['Hora_Pico'] = [1, 0, 1, 0, 1, 0, 0]  # 1=En hora pico, 0=Fuera de hora pico
hora_pico_summary = rutas_df.groupby('Hora_Pico').mean()

# Visualizar patrones de uso horario
hora_pico_summary[['Distancia_km', 'Tiempo_min', 'Costo_COP']].plot(kind='bar', figsize=(10, 6))
plt.title('Patrones de Uso por Hora Pico', fontsize=14)
plt.xlabel('Hora Pico (1=Sí, 0=No)', fontsize=12)
plt.ylabel('Promedio de Métricas', fontsize=12)
plt.grid(axis='y')
plt.show()

# Mostrar los datos enriquecidos
print(rutas_df)
```



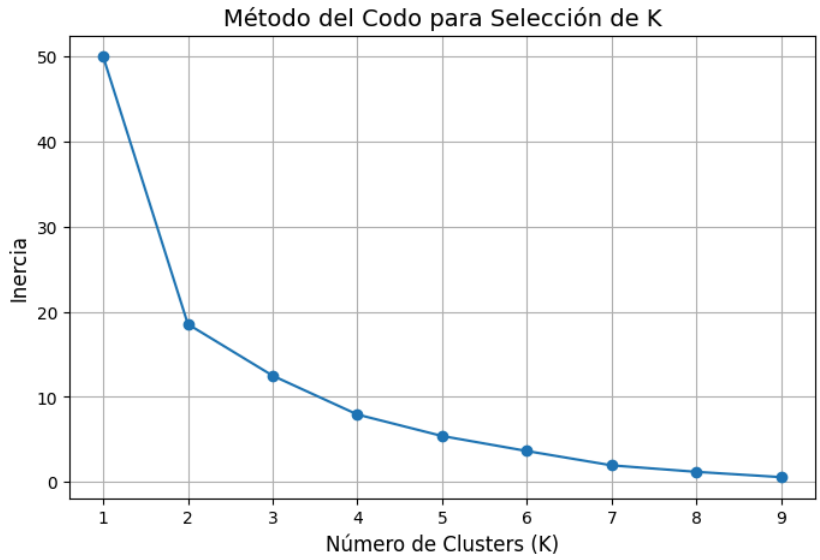

| | Distancia_km | Tiempo_min | Costo_COP | Transbordos | Disponible | Cluster | \ |
|---|---|---|---|---|---|---|---|
| 0 | 5 | 10 | 2000 | 0 | 1 | 0 | |
| 1 | 10 | 25 | 3000 | 1 | 1 | 2 | |
| 2 | 7 | 15 | 1500 | 0 | 1 | 0 | |
| 3 | 12 | 30 | 2500 | 1 | 1 | 2 | |
| 4 | 8 | 20 | 2000 | 1 | 0 | 2 | |
| 5 | 15 | 40 | 3500 | 2 | 1 | 1 | |
| 6 | 20 | 50 | 4000 | 2 | 1 | 1 | |

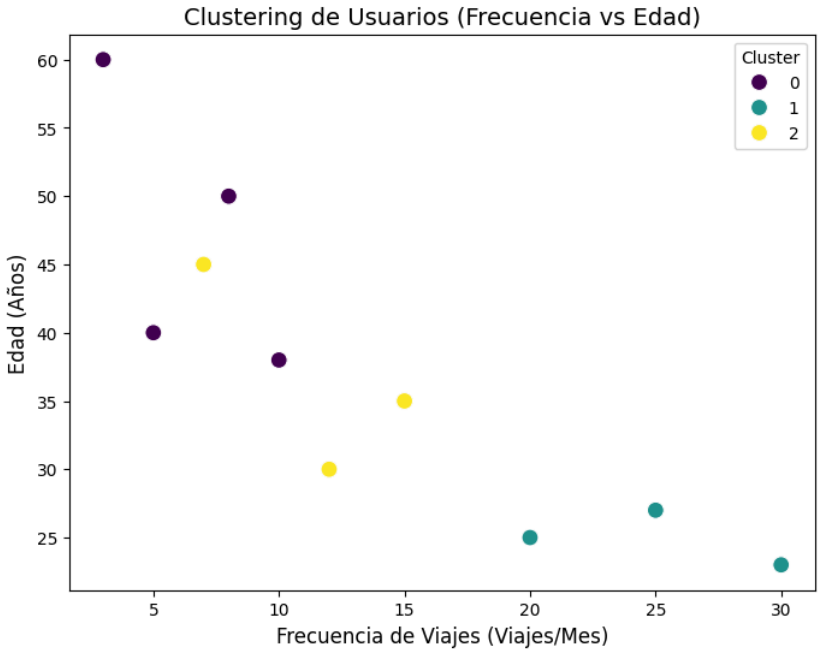
| | Hora_Pico |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 2 | 1 |
| 3 | 0 |

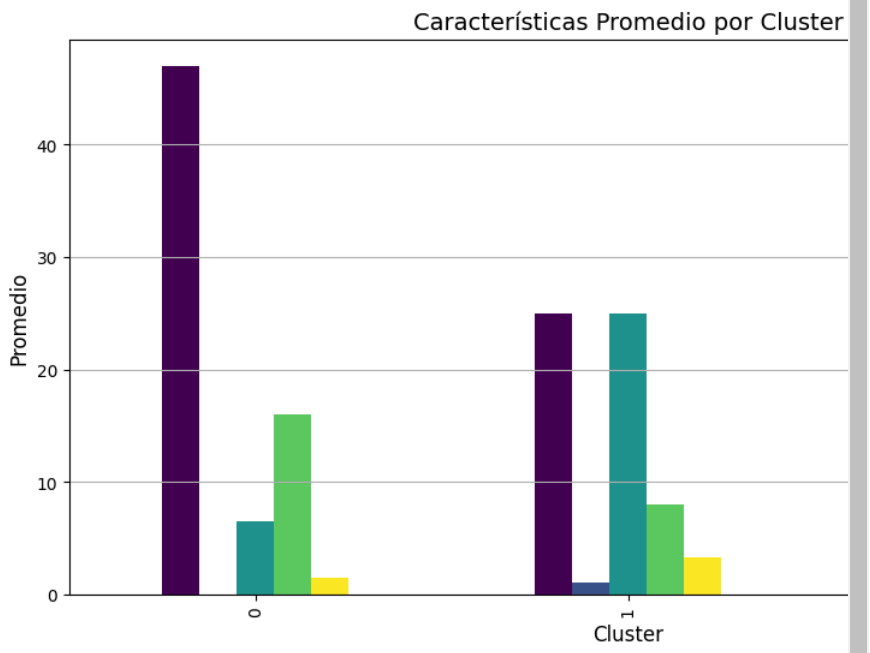
```
4          1
5          0
6          0
```

Segmentación de los usuarios y búsqueda de perfiles basados en los datos de pasajeros

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Simular datos de pasajeros
data = {
    'Edad': [25, 40, 35, 50, 23, 38, 30, 60, 45, 27],
    'Genero': [1, 0, 1, 0, 1, 0, 1, 0, 1, 1],  # 1=Mujer, 0=Hombre
    'Frecuencia_viajes': [20, 5, 15, 8, 30, 10, 12, 3, 7, 25],  # Número de viajes/mes
    'Horario_preferido': [8, 18, 8, 19, 7, 8, 17, 19, 7, 9],  # Hora del día
    'Ocupacion': [3, 2, 4, 1, 3, 2, 4, 1, 3, 4]  # 1=Empleado, 2=Independiente, 3=Estudiante, 4=Otro
}
pasajeros_df = pd.DataFrame(data)

# Escalar los datos para clustering
scaler = StandardScaler()
scaled_data = scaler.fit_transform(pasajeros_df)

# Determinar el número óptimo de clusters usando el método del codo
inertia = []
k_range = range(1, min(len(pasajeros_df) + 1, 10))
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    inertia.append(kmeans.inertia_)

# Graficar el método del codo
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.title('Método del Codo para Selección de K', fontsize=14)
plt.xlabel('Número de Clusters (K)', fontsize=12)
plt.ylabel('Inercia', fontsize=12)
plt.grid()
plt.show()

# Aplicar KMeans con el número óptimo de clusters (asumimos K=3 para este ejemplo)
kmeans = KMeans(n_clusters=3, random_state=42)
pasajeros_df['Cluster'] = kmeans.fit_predict(scaled_data)

# Visualizar los clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=pasajeros_df, x='Frecuencia_viajes', y='Edad', hue='Cluster', palette='viridis'
plt.title('Clustering de Usuarios (Frecuencia vs Edad)', fontsize=14)
plt.xlabel('Frecuencia de Viajes (Viajes/Mes)', fontsize=12)
plt.ylabel('Edad (Años)', fontsize=12)
plt.legend(title='Cluster')
plt.show()

# Resumen de clusters
clusters_summary = pasajeros_df.groupby('Cluster').mean()

# Visualizar características promedio de cada cluster
clusters_summary.plot(kind='bar', figsize=(12, 6), colormap='viridis')
plt.title('Características Promedio por Cluster', fontsize=14)
plt.xlabel('Cluster', fontsize=12)
plt.ylabel('Promedio', fontsize=12)
plt.grid(axis='y')
plt.show()

# Mostrar los datos enriquecidos con la asignación de clusters
print("Segmentación de Usuarios y Clustering:")
print(pasajeros_df)
```




Segmentación de Usuarios y Clustering:

| | Edad | Genero | Frecuencia_viajes | Horario_preferido | Ocupacion | Cluster |
|---|---|---|---|---|---|---|
| 0 | 25 | 1 | 20 | 8 | 3 | 1 |
| 1 | 40 | 0 | 5 | 18 | 2 | 0 |
| 2 | 35 | 1 | 15 | 8 | 4 | 2 |
| 3 | 50 | 0 | 8 | 19 | 1 | 0 |
| 4 | 23 | 1 | 30 | 7 | 3 | 1 |
| 5 | 38 | 0 | 10 | 8 | 2 | 0 |
| 6 | 30 | 1 | 12 | 17 | 4 | 2 |