Explicación Completa del Código en tu Notebook

1. Carga y Preparación de los Datos

Código:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# 📌 2. Cargar y preparar los datos
file_path = "data_2/country_resumido.csv"  # Ajusta la ruta del
archivo
df = pd.read_csv(file_path, delimiter=";")  # Cargar CSV con separador
";"

# Seleccionar solo las columnas numéricas
numeric_cols = df.select_dtypes(include=['float64']).columns
df_numeric = df[numeric_cols]
```

¿Qué hace esta parte?

- Importa las librerías necesarias (pandas, numpy, matplotlib, sklearn).
- Carga los datos desde country_resumido.csv.
- Filtra solo las columnas numéricas para hacer el clustering.

¿Qué variables se usan?

- Todas las variables numéricas en el dataset, como PIB per cápita, empleo, salud, facilidad de negocios, contaminación, etc.

2. Visualización de la Distribución de Variables

Código:

```
df_numeric.hist(figsize=(15, 12), bins=20, edgecolor="black")
plt.suptitle("📊 Distribución de Variables Numéricas", fontsize=16)
plt.show()
```

¿Qué hace?

- Genera histogramas para ver cómo están distribuidos los valores en cada variable.
- Nos ayuda a ver valores extremos y posibles sesgos en los datos.

¿Por qué es útil?

- Si algunas variables tienen una distribución muy desigual, puede afectar el clustering.

3. Normalización de los Datos

Código:

```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_numeric)
```

¿Qué hace?

- Convierte todas las variables a una escala similar.
- Esto es importante porque en clustering las variables con valores grandes podrían dominar el modelo.

Ejemplo:

- PIB per cápita puede estar en miles de dólares, mientras que tasa de desempleo está entre 0 y 100.
- StandardScaler convierte todo a la misma escala.

4. Determinar el Número Óptimo de Clusters

Código:

```
inertia = []
silhouette_scores = []
K_range = range(2, 11)  # Evaluamos de 2 a 10 clusters

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(df_scaled,
kmeans.labels_))
```

¿Qué hace?

- Prueba con diferentes valores de k (número de clusters).
- Calcula dos métricas para evaluar qué número de clusters es mejor:

1. Método del Codo (Inercia) → Mide qué tan compactos están los clusters.
2. Índice de Silueta → Mide qué tan bien separados están los clusters.

Gráfica para visualizarlo:

```
fig, ax1 = plt.subplots(figsize=(12, 5))

ax1.plot(K_range, inertia, 'bo-', label='Inercia (Método del Codo)')
ax1.set_xlabel('Número de Clusters (k)')
ax1.set_ylabel('Inercia')
ax1.set_title('📈 Método del Codo y Silueta para K-Means')
ax1.legend(loc='upper right')

ax2 = ax1.twinx()
ax2.plot(K_range, silhouette_scores, 'ro-', label='Índice de Silueta')
ax2.set_ylabel('Índice de Silueta')
ax2.legend(loc='lower right')

plt.show()
```

¿Qué mirar en la gráfica?

- Si hay un cambio de pendiente fuerte en el gráfico, ese es el mejor k.
- Si el índice de silueta es mayor a 0.3, el clustering es aceptable.

5. Aplicar K-Means con el Mejor Número de Clusters

Código:

```
best_k = 3  # Puedes cambiarlo según el gráfico del codo
kmeans = KMeans(n_clusters=best_k, random_state=42, n_init=10)
df["Cluster"] = kmeans.fit_predict(df_scaled)
```

¿Qué hace?

- Usa KMeans con el número de clusters óptimo (k=3).
- Agrupa los países en 3 clusters basándose en sus características económicas, de salud, empleo, etc.

6. Definir Preguntas para Clasificar Usuarios

Código:

```
preguntas = [
    "¿Qué tan importante es para ti la calidad del sistema de salud?",
    "¿Prefieres un país con alta empleabilidad y bajo desempleo?",
    "¿Es importante que el país tenga un crecimiento económico
positivo?",
    "¿Prefieres un país con alta facilidad para hacer negocios?",
```

```
    "¿Te interesa la estabilidad laboral en un país con más
asalariados?",
]
```

¿De dónde salieron estas preguntas?

- Fueron creadas basándonos en las variables clave del dataset.

7. Clasificación del Usuario

Código:

```
def clasificar_usuario(respuestas):
    respuestas_completas = df_numeric.mean().to_dict()  # Usamos
valores medios por defecto
    respuestas_completas.update(respuestas)

    usuario_vector = np.array([respuestas_completas[col] for col in
numeric_cols]).reshape(1, -1)
    usuario_vector_scaled = scaler.transform(usuario_vector)

    return kmeans.predict(usuario_vector_scaled)[0]
```

¿Qué hace?

- Recibe respuestas del usuario.
- Llena los valores faltantes con la media del dataset.
- Escala los valores igual que en el clustering.
- Asigna el usuario a un cluster con KMeans.

8. Cómo Ponerlo en una Aplicación Web

Código para la Web:

```
import streamlit as st

st.title("🌎 Recomendador de Países")

medicos = st.slider("Médicos por cada 1.000 personas", 0.0, 10.0, 5.0)
empleo = st.slider("Personas desempleadas educación avanzada", 0, 100,
20)

respuestas_usuario = {
    "Médicos (por cada 1.000 personas)": medicos,
    "Personas desempleadas educación avanzada": empleo,
}
```

```
if st.button("Obtener Recomendaciones"):
    cluster_usuario = clasificar_usuario(respuestas_usuario)
    st.write(f"✔ Estás en el Cluster {cluster_usuario}")
```

Para ejecutar:

```
streamlit run app.py
```