mas-alla-de-los-numeros-v0-91

September 26, 2024

1 El Mundo en Datos: Más Allá de los Números en la Comprensión Global

Este completo conjunto de datos proporciona abundante información sobre todos los países del mundo y abarca una amplia gama de indicadores y atributos. Abarca estadísticas demográficas, indicadores económicos, factores medioambientales, métricas sanitarias, estadísticas educativas y mucho más. Con todos los países representados, este conjunto de datos ofrece una perspectiva global completa sobre diversos aspectos de las naciones, lo que permite realizar análisis en profundidad y comparaciones entre países.

- **Cargar dataset en colab**

[ ]:
```
from google.colab import files

uploaded = files.upload()
csv_file = list(uploaded.keys())[0]
```

```
<IPython.core.display.HTML object>
```

```
Saving world-data-2023.csv to world-data-2023.csv
```

- **Cargar dataset con pandas**

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/world-data-2023.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
↪visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land( %)',
        'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Code',
        'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
```

```
       'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
       'Gasoline Price', 'GDP', 'Gross primary education enrollment (%)',
       'Gross tertiary education enrollment (%)', 'Infant mortality',
       'Largest city', 'Life expectancy', 'Maternal mortality ratio',
       'Minimum wage', 'Official language', 'Out of pocket health expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object'),
       Country Density\n(P/Km2) Abbreviation Agricultural Land( %)  \
0  Afghanistan               60           AF                58.10%
1      Albania              105           AL                43.10%
2      Algeria               18           DZ                17.40%
3      Andorra              164           AD                40.00%
4       Angola               26           AO                47.50%

  Land Area(Km2) Armed Forces size  Birth Rate  Calling Code  \
0        652,230           323,000       32.49          93.0
1         28,748             9,000       11.78         355.0
2      2,381,741           317,000       24.28         213.0
3            468               NaN        7.20         376.0
4      1,246,700           117,000       40.73         244.0

  Capital/Major City Co2-Emissions  … Out of pocket health expenditure  \
0              Kabul         8,672  …                           78.40%
1             Tirana         4,536  …                           56.90%
2            Algiers       150,006  …                           28.10%
3   Andorra la Vella           469  …                           36.40%
4             Luanda        34,693  …                           33.40%

  Physicians per thousand  Population  \
0                    0.28  38,041,754
1                    1.20   2,854,191
2                    1.72  43,053,054
3                    3.33      77,142
4                    0.21  31,825,295

   Population: Labor force participation (%) Tax revenue (%) Total tax rate  \
0                                     48.90%           9.30%         71.40%
1                                     55.70%          18.60%         36.60%
2                                     41.20%          37.20%         66.10%
3                                        NaN             NaN            NaN
4                                     77.50%           9.20%         49.10%

  Unemployment rate Urban_population   Latitude  Longitude
0            11.12%        9,797,273  33.939110  67.709953
```

```
1            12.33%        1,747,593  41.153332  20.168331
2            11.70%       31,510,100  28.033886   1.659626
3               NaN           67,873  42.506285   1.521801
4             6.89%       21,061,025 -11.202692  17.873887

[5 rows x 35 columns])
```

*El dataset contiene **35 columnas con diferentes atributos para cada país**. Aquí están las columnas originales:*

1. Country
2. Density (P/Km2)
3. Abbreviation
4. Agricultural Land (%)
5. Land Area (Km2)
6. Armed Forces Size
7. Birth Rate
8. Calling Code
9. Capital/Major City
10. CO2 Emissions
11. CPI
12. CPI Change (%)
13. Currency Code
14. Fertility Rate
15. Forested Area (%)
16. Gasoline Price
17. GDP
18. Gross Primary Education Enrollment (%)
19. Gross Tertiary Education Enrollment (%)
20. Largest City
21. Life Expectancy
22. Maternal Mortality Ratio
23. Minimum Wage
24. Official Language
25. Out of Pocket Health Expenditure (%)
26. Physicians per Thousand
27. Population
28. Labor Force Participation (%)
29. Tax Revenue (%)
30. Total Tax Rate
31. Unemployment Rate
32. Urban Population
33. Latitude
34. Longitude

- **Traducir cada columna al español.**

[ ]:
```
# Diccionario para traducir los nombres de las columnas al español
column_translation = {
    'Country': 'País',
    'Density\n(P/Km2)': 'Densidad (P/Km2)',
    'Abbreviation': 'Abreviación',
    'Agricultural Land( %)': 'Terreno Agrícola (%)',
    'Land Area(Km2)': 'Área de Tierra (Km2)',
    'Armed Forces size': 'Tamaño de las Fuerzas Armadas',
    'Birth Rate': 'Tasa de Natalidad',
    'Calling Code': 'Código Telefónico',
    'Capital/Major City': 'Capital/Ciudad Principal',
    'Co2-Emissions': 'Emisiones de CO2',
    'CPI': 'Índice de Precios al Consumidor (IPC)',
    'CPI Change (%)': 'Cambio del IPC (%)',
    'Currency-Code': 'Código de Moneda',
    'Fertility Rate': 'Tasa de Fertilidad',
    'Forested Area (%)': 'Área Boscosa (%)',
    'Gasoline Price': 'Precio de la Gasolina',
    'GDP': 'Producto Interno Bruto (PIB)',
    'Gross primary education enrollment (%)': 'Inscripción Bruta en Educación␣
 ↪Primaria (%)',
    'Gross tertiary education enrollment (%)': 'Inscripción Bruta en Educación␣
 ↪Terciaria (%)',
    'Infant mortality': 'Mortalidad Infantil',
    'Largest city': 'Ciudad Más Grande',
    'Life expectancy': 'Esperanza de Vida',
    'Maternal mortality ratio': 'Tasa de Mortalidad Materna',
    'Minimum wage': 'Salario Mínimo',
    'Official language': 'Idioma Oficial',
    'Out of pocket health expenditure': 'Gastos de Salud de Bolsillo (%)',
    'Physicians per thousand': 'Médicos por Mil Habitantes',
    'Population': 'Población',
    'Population: Labor force participation (%)': 'Participación en la Fuerza␣
 ↪Laboral (%)',
    'Tax revenue (%)': 'Ingresos Fiscales (%)',
    'Total tax rate': 'Tasa Impositiva Total',
    'Unemployment rate': 'Tasa de Desempleo',
    'Urban_population': 'Población Urbana',
    'Latitude': 'Latitud',
    'Longitude': 'Longitud'
}

# Renombrar las columnas del DataFrame
df.rename(columns=column_translation, inplace=True)

# Mostrar las primeras filas del DataFrame con las columnas traducidas
df_head_translated = df.head()
```

```
df_head_translated
```

[ ]:

	País	Densidad (P/Km2)	Abreviación	Terreno Agrícola (%)	\
0	Afghanistan	60	AF	58.10%	
1	Albania	105	AL	43.10%	
2	Algeria	18	DZ	17.40%	
3	Andorra	164	AD	40.00%	
4	Angola	26	AO	47.50%	

	Área de Tierra (Km2)	Tamaño de las Fuerzas Armadas	Tasa de Natalidad	\
0	652,230	323,000	32.49	
1	28,748	9,000	11.78	
2	2,381,741	317,000	24.28	
3	468	NaN	7.20	
4	1,246,700	117,000	40.73	

	Código Telefónico	Capital/Ciudad Principal	Emisiones de CO2	…	\
0	93.0	Kabul	8,672	…	
1	355.0	Tirana	4,536	…	
2	213.0	Algiers	150,006	…	
3	376.0	Andorra la Vella	469	…	
4	244.0	Luanda	34,693	…	

	Gastos de Salud de Bolsillo (%)	Médicos por Mil Habitantes	Población	\
0	78.40%	0.28	38,041,754	
1	56.90%	1.20	2,854,191	
2	28.10%	1.72	43,053,054	
3	36.40%	3.33	77,142	
4	33.40%	0.21	31,825,295	

	Participación en la Fuerza Laboral (%)	Ingresos Fiscales (%)	\
0	48.90%	9.30%	
1	55.70%	18.60%	
2	41.20%	37.20%	
3	NaN	NaN	
4	77.50%	9.20%	

	Tasa Impositiva Total	Tasa de Desempleo	Población Urbana	Latitud	\
0	71.40%	11.12%	9,797,273	33.939110	
1	36.60%	12.33%	1,747,593	41.153332	
2	66.10%	11.70%	31,510,100	28.033886	
3	NaN	NaN	67,873	42.506285	
4	49.10%	6.89%	21,061,025	-11.202692	

	Longitud
0	67.709953
1	20.168331

```
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns]
```

- Las columnas del dataset han sido traducidas al español. **Ahora, el siguiente paso es inspeccionar los tipos de datos y obtener más información sobre el dataset** usando el método df.info(). Vamos a continuar con ese análisis.

[ ]:
```
# Guardar el DataFrame traducido en un archivo CSV para descarga
cleaned_file_path_final = '/content/sample_data/
 ↪world-data-2023-traduccion-final.csv'
df.to_csv(cleaned_file_path_final, index=False)
```

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-traduccion-final.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
 ↪visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
          País Densidad (P/Km2) Abreviación Terreno Agrícola (%)  \
0  Afghanistan              60          AF               58.10%
1      Albania             105          AL               43.10%
```

```
2      Algeria               18          DZ               17.40%
3      Andorra              164          AD               40.00%
4       Angola               26          AO               47.50%

  Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652,230                       323,000              32.49
1               28,748                         9,000              11.78
2            2,381,741                       317,000              24.28
3                  468                           NaN               7.20
4            1,246,700                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal Emisiones de CO2  …  \
0               93.0                    Kabul            8,672  …
1              355.0                   Tirana            4,536  …
2              213.0                  Algiers          150,006  …
3              376.0        Andorra la Vella              469  …
4              244.0                   Luanda           34,693  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                          78.40%                       0.28  38,041,754
1                          56.90%                       1.20   2,854,191
2                          28.10%                       1.72  43,053,054
3                          36.40%                       3.33      77,142
4                          33.40%                       0.21  31,825,295

   Participación en la Fuerza Laboral (%) Ingresos Fiscales (%)  \
0                                  48.90%                 9.30%
1                                  55.70%                18.60%
2                                  41.20%                37.20%
3                                     NaN                   NaN
4                                  77.50%                 9.20%

  Tasa Impositiva Total Tasa de Desempleo Población Urbana    Latitud  \
0                71.40%            11.12%        9,797,273  33.939110
1                36.60%            12.33%        1,747,593  41.153332
2                66.10%            11.70%       31,510,100  28.033886
3                   NaN               NaN           67,873  42.506285
4                49.10%             6.89%       21,061,025 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

```
[ ]: # Obtener información detallada del dataframe, incluyendo el tipo de datos
     df_info = df.info()
     df_info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 35 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   País                                      195 non-null    object
 1   Densidad (P/Km2)                          195 non-null    object
 2   Abreviación                               188 non-null    object
 3   Terreno Agrícola (%)                      188 non-null    object
 4   Área de Tierra (Km2)                      194 non-null    object
 5   Tamaño de las Fuerzas Armadas             171 non-null    object
 6   Tasa de Natalidad                         189 non-null    float64
 7   Código Telefónico                         194 non-null    float64
 8   Capital/Ciudad Principal                  192 non-null    object
 9   Emisiones de CO2                          188 non-null    object
 10  Índice de Precios al Consumidor (IPC)     178 non-null    object
 11  Cambio del IPC (%)                        179 non-null    object
 12  Código de Moneda                          180 non-null    object
 13  Tasa de Fertilidad                        188 non-null    float64
 14  Área Boscosa (%)                          188 non-null    object
 15  Precio de la Gasolina                     175 non-null    object
 16  Producto Interno Bruto (PIB)              193 non-null    object
 17  Inscripción Bruta en Educación Primaria (%)  188 non-null  object
 18  Inscripción Bruta en Educación Terciaria (%)  183 non-null  object
 19  Mortalidad Infantil                       189 non-null    float64
 20  Ciudad Más Grande                         189 non-null    object
 21  Esperanza de Vida                         187 non-null    float64
 22  Tasa de Mortalidad Materna                181 non-null    float64
 23  Salario Mínimo                            150 non-null    object
 24  Idioma Oficial                            190 non-null    object
 25  Gastos de Salud de Bolsillo (%)           188 non-null    object
 26  Médicos por Mil Habitantes                188 non-null    float64
 27  Población                                 194 non-null    object
 28  Participación en la Fuerza Laboral (%)    176 non-null    object
 29  Ingresos Fiscales (%)                     169 non-null    object
 30  Tasa Impositiva Total                     183 non-null    object
 31  Tasa de Desempleo                         176 non-null    object
 32  Población Urbana                          190 non-null    object
 33  Latitud                                   194 non-null    float64
 34  Longitud                                  194 non-null    float64
dtypes: float64(9), object(26)
memory usage: 53.4+ KB
```

- **Los tipos de datos se distribuyen de la siguiente manera:**

9 columnas son de tipo float64: principalmente indicadores numéricos como tasas, índices, etc.

26 columnas son de tipo object: estas incluyen cadenas de texto, porcentajes representados como texto, y algunos valores numéricos que están formateados como texto.

- Vamos a **separar las columnas por tipo de dato** (float y object) para tener una visión más clara y proceder con el chequeo de la calidad de los datos, como los valores duplicados y faltantes.

[ ]:
```
# Separar columnas por tipo de dato
float_columns = df.select_dtypes(include=['float64']).columns.tolist()
object_columns = df.select_dtypes(include=['object']).columns.tolist()

# Mostrar las columnas de cada tipo
float_columns, object_columns
```

[ ]:
```
(['Tasa de Natalidad',
  'Código Telefónico',
  'Tasa de Fertilidad',
  'Mortalidad Infantil',
  'Esperanza de Vida',
  'Tasa de Mortalidad Materna',
  'Médicos por Mil Habitantes',
  'Latitud',
  'Longitud'],
 ['País',
  'Densidad (P/Km2)',
  'Abreviación',
  'Terreno Agrícola (%)',
  'Área de Tierra (Km2)',
  'Tamaño de las Fuerzas Armadas',
  'Capital/Ciudad Principal',
  'Emisiones de CO2',
  'Índice de Precios al Consumidor (IPC)',
  'Cambio del IPC (%)',
  'Código de Moneda',
  'Área Boscosa (%)',
  'Precio de la Gasolina',
  'Producto Interno Bruto (PIB)',
  'Inscripción Bruta en Educación Primaria (%)',
  'Inscripción Bruta en Educación Terciaria (%)',
  'Ciudad Más Grande',
  'Salario Mínimo',
  'Idioma Oficial',
  'Gastos de Salud de Bolsillo (%)',
  'Población',
  'Participación en la Fuerza Laboral (%)',
  'Ingresos Fiscales (%)',
```

```
'Tasa Impositiva Total',
'Tasa de Desempleo',
'Población Urbana'])
```

- **Verificar la calidad de los datos comprobando si hay valores duplicados o faltantes.** También evaluaremos la mejor manera de manejar los valores faltantes para preservar la integridad del dataset.

[ ]:
```
# Comprobación de valores duplicados
duplicated_rows = df.duplicated().sum()

# Comprobación de valores faltantes por columna
missing_values = df.isnull().sum()

duplicated_rows, missing_values
```

[ ]:
```
(0,
 País                                          0
 Densidad (P/Km2)                              0
 Abreviación                                   7
 Terreno Agrícola (%)                          7
 Área de Tierra (Km2)                          1
 Tamaño de las Fuerzas Armadas                24
 Tasa de Natalidad                             6
 Código Telefónico                             1
 Capital/Ciudad Principal                      3
 Emisiones de CO2                              7
 Índice de Precios al Consumidor (IPC)        17
 Cambio del IPC (%)                           16
 Código de Moneda                             15
 Tasa de Fertilidad                            7
 Área Boscosa (%)                              7
 Precio de la Gasolina                        20
 Producto Interno Bruto (PIB)                  2
 Inscripción Bruta en Educación Primaria (%)   7
 Inscripción Bruta en Educación Terciaria (%) 12
 Mortalidad Infantil                           6
 Ciudad Más Grande                             6
 Esperanza de Vida                             8
 Tasa de Mortalidad Materna                   14
 Salario Mínimo                               45
 Idioma Oficial                                5
 Gastos de Salud de Bolsillo (%)               7
 Médicos por Mil Habitantes                    7
 Población                                     1
 Participación en la Fuerza Laboral (%)       19
 Ingresos Fiscales (%)                        26
```

```
Tasa Impositiva Total                           12
Tasa de Desempleo                               19
Población Urbana                                 5
Latitud                                          1
Longitud                                         1
dtype: int64)
```

No se encontraron filas duplicadas en el dataset. **Sin embargo, hay varias columnas con valores faltantes**. Algunas de las columnas con la mayor cantidad de valores faltantes son:

- Salario Mínimo: 45 valores faltantes
- Ingresos Fiscales (%): 26 valores faltantes
- Precio de la Gasolina: 20 valores faltantes
- Participación en la Fuerza Laboral (%): 19 valores faltantes

- **Próximo paso: Rellenar los valores faltantes**
  Pero antes vamos a cambiar la **codificación**

[ ]:
```
import pandas as pd
# Cambia 'utf-8' por la codificación correcta
df = pd.read_csv('/content/sample_data/world-data-2023-traduccion-final.csv',␣
 ↪encoding='utf-8')
```

- **Cargar** nuevamente el dataset

[ ]:
```
import chardet

with open('/content/sample_data/world-data-2023-traduccion-final.csv', 'rb') as␣
 ↪f:
    resultado = chardet.detect(f.read())
    print(resultado)
```

```
{'encoding': 'utf-8', 'confidence': 0.99, 'language': ''}
```

- Vamos a **rellenar los valores** de salario mínimo con los datos correctos mensuales. *Busqueda por internet*

[ ]:
```
# Vamos a rellenar los valores de salario mínimo con los datos correctos␣
 ↪mensuales
df.loc[df['País'] == 'Argentina', 'Salario Mínimo'] = '100-130 USD mensuales'
df.loc[df['País'] == 'Brasil', 'Salario Mínimo'] = '290 USD mensuales'
df.loc[df['País'] == 'México', 'Salario Mínimo'] = '265 USD mensuales'

# Revisamos los datos actualizados
df[['País', 'Salario Mínimo']].head()
```

[ ]:
```
          País Salario Mínimo
0  Afghanistan         $0.43
```

```
1       Albania          $1.12
2       Algeria          $0.95
3       Andorra          $6.63
4        Angola          $0.71
```

[ ]:
```
# Vamos a rellenar los valores de emisiones de CO2 con los datos encontrados␣
 ↪para Brasil y México
df.loc[df['País'] == 'Brasil', 'Emisiones de CO2'] = '466.77 kt'
df.loc[df['País'] == 'México', 'Emisiones de CO2'] = '487.77 kt'

# Revisamos los datos actualizados
df[['País', 'Emisiones de CO2']].head()
```

[ ]:
```
          País Emisiones de CO2
0  Afghanistan           8,672
1      Albania           4,536
2      Algeria         150,006
3      Andorra             469
4       Angola          34,693
```

- Pude observar un **caracter especial** en un pais = (Lo vi cuando hice un chequeo en excel)

[ ]:
```
valor = df.loc[150, 'País']
print(valor)
```

```
S
```

[ ]:
```
# Rellenamos los datos en el dataset con la información obtenida

# Corregir nombre de pais
df.loc[df['País'] == df.loc[150, 'País'], 'País'] = 'Santo Tomé y Príncipe'
# Abreviación (ISO)
df.loc[df['País'] == 'Brasil', 'Abreviación'] = 'BR'
df.loc[df['País'] == 'México', 'Abreviación'] = 'MX'
df.loc[df['País'] == 'Argentina', 'Abreviación'] = 'AR'
df.loc[df['País'] == 'Eswatini', 'Abreviación'] = 'SZ'
df.loc[df['País'] == 'Gabon', 'Abreviación'] = 'GA'
df.loc[df['País'] == 'Guam', 'Abreviación'] = 'GU'
df.loc[df['País'] == 'Jersey', 'Abreviación'] = 'JE'
df.loc[df['País'] == 'Kazakhstan', 'Abreviación'] = 'KZ'
df.loc[df['País'] == 'Micronesia', 'Abreviación'] = 'FM'
df.loc[df['País'] == 'Tuvalu', 'Abreviación'] = 'TV'

# Código Telefónico
df.loc[df['País'] == 'Brasil', 'Código Telefónico'] = '+55'
df.loc[df['País'] == 'México', 'Código Telefónico'] = '+52'
df.loc[df['País'] == 'Argentina', 'Código Telefónico'] = '+54'
```

```
df.loc[df['País'] == 'Botsuana', 'Código Telefónico'] = '+267'
df.loc[df['País'] == 'Bosnia y Herzegovina', 'Código Telefónico'] = '+387'
df.loc[df['País'] == 'Cabo Verde', 'Código Telefónico'] = '+238'
df.loc[df['País'] == 'Chad', 'Código Telefónico'] = '+235'
df.loc[df['País'] == 'Comoras', 'Código Telefónico'] = '+269'
df.loc[df['País'] == 'Micronesia', 'Código Telefónico'] = '+691'
df.loc[df['País'] == 'Tuvalu', 'Código Telefónico'] = '+688'

# Capital/Ciudad Principal
df.loc[df['País'] == 'Brasil', 'Capital/Ciudad Principal'] = 'Brasília'
df.loc[df['País'] == 'México', 'Capital/Ciudad Principal'] = 'Ciudad de México'
df.loc[df['País'] == 'Argentina', 'Capital/Ciudad Principal'] = 'Buenos Aires'
df.loc[df['País'] == 'Micronesia', 'Capital/Ciudad Principal'] = 'Palikir'
df.loc[df['País'] == 'Montserrat', 'Capital/Ciudad Principal'] = 'Plymouth␣
 ↪(oficial), Brades (de facto)'
df.loc[df['País'] == 'San Vicente y las Granadinas', 'Capital/Ciudad␣
 ↪Principal'] = 'Kingstown'
df.loc[df['País'] == 'Comoras', 'Capital/Ciudad Principal'] = 'Moroni'
df.loc[df['País'] == 'Cabo Verde', 'Capital/Ciudad Principal'] = 'Praia'
df.loc[df['País'] == 'Bosnia y Herzegovina', 'Capital/Ciudad Principal'] =␣
 ↪'Sarajevo'
df.loc[df['País'] == 'Chad', 'Capital/Ciudad Principal'] = 'N\'Djamena'
df.loc[df['País'] == 'Botsuana', 'Capital/Ciudad Principal'] = 'Gaborone'
df.loc[df['País'] == 'Burundi', 'Capital/Ciudad Principal'] = 'Bujumbura'

# Código de Moneda
df.loc[df['País'] == 'Brasil', 'Código de Moneda'] = 'BRL'
df.loc[df['País'] == 'México', 'Código de Moneda'] = 'MXN'
df.loc[df['País'] == 'Argentina', 'Código de Moneda'] = 'ARS'
df.loc[df['País'] == 'Eswatini', 'Código de Moneda'] = 'SZL'
df.loc[df['País'] == 'Gabon', 'Código de Moneda'] = 'XAF'
df.loc[df['País'] == 'Guam', 'Código de Moneda'] = 'USD'
df.loc[df['País'] == 'Jersey', 'Código de Moneda'] = 'GBP'
df.loc[df['País'] == 'Kazakhstan', 'Código de Moneda'] = 'KZT'
df.loc[df['País'] == 'Micronesia', 'Código de Moneda'] = 'USD'
df.loc[df['País'] == 'Tuvalu', 'Código de Moneda'] = 'AUD'

# Ciudad Más Grande
df.loc[df['País'] == 'Brasil', 'Ciudad Más Grande'] = 'São Paulo'
df.loc[df['País'] == 'México', 'Ciudad Más Grande'] = 'Ciudad de México'
df.loc[df['País'] == 'Argentina', 'Ciudad Más Grande'] = 'Buenos Aires'
df.loc[df['País'] == 'Micronesia', 'Ciudad Más Grande'] = 'Weno'
df.loc[df['País'] == 'Montserrat', 'Ciudad Más Grande'] = 'Brades'
df.loc[df['País'] == 'San Vicente y las Granadinas', 'Ciudad Más Grande'] =␣
 ↪'Kingstown'
df.loc[df['País'] == 'Burundi', 'Ciudad Más Grande'] = 'Bujumbura'
df.loc[df['País'] == 'Bosnia y Herzegovina', 'Ciudad Más Grande'] = 'Sarajevo'
```

```
df.loc[df['País'] == 'Cabo Verde', 'Ciudad Más Grande'] = 'Praia'
df.loc[df['País'] == 'Comoras', 'Ciudad Más Grande'] = 'Moroni'
df.loc[df['País'] == 'Chad', 'Ciudad Más Grande'] = 'N\'Djamena'
df.loc[df['País'] == 'Botsuana', 'Ciudad Más Grande'] = 'Gaborone'

# Población
df.loc[df['País'] == 'Brasil', 'Población'] = '215.3 millones'
df.loc[df['País'] == 'México', 'Población'] = '127.5 millones'
df.loc[df['País'] == 'Argentina', 'Población'] = '46.2 millones'
df.loc[df['País'] == 'Tuvalu', 'Población'] = '11,792'
df.loc[df['País'] == 'Chad', 'Población'] = '17.18 millones'


# Latitud y Longitud
df.loc[df['País'] == 'Brasil', 'Latitud'] = '-15.793889'
df.loc[df['País'] == 'Brasil', 'Longitud'] = '-47.882778'
df.loc[df['País'] == 'México', 'Latitud'] = '19.432608'
df.loc[df['País'] == 'México', 'Longitud'] = '-99.133209'
df.loc[df['País'] == 'Argentina', 'Latitud'] = '-34.603722'
df.loc[df['País'] == 'Argentina', 'Longitud'] = '-58.381592'
df.loc[df['País'] == 'Tuvalu', 'Latitud'] = '-7.1095'
df.loc[df['País'] == 'Tuvalu', 'Longitud'] = '179.1940'
df.loc[df['País'] == 'Chad', 'Latitud'] = '12.1348'
df.loc[df['País'] == 'Chad', 'Longitud'] = '15.0557'

# Idioma Oficial
df.loc[df['País'] == 'Bosnia y Herzegovina', 'Idioma Oficial'] = 'Bosnio,␣
 ↪Croata, Serbio'
df.loc[df['País'] == 'Cabo Verde', 'Idioma Oficial'] = 'Portugués'
df.loc[df['País'] == 'Chad', 'Idioma Oficial'] = 'Francés'
df.loc[df['País'] == 'Comoras', 'Idioma Oficial'] = 'Inglés'
df.loc[df['País'] == 'Guam', 'Idioma Oficial'] = 'Inglés'
df.loc[df['País'] == 'Jersey', 'Idioma Oficial'] = 'Inglés'
df.loc[df['País'] == 'Kazakhstan', 'Idioma Oficial'] = 'Kazakh'
df.loc[df['País'] == 'Micronesia', 'Idioma Oficial'] = 'Inglés'
df.loc[df['País'] == 'Tuvalu', 'Idioma Oficial'] = 'Inglés'

# Verificamos que los datos se hayan actualizado correctamente
df[['País', 'Abreviación', 'Código Telefónico', 'Capital/Ciudad Principal',␣
 ↪'Código de Moneda',
                    'Ciudad Más Grande', 'Población', 'Latitud', 'Longitud',␣
 ↪'Idioma Oficial']].head()
```

```
<ipython-input-14-9365bbd9aa7a>:18: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise in a future error of pandas.
Value '+55' has dtype incompatible with float64, please explicitly cast to a
compatible dtype first.
```

```
  df.loc[df['País'] == 'Brasil', 'Código Telefónico'] = '+55'
<ipython-input-14-9365bbd9aa7a>:78: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise in a future error of pandas.
Value '-15.793889' has dtype incompatible with float64, please explicitly cast
to a compatible dtype first.
  df.loc[df['País'] == 'Brasil', 'Latitud'] = '-15.793889'
<ipython-input-14-9365bbd9aa7a>:79: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise in a future error of pandas.
Value '-47.882778' has dtype incompatible with float64, please explicitly cast
to a compatible dtype first.
  df.loc[df['País'] == 'Brasil', 'Longitud'] = '-47.882778'
```

[ ]:

	País	Abreviación	Código Telefónico	Capital/Ciudad Principal	Código de Moneda	Ciudad Más Grande	Población	Latitud	Longitud	Idioma Oficial
0	Afghanistan	AF	93.0	Kabul	AFN	Kabul	38,041,754	33.93911	67.709953	Pashto
1	Albania	AL	355.0	Tirana	ALL	Tirana	2,854,191	41.153332	20.168331	Albanian
2	Algeria	DZ	213.0	Algiers	DZD	Algiers	43,053,054	28.033886	1.659626	Arabic
3	Andorra	AD	376.0	Andorra la Vella	EUR	Andorra la Vella	77,142	42.506285	1.521801	Catalan
4	Angola	AO	244.0	Luanda	AOA	Luanda	31,825,295	-11.202692	17.873887	Portuguese

- **Chequeamos** ese pais con ese caracter especial para ver si, efectivamente esta todo bien

[ ]:
```
valor = df.loc[150, 'País']
print(valor)
```

```
Santo Tomé y Príncipe
```

- Vamos a **guardar** esto para hacer un **chequeo manual** en excel

[ ]:
```
# Guardar el DataFrame limpio y sin símbolos en un archivo CSV para descarga
cleaned_file_path_final = '/content/sample_data/world-data-2023-cleaned-1-1.csv'
df.to_csv(cleaned_file_path_final, index=False)

cleaned_file_path_final, df.head()
```

[ ]: ('/content/sample_data/world-data-2023-cleaned-1-1.csv',

	País	Densidad (P/Km2)	Abreviación	Terreno Agrícola (%)	\
0	Afghanistan	60	AF	58.10%	
1	Albania	105	AL	43.10%	
2	Algeria	18	DZ	17.40%	
3	Andorra	164	AD	40.00%	
4	Angola	26	AO	47.50%	

	Área de Tierra (Km2)	Tamaño de las Fuerzas Armadas	Tasa de Natalidad	\
0	652,230	323,000	32.49	
1	28,748	9,000	11.78	
2	2,381,741	317,000	24.28	
3	468	NaN	7.20	
4	1,246,700	117,000	40.73	

	Código Telefónico	Capital/Ciudad Principal	Emisiones de CO2	…	\
0	93.0	Kabul	8,672	…	
1	355.0	Tirana	4,536	…	
2	213.0	Algiers	150,006	…	
3	376.0	Andorra la Vella	469	…	
4	244.0	Luanda	34,693	…	

	Gastos de Salud de Bolsillo (%)	Médicos por Mil Habitantes	Población	\
0	78.40%	0.28	38,041,754	
1	56.90%	1.20	2,854,191	
2	28.10%	1.72	43,053,054	
3	36.40%	3.33	77,142	
4	33.40%	0.21	31,825,295	

	Participación en la Fuerza Laboral (%)	Ingresos Fiscales (%)	\
0	48.90%	9.30%	
1	55.70%	18.60%	
2	41.20%	37.20%	
3	NaN	NaN	
4	77.50%	9.20%	

	Tasa Impositiva Total	Tasa de Desempleo	Población Urbana	Latitud	\
0	71.40%	11.12%	9,797,273	33.93911	
1	36.60%	12.33%	1,747,593	41.153332	
2	66.10%	11.70%	31,510,100	28.033886	
3	NaN	NaN	67,873	42.506285	
4	49.10%	6.89%	21,061,025	-11.202692	

	Longitud
0	67.709953
1	20.168331
2	1.659626

```
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- Aun encontramos **caracteres especiales**. Tal vez sea porque el en idioma español tenemos acentos y esto puede perjudicar, vamos a corregir

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-cleaned-1-1.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
 ↪visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
          País Densidad (P/Km2) Abreviación Terreno Agrícola (%)  \
0  Afghanistan              60          AF               58.10%
1      Albania             105          AL               43.10%
2      Algeria              18          DZ               17.40%
3      Andorra             164          AD               40.00%
4       Angola              26          AO               47.50%

  Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652,230                       323,000              32.49
1               28,748                         9,000              11.78
```

```
2            2,381,741                       317,000              24.28
3                  468                           NaN               7.20
4            1,246,700                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal Emisiones de CO2  …  \
0               93.0                     Kabul           8,672  …
1              355.0                    Tirana           4,536  …
2              213.0                   Algiers         150,006  …
3              376.0           Andorra la Vella             469  …
4              244.0                    Luanda          34,693  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                          78.40%                       0.28  38,041,754
1                          56.90%                       1.20   2,854,191
2                          28.10%                       1.72  43,053,054
3                          36.40%                       3.33      77,142
4                          33.40%                       0.21  31,825,295

   Participación en la Fuerza Laboral (%) Ingresos Fiscales (%)  \
0                                  48.90%                 9.30%
1                                  55.70%                18.60%
2                                  41.20%                37.20%
3                                     NaN                   NaN
4                                  77.50%                 9.20%

  Tasa Impositiva Total Tasa de Desempleo Población Urbana    Latitud  \
0                71.40%            11.12%        9,797,273  33.939110
1                36.60%            12.33%        1,747,593  41.153332
2                66.10%            11.70%       31,510,100  28.033886
3                   NaN               NaN           67,873  42.506285
4                49.10%             6.89%       21,061,025 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- **Caracteres corruptos**

[ ]:
```
# Revisar el dataset en busca de caracteres corruptos como "¿½i", "Ã", etc.
import numpy as np

# Identificar filas que contienen caracteres especiales problemáticos
```

```
problematic_rows = df.apply(lambda x: x.astype(str).str.
 ↪contains(r'[^\x00-\x7F]', na=False)).any(axis=1)

# Mostrar las filas problemáticas para revisión
df_problematic = df[problematic_rows]

df_problematic
```

[ ]:

	País	Densidad (P/Km2)	Abreviación	Terreno Agrícola (%)	\
23	Brazil	25	BR	33.90%	
31	Cameroon	56	CM	20.60%	
34	Chad	13	TD	39.70%	
37	Colombia	46	CO	40.30%	
40	Costa Rica	100	CR	34.50%	
43	Cyprus	131	CY	12.20%	
76	Iceland	3	IS	18.70%	
104	Maldives	1,802	MV	26.30%	
112	Moldova	123	MD	74.20%	
136	Paraguay	18	PY	55.10%	
150	Santo Tomé y Príncipe	228	ST	50.70%	
168	Sweden	25	SE	7.40%	
169	Switzerland	219	CH	38.40%	
175	Togo	152	TG	70.20%	
176	Tonga	147	TO	45.80%	
181	Tuvalu	393	TV	60.00%	

	Área de Tierra (Km2)	Tamaño de las Fuerzas Armadas	Tasa de Natalidad	\
23	8,515,770	730,000	13.92	
31	475,440	24,000	35.39	
34	1,284,000	35,000	42.17	
37	1,138,910	481,000	14.88	
40	51,100	10,000	13.97	
43	9,251	16,000	10.46	
76	103,000	0	12.00	
104	298	5,000	14.20	
112	33,851	7,000	10.10	
136	406,752	27,000	20.57	
150	964	1,000	31.54	
168	450,295	30,000	11.40	
169	41,277	21,000	10.00	
175	56,785	10,000	33.11	
176	747	NaN	24.30	
181	26	NaN	NaN	

	Código Telefónico	Capital/Ciudad Principal	Emisiones de CO2	…	\
23	55.0	Bras	462,299	…	
31	237.0	Yaound	8,291	…	

34	235.0	N'Djamena	1,016	…
37	57.0	Bogot	97,814	…
40	506.0	San Jos	8,023	…
43	357.0	Nicosia	6,626	…
76	354.0	Reykjav	2,065	…
104	960.0	Mal	1,445	…
112	373.0	Chi	5,115	…
136	595.0	Asunci	7,407	…
150	239.0	S	121	…
168	46.0	Stockholm	43,252	…
169	41.0	Bern	34,477	…
175	228.0	Lom	3,000	…
176	676.0	Nuku	128	…
181	688.0	Funafuti	11	…

	Gastos de Salud de Bolsillo (%)	Médicos por Mil Habitantes	\
23	28.30%	2.15	
31	69.70%	0.09	
34	56.40%	0.04	
37	18.30%	2.18	
40	21.50%	2.89	
43	43.90%	1.95	
76	17.00%	4.08	
104	16.40%	4.56	
112	46.20%	3.21	
136	36.50%	1.35	
150	11.70%	0.05	
168	15.20%	3.98	
169	28.30%	4.30	
175	51.00%	0.08	
176	10.20%	0.52	
181	0.70%	0.92	

	Población	Participación en la Fuerza Laboral (%)	\
23	212,559,417	63.90%	
31	25,876,380	76.10%	
34	17.18 millones	70.70%	
37	50,339,443	68.80%	
40	5,047,561	62.10%	
43	1,198,575	63.10%	
76	361,313	75.00%	
104	530,953	69.80%	
112	2,657,637	43.10%	
136	7,044,636	72.10%	
150	215,056	57.80%	
168	10,285,453	64.60%	
169	8,574,832	68.30%	

```
175       8,082,366                                    77.60%
176         100,209                                    59.80%
181          11,792                                       NaN

    Ingresos Fiscales (%) Tasa Impositiva Total Tasa de Desempleo  \
23                 14.20%                 65.10%             12.08%
31                 12.80%                 57.70%              3.38%
34                    NaN                 63.50%              1.89%
37                 14.40%                 71.20%              9.71%
40                 13.60%                 58.30%             11.85%
43                 24.50%                 22.40%              7.27%
76                 23.30%                 31.90%              2.84%
104                19.50%                 30.20%              6.14%
112                17.70%                 38.70%              5.47%
136                10.00%                 35.00%              4.81%
150                14.60%                 37.00%             13.37%
168                27.90%                 49.10%              6.48%
169                10.10%                 28.80%              4.58%
175                16.90%                 48.20%              2.04%
176                22.30%                 27.50%              1.12%
181                   NaN                    NaN                NaN

    Población Urbana    Latitud    Longitud
23       183,241,641 -14.235004  -51.925280
31        14,741,256   7.369722   12.354722
34         3,712,273  12.134800   15.055700
37        40,827,302   4.570868  -74.297333
40         4,041,885   9.748917  -83.753428
43           800,708  35.126413   33.429859
76           339,110  64.963051  -19.020835
104          213,645   3.202778   73.220680
112        1,135,502  47.411631   28.369885
136        4,359,150 -23.442503  -58.443832
150          158,277        NaN         NaN
168        9,021,165  60.128161   18.643501
169        6,332,428  46.818188    8.227512
175        3,414,638   8.619543    0.824782
176           24,145 -21.178986 -175.198242
181            7,362  -7.109500  179.194000

[16 rows x 35 columns]
```

- Existen **caracteres especiales en capitales** ( ) pero como no voy a usar estas columnas las dejamos como estan

[ ]:
```
# Reemplazar caracteres corruptos en las columnas afectadas
```

```
df_updated_cleaned = df.apply(lambda x: x.str.encode('ascii', 'ignore').str.
 ↪decode('utf-8') if x.dtype == "object" else x)

# Verificamos nuevamente si quedan filas con caracteres corruptos
problematic_rows_cleaned = df_updated_cleaned.apply(lambda x: x.astype(str).str.
 ↪contains(r'[^\x00-\x7F]', na=False)).any(axis=1)

# Filas problemáticas después de la limpieza
df_problematic_cleaned = df_updated_cleaned[problematic_rows_cleaned]

df_problematic_cleaned
```

[ ]:
```
Empty DataFrame
Columns: [País, Densidad (P/Km2), Abreviación, Terreno Agrícola (%), Área de
Tierra (Km2), Tamaño de las Fuerzas Armadas, Tasa de Natalidad, Código
Telefónico, Capital/Ciudad Principal, Emisiones de CO2, Índice de Precios al
Consumidor (IPC), Cambio del IPC (%), Código de Moneda, Tasa de Fertilidad, Área
Boscosa (%), Precio de la Gasolina, Producto Interno Bruto (PIB), Inscripción
Bruta en Educación Primaria (%), Inscripción Bruta en Educación Terciaria (%),
Mortalidad Infantil, Ciudad Más Grande, Esperanza de Vida, Tasa de Mortalidad
Materna, Salario Mínimo, Idioma Oficial, Gastos de Salud de Bolsillo (%),
Médicos por Mil Habitantes, Población, Participación en la Fuerza Laboral (%),
Ingresos Fiscales (%), Tasa Impositiva Total, Tasa de Desempleo, Población
Urbana, Latitud, Longitud]
Index: []

[0 rows x 35 columns]
```

- Vamos a **guardar** este archivo para chequearlo en excel

[ ]:
```
# Guardar el DataFrame limpio y sin símbolos en un archivo CSV para descarga
cleaned_file_path_final = '/content/sample_data/world-data-2023-cleaned-1-2.csv'
df.to_csv(cleaned_file_path_final, index=False)

cleaned_file_path_final, df.head()
```

[ ]:
```
('/content/sample_data/world-data-2023-cleaned-1-2.csv',
          País Densidad (P/Km2) Abreviación Terreno Agrícola (%)  \
 0  Afghanistan              60          AF               58.10%
 1      Albania             105          AL               43.10%
 2      Algeria              18          DZ               17.40%
 3      Andorra             164          AD               40.00%
 4       Angola              26          AO               47.50%

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
 0              652,230                       323,000              32.49
 1               28,748                         9,000              11.78
```

```
2            2,381,741                       317,000              24.28
3                  468                           NaN               7.20
4            1,246,700                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal Emisiones de CO2  …  \
0               93.0                     Kabul           8,672  …
1              355.0                    Tirana           4,536  …
2              213.0                   Algiers         150,006  …
3              376.0           Andorra la Vella             469  …
4              244.0                    Luanda          34,693  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                          78.40%                       0.28  38,041,754
1                          56.90%                       1.20   2,854,191
2                          28.10%                       1.72  43,053,054
3                          36.40%                       3.33      77,142
4                          33.40%                       0.21  31,825,295

   Participación en la Fuerza Laboral (%) Ingresos Fiscales (%)  \
0                                  48.90%                 9.30%
1                                  55.70%                18.60%
2                                  41.20%                37.20%
3                                     NaN                   NaN
4                                  77.50%                 9.20%

  Tasa Impositiva Total Tasa de Desempleo Población Urbana    Latitud  \
0                71.40%            11.12%        9,797,273  33.939110
1                36.60%            12.33%        1,747,593  41.153332
2                66.10%            11.70%       31,510,100  28.033886
3                   NaN               NaN           67,873  42.506285
4                49.10%             6.89%       21,061,025 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- **Por las dudas, vamos a cambiar nuevamente la codificacion**

[ ]:
```
import pandas as pd
# Cambia 'utf-8' por la codificación correcta
df = pd.read_csv('/content/sample_data/world-data-2023-cleaned-1-2.csv',␣
 ↪encoding='utf-8')
```

- **Cargamos nuevamente** para trabajar los datos faltantes

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-cleaned-1-2.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
 ↪visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
          País Densidad (P/Km2) Abreviación Terreno Agrícola (%)  \
0  Afghanistan              60          AF               58.10%
1      Albania             105          AL               43.10%
2      Algeria              18          DZ               17.40%
3      Andorra             164          AD               40.00%
4       Angola              26          AO               47.50%

  Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652,230                       323,000              32.49
1               28,748                         9,000              11.78
2            2,381,741                       317,000              24.28
3                  468                           NaN               7.20
4            1,246,700                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal Emisiones de CO2  …  \
0               93.0                   Kabul           8,672  …
```

```
1              355.0                  Tirana           4,536  …
2              213.0                 Algiers         150,006  …
3              376.0        Andorra la Vella             469  …
4              244.0                  Luanda          34,693  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                          78.40%                       0.28  38,041,754
1                          56.90%                       1.20   2,854,191
2                          28.10%                       1.72  43,053,054
3                          36.40%                       3.33      77,142
4                          33.40%                       0.21  31,825,295

   Participación en la Fuerza Laboral (%) Ingresos Fiscales (%)  \
0                                  48.90%                 9.30%
1                                  55.70%                18.60%
2                                  41.20%                37.20%
3                                     NaN                   NaN
4                                  77.50%                 9.20%

  Tasa Impositiva Total Tasa de Desempleo Población Urbana    Latitud  \
0                71.40%            11.12%        9,797,273  33.939110
1                36.60%            12.33%        1,747,593  41.153332
2                66.10%            11.70%       31,510,100  28.033886
3                   NaN               NaN           67,873  42.506285
4                49.10%             6.89%       21,061,025 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- **Chequear** datos faltantes

[ ]:
```
# Chequear datos faltantes

# Comprobación de valores faltantes por columna
missing_values = df.isnull().sum()

# Mostrar la cantidad de valores faltantes por columna
print(missing_values)
```

```
País                                0
Densidad (P/Km2)                    0
Abreviación                         6
```

```
Terreno Agrícola (%)                          7
Área de Tierra (Km2)                          1
Tamaño de las Fuerzas Armadas                24
Tasa de Natalidad                             6
Código Telefónico                             1
Capital/Ciudad Principal                      3
Emisiones de CO2                              7
Índice de Precios al Consumidor (IPC)        17
Cambio del IPC (%)                           16
Código de Moneda                             14
Tasa de Fertilidad                            7
Área Boscosa (%)                              7
Precio de la Gasolina                        20
Producto Interno Bruto (PIB)                  2
Inscripción Bruta en Educación Primaria (%)   7
Inscripción Bruta en Educación Terciaria (%) 12
Mortalidad Infantil                           6
Ciudad Más Grande                             6
Esperanza de Vida                             8
Tasa de Mortalidad Materna                   14
Salario Mínimo                               45
Idioma Oficial                                5
Gastos de Salud de Bolsillo (%)               7
Médicos por Mil Habitantes                    7
Población                                     1
Participación en la Fuerza Laboral (%)       19
Ingresos Fiscales (%)                        26
Tasa Impositiva Total                        12
Tasa de Desempleo                            19
Población Urbana                              5
Latitud                                       1
Longitud                                      1
dtype: int64
```

- **Rellenar** los valores faltantes

[ ]:
```
# Rellenar los valores faltantes
# Para las columnas numéricas (float64), vamos a rellenar con la media de la columna
df.fillna(df.mean(numeric_only=True), inplace=True)

# Para las columnas de tipo texto (object), vamos a rellenar con una cadena vacía donde sea apropiado
df.fillna('', inplace=True)

# Verificación de valores faltantes después del relleno
missing_values_after = df.isnull().sum()
```

```
# Realizar un chequeo de caracteres especiales en todo el dataset
# Reemplazaremos caracteres especiales en las columnas de tipo 'object'
import re

def clean_special_characters(text):
    if isinstance(text, str):
        return re.sub(r'[^\w\s.,%-]', '', text)
    return text

# Aplicar la limpieza de caracteres especiales a todo el DataFrame en las␣
 ↪columnas de tipo object
df[object_columns] = df[object_columns].applymap(clean_special_characters)

# Verificar si queda algún valor faltante o caracteres especiales
missing_values_after, df.head()
```

```
<ipython-input-24-0d3f99d904cb>:21: FutureWarning: DataFrame.applymap has been
deprecated. Use DataFrame.map instead.
  df[object_columns] = df[object_columns].applymap(clean_special_characters)
```

[ ]:
```
(País                                          0
 Densidad (P/Km2)                              0
 Abreviación                                   0
 Terreno Agrícola (%)                          0
 Área de Tierra (Km2)                          0
 Tamaño de las Fuerzas Armadas                 0
 Tasa de Natalidad                             0
 Código Telefónico                             0
 Capital/Ciudad Principal                      0
 Emisiones de CO2                              0
 Índice de Precios al Consumidor (IPC)         0
 Cambio del IPC (%)                            0
 Código de Moneda                              0
 Tasa de Fertilidad                            0
 Área Boscosa (%)                              0
 Precio de la Gasolina                         0
 Producto Interno Bruto (PIB)                  0
 Inscripción Bruta en Educación Primaria (%)   0
 Inscripción Bruta en Educación Terciaria (%)  0
 Mortalidad Infantil                           0
 Ciudad Más Grande                             0
 Esperanza de Vida                             0
 Tasa de Mortalidad Materna                    0
 Salario Mínimo                                0
 Idioma Oficial                                0
 Gastos de Salud de Bolsillo (%)               0
 Médicos por Mil Habitantes                    0
```

```
Población                                  0
Participación en la Fuerza Laboral (%)     0
Ingresos Fiscales (%)                      0
Tasa Impositiva Total                      0
Tasa de Desempleo                          0
Población Urbana                           0
Latitud                                    0
Longitud                                   0
dtype: int64,
          País Densidad (P/Km2) Abreviación Terreno Agrícola (%)  \
0  Afghanistan               60          AF               58.10%
1      Albania              105          AL               43.10%
2      Algeria               18          DZ               17.40%
3      Andorra              164          AD               40.00%
4       Angola               26          AO               47.50%

  Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652,230                       323,000              32.49
1               28,748                         9,000              11.78
2            2,381,741                       317,000              24.28
3                  468                                             7.20
4            1,246,700                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal Emisiones de CO2  …  \
0               93.0                     Kabul           8,672  …
1              355.0                    Tirana           4,536  …
2              213.0                   Algiers         150,006  …
3              376.0          Andorra la Vella             469  …
4              244.0                    Luanda          34,693  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                          78.40%                       0.28  38,041,754
1                          56.90%                       1.20   2,854,191
2                          28.10%                       1.72  43,053,054
3                          36.40%                       3.33      77,142
4                          33.40%                       0.21  31,825,295

   Participación en la Fuerza Laboral (%) Ingresos Fiscales (%)  \
0                                  48.90%                9.30%
1                                  55.70%               18.60%
2                                  41.20%               37.20%
3
4                                  77.50%                9.20%

  Tasa Impositiva Total Tasa de Desempleo Población Urbana    Latitud  \
0                71.40%            11.12%        9,797,273  33.939110
1                36.60%            12.33%        1,747,593  41.153332
```

```
2                66.10%            11.70%       31,510,100  28.033886
3                                                    67,873  42.506285
4                49.10%             6.89%       21,061,025 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- **Los valores faltantes han sido rellenados y ya no hay valores nulos** en el dataset. Además, **se ha realizado la limpieza de caracteres especiales en las columnas de tipo object**. A continuacion vamos a **limpiar los caracteres especiales tales como “,” “%” y “$”**

[ ]:
```python
# Función para eliminar símbolos y convertir a números
def clean_numeric_column(column):
    # Reemplazar símbolos como '%' y ',' y convertir a float
    column = column.str.replace('%', '', regex=False)
    column = column.str.replace(',', '', regex=False)
    column = column.str.replace('$', '', regex=False)

    # Convertir a float
    return pd.to_numeric(column, errors='coerce')

# Limpiar columnas que tienen símbolos y convertirlas a numéricas
columns_with_symbols = [
    'Densidad (P/Km2)', 'Terreno Agrícola (%)', 'Área de Tierra (Km2)',
    'Emisiones de CO2', 'Área Boscosa (%)', 'Precio de la Gasolina',
    'Inscripción Bruta en Educación Primaria (%)', 'Inscripción Bruta en␣
 ↪Educación Terciaria (%)',
    'Gastos de Salud de Bolsillo (%)', 'Participación en la Fuerza Laboral (%)',
    'Ingresos Fiscales (%)', 'Tasa Impositiva Total', 'Tasa de Desempleo',␣
 ↪'Población Urbana'
]

for column in columns_with_symbols:
    df[column] = clean_numeric_column(df[column])

# Verificación de la conversión y limpieza
df_info_after_cleaning = df.info()
df_head_after_cleaning = df.head()

df_info_after_cleaning, df_head_after_cleaning
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 35 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   País                                        195 non-null    object
 1   Densidad (P/Km2)                            195 non-null    int64
 2   Abreviación                                 195 non-null    object
 3   Terreno Agrícola (%)                        188 non-null    float64
 4   Área de Tierra (Km2)                        194 non-null    float64
 5   Tamaño de las Fuerzas Armadas               195 non-null    object
 6   Tasa de Natalidad                           195 non-null    float64
 7   Código Telefónico                           195 non-null    float64
 8   Capital/Ciudad Principal                    195 non-null    object
 9   Emisiones de CO2                            188 non-null    float64
 10  Índice de Precios al Consumidor (IPC)       195 non-null    object
 11  Cambio del IPC (%)                          195 non-null    object
 12  Código de Moneda                            195 non-null    object
 13  Tasa de Fertilidad                          195 non-null    float64
 14  Área Boscosa (%)                            188 non-null    float64
 15  Precio de la Gasolina                       175 non-null    float64
 16  Producto Interno Bruto (PIB)                195 non-null    object
 17  Inscripción Bruta en Educación Primaria (%) 188 non-null    float64
 18  Inscripción Bruta en Educación Terciaria (%) 183 non-null   float64
 19  Mortalidad Infantil                         195 non-null    float64
 20  Ciudad Más Grande                           195 non-null    object
 21  Esperanza de Vida                           195 non-null    float64
 22  Tasa de Mortalidad Materna                  195 non-null    float64
 23  Salario Mínimo                              195 non-null    object
 24  Idioma Oficial                              195 non-null    object
 25  Gastos de Salud de Bolsillo (%)             188 non-null    float64
 26  Médicos por Mil Habitantes                  195 non-null    float64
 27  Población                                   195 non-null    object
 28  Participación en la Fuerza Laboral (%)      176 non-null    float64
 29  Ingresos Fiscales (%)                       169 non-null    float64
 30  Tasa Impositiva Total                       183 non-null    float64
 31  Tasa de Desempleo                           176 non-null    float64
 32  Población Urbana                            190 non-null    float64
 33  Latitud                                     195 non-null    float64
 34  Longitud                                    195 non-null    float64
dtypes: float64(22), int64(1), object(12)
memory usage: 53.4+ KB
```

[ ]: (None,

```
          País  Densidad (P/Km2) Abreviación  Terreno Agrícola (%)  \
 0  Afghanistan                60          AF                  58.1
 1      Albania               105          AL                  43.1
```

```
2      Algeria                18          DZ                  17.4
3      Andorra               164          AD                  40.0
4       Angola                26          AO                  47.5

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0             652230.0                       323,000              32.49
1              28748.0                         9,000              11.78
2            2381741.0                       317,000              24.28
3                468.0                                             7.20
4            1246700.0                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal  Emisiones de CO2  …  \
0               93.0                    Kabul            8672.0  …
1              355.0                   Tirana            4536.0  …
2              213.0                  Algiers          150006.0  …
3              376.0         Andorra la Vella             469.0  …
4              244.0                   Luanda           34693.0  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                            78.4                       0.28  38,041,754
1                            56.9                       1.20   2,854,191
2                            28.1                       1.72  43,053,054
3                            36.4                       3.33      77,142
4                            33.4                       0.21  31,825,295

   Participación en la Fuerza Laboral (%)  Ingresos Fiscales (%)  \
0                                    48.9                   9.3
1                                    55.7                  18.6
2                                    41.2                  37.2
3                                     NaN                   NaN
4                                    77.5                   9.2

   Tasa Impositiva Total Tasa de Desempleo  Población Urbana    Latitud  \
0                   71.4             11.12        9797273.0  33.939110
1                   36.6             12.33        1747593.0  41.153332
2                   66.1             11.70       31510100.0  28.033886
3                    NaN               NaN          67873.0  42.506285
4                   49.1              6.89       21061025.0 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- **Generar archivo CSV** para descargar el dataset limpio. Mostrar nuevamente las primeras filas del dataset limpio para confirmar que todo está en orden.

```
[ ]: # Guardar el DataFrame limpio y sin símbolos en un archivo CSV para descarga
cleaned_file_path_final = '/content/sample_data/world-data-2023-cleaned-final.
 ↪csv'
df.to_csv(cleaned_file_path_final, index=False)

cleaned_file_path_final, df.head()
```

```
[ ]: ('/content/sample_data/world-data-2023-cleaned-final.csv',
          País  Densidad (P/Km2) Abreviación  Terreno Agrícola (%)  \
0  Afghanistan               60          AF                  58.1
1      Albania              105          AL                  43.1
2      Algeria               18          DZ                  17.4
3      Andorra              164          AD                  40.0
4       Angola               26          AO                  47.5

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0             652230.0                       323,000              32.49
1              28748.0                         9,000              11.78
2            2381741.0                       317,000              24.28
3                468.0                                             7.20
4            1246700.0                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal  Emisiones de CO2  …  \
0               93.0                    Kabul            8672.0  …
1              355.0                   Tirana            4536.0  …
2              213.0                  Algiers          150006.0  …
3              376.0          Andorra la Vella             469.0  …
4              244.0                   Luanda           34693.0  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                            78.4                       0.28  38,041,754
1                            56.9                       1.20   2,854,191
2                            28.1                       1.72  43,053,054
3                            36.4                       3.33      77,142
4                            33.4                       0.21  31,825,295

   Participación en la Fuerza Laboral (%)  Ingresos Fiscales (%)  \
0                                    48.9                   9.3
1                                    55.7                  18.6
2                                    41.2                  37.2
3                                     NaN                   NaN
4                                    77.5                   9.2

   Tasa Impositiva Total Tasa de Desempleo  Población Urbana    Latitud  \
```

```
0                  71.4             11.12         9797273.0  33.939110
1                  36.6             12.33         1747593.0  41.153332
2                  66.1             11.70        31510100.0  28.033886
3                   NaN               NaN           67873.0  42.506285
4                  49.1              6.89        21061025.0 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- **Cargar el dataset final**

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-cleaned-final.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
           País  Densidad (P/Km2) Abreviación  Terreno Agrícola (%)  \
```

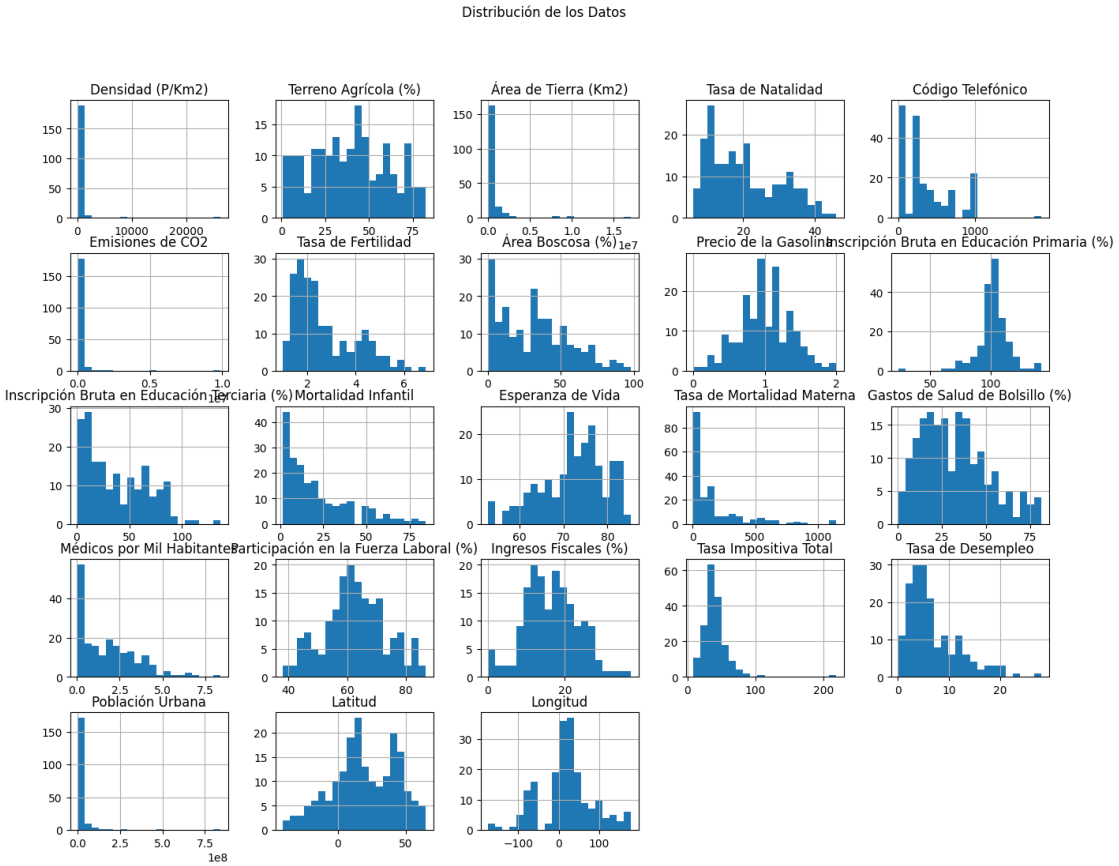
```
0  Afghanistan                60          AF                  58.1
1      Albania               105          AL                  43.1
2      Algeria                18          DZ                  17.4
3      Andorra               164          AD                  40.0
4       Angola                26          AO                  47.5

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652230.0                       323,000              32.49
1               28748.0                         9,000              11.78
2             2381741.0                       317,000              24.28
3                 468.0                           NaN               7.20
4             1246700.0                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal  Emisiones de CO2  …  \
0               93.0                    Kabul            8672.0  …
1              355.0                   Tirana            4536.0  …
2              213.0                  Algiers          150006.0  …
3              376.0        Andorra la Vella             469.0  …
4              244.0                   Luanda           34693.0  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                           78.4                       0.28  38,041,754
1                           56.9                       1.20   2,854,191
2                           28.1                       1.72  43,053,054
3                           36.4                       3.33      77,142
4                           33.4                       0.21  31,825,295

   Participación en la Fuerza Laboral (%)  Ingresos Fiscales (%)  \
0                                    48.9                   9.3
1                                    55.7                  18.6
2                                    41.2                  37.2
3                                     NaN                   NaN
4                                    77.5                   9.2

   Tasa Impositiva Total Tasa de Desempleo  Población Urbana    Latitud  \
0                   71.4             11.12        9797273.0  33.939110
1                   36.6             12.33        1747593.0  41.153332
2                   66.1             11.70       31510100.0  28.033886
3                    NaN               NaN          67873.0  42.506285
4                   49.1              6.89       21061025.0 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887
```

```
[5 rows x 35 columns])
```

- **Revisar la distribución de los datos**

[ ]:
```
import matplotlib.pyplot as plt

# Función para hacer un análisis básico de la distribución de los datos
def plot_distribution(df, title):
    plt.figure(figsize=(12, 8))
    df.hist(bins=20, figsize=(16, 12))
    plt.suptitle(title)
    plt.show()

# Hacer un análisis básico de la distribución de los datos
plot_distribution(df, 'Distribución de los Datos')
```

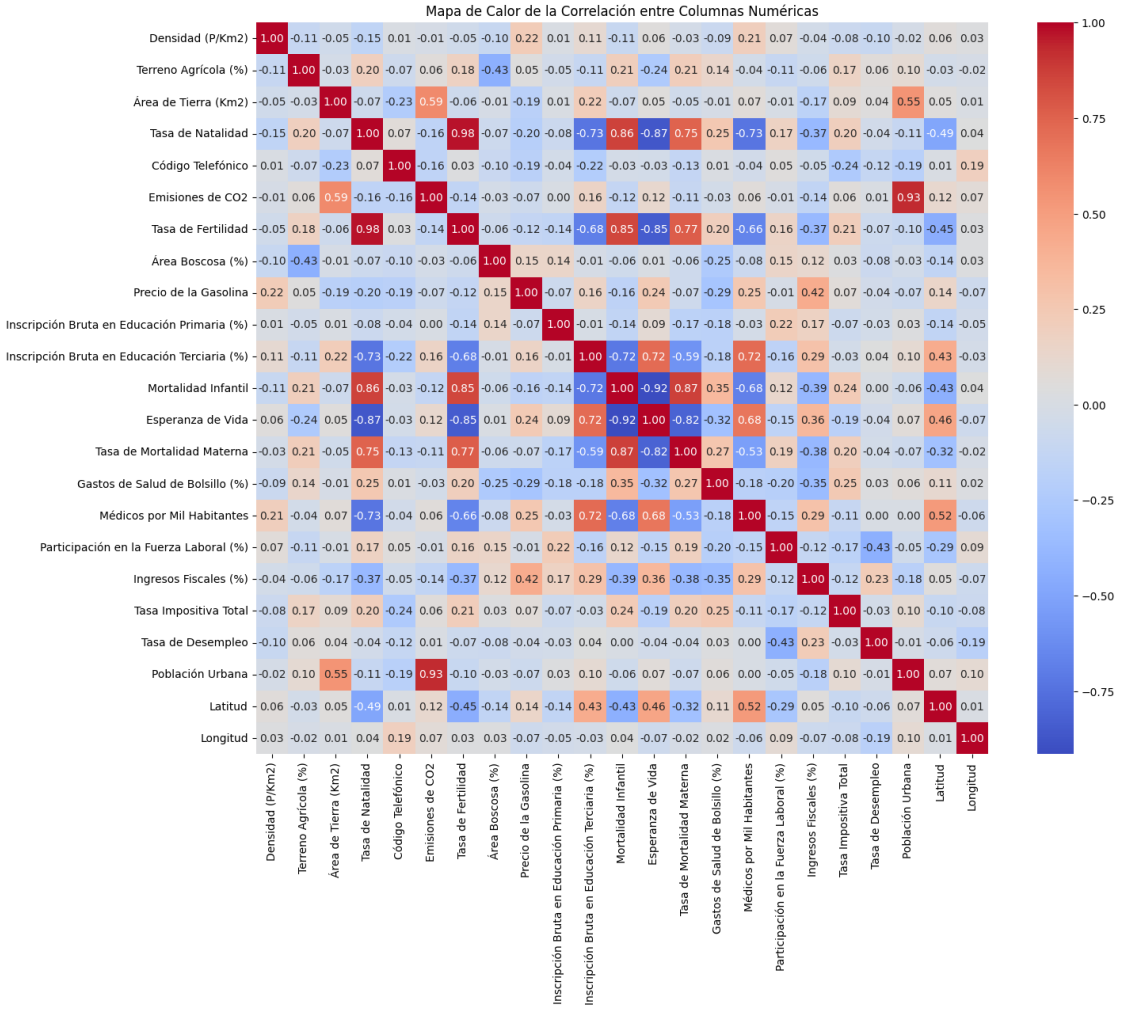
```
<Figure size 1200x800 with 0 Axes>
```



- **Vamos a proceder con dos tareas:** Primero vamos a visualizar la correlación entre columnas: Voy a generar un mapa de calor que muestre las correlaciones entre las columnas numéri-

cas del conjunto de datos.

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

# Calcular la matriz de correlación
correlation_matrix = df.select_dtypes(include=np.number).corr() # Seleccionar␣
 ↪sólo columnas numéricas para correlación

# Configurar el tamaño de la figura
plt.figure(figsize=(16, 12))

# Generar un mapa de calor con la matriz de correlación
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm",␣
 ↪square=True)

# Mostrar la visualización
plt.title('Mapa de Calor de la Correlación entre Columnas Numéricas')
plt.show()
```



Este mapa de calor muestra la **correlación entre las columnas numéricas del conjunto de datos.** Este gráfico **permite identificar relaciones lineales entre diferentes variables.** Los valores más cercanos a 1 o -1 indican una fuerte correlación positiva o negativa, respectivamente.

Vamos a proceder con la segunda tarea:

Análisis estadístico completo: Proporcionaré un resumen estadístico detallado de las columnas numéricas, incluyendo medidas de tendencia central, dispersión, y distribución.

Antes de ejecutar el analisis propiamente dicho vamos a instalar ace_tools_open. Basicamenten es una implementación abierta de ace_tools, referenciada en GPT4o. Este paquete permite a los usuarios utilizar herramientas específicas para la manipulación y visualización de datos.

Me di cuenta de que no necesito esa biblioteca, ya que Pandas ya cuenta con su propio ecosistema.

[ ]:
```
!pip install ace_tools
```

```
Collecting ace_tools
```

```
  Downloading ace_tools-0.0-py3-none-any.whl.metadata (300 bytes)
Downloading ace_tools-0.0-py3-none-any.whl (1.1 kB)
Installing collected packages: ace_tools
Successfully installed ace_tools-0.0
```

[ ]:
```
!pip install ace_tools_open
```

```
Collecting ace_tools_open
  Downloading ace_tools_open-0.1.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from ace_tools_open) (2.1.4)
Collecting itables (from ace_tools_open)
  Downloading itables-2.2.1-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: IPython in /usr/local/lib/python3.10/dist-
packages (from ace_tools_open) (7.34.0)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.10/dist-packages (from IPython->ace_tools_open) (71.0.4)
Collecting jedi>=0.16 (from IPython->ace_tools_open)
  Using cached jedi-0.19.1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-
packages (from IPython->ace_tools_open) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-
packages (from IPython->ace_tools_open) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.10/dist-
packages (from IPython->ace_tools_open) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from IPython->ace_tools_open) (3.0.47)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-
packages (from IPython->ace_tools_open) (2.18.0)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-
packages (from IPython->ace_tools_open) (0.2.0)
Requirement already satisfied: matplotlib-inline in
/usr/local/lib/python3.10/dist-packages (from IPython->ace_tools_open) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-
packages (from IPython->ace_tools_open) (4.9.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from itables->ace_tools_open) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->ace_tools_open) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->ace_tools_open) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->ace_tools_open) (2024.1)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
jedi>=0.16->IPython->ace_tools_open) (0.8.4)
Requirement already satisfied: ptyprocess>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
```

```
pexpect>4.3->IPython->ace_tools_open) (0.7.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-
packages (from prompt-
toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->IPython->ace_tools_open) (0.2.13)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas->ace_tools_open) (1.16.0)
Downloading ace_tools_open-0.1.0-py3-none-any.whl (3.0 kB)
Downloading itables-2.2.1-py3-none-any.whl (1.4 MB)
                         1.4/1.4 MB
13.2 MB/s eta 0:00:00
Using cached jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
Installing collected packages: jedi, itables, ace_tools_open
Successfully installed ace_tools_open-0.1.0 itables-2.2.1 jedi-0.19.1
```

[ ]:
```
# NO EJECUTAR, SE PUEDE HACER CON PANDAS.
#import pandas as pd
#data = {'col1': [1, 2, 3], 'col2': [4, 5, 6]}
#df_cleaned = pd.DataFrame(data)

# Generar un análisis estadístico completo para las columnas numéricas
#statistical_summary = df_cleaned.describe()

#import ace_tools_open as tools # Import the package you installed
#tools.display_dataframe_to_user(name="Resumen Estadístico del Dataset",␣
 ↪dataframe=statistical_summary)
```

```
Resumen Estadístico del Dataset

<IPython.core.display.HTML object>
```

[ ]:
```
# Generar un análisis estadístico completo para las columnas numéricas con␣
 ↪pandas

df_cleaned = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]}) # Ejemplo

statistical_summary = df_cleaned.describe()

# Visualizar el resumen estadístico mediante la funcionalidad de pandas
print("Resumen Estadístico del Dataset:")
print(statistical_summary)
```

```
Resumen Estadístico del Dataset:
              A    B
count  3.0  3.0
mean   2.0  5.0
std    1.0  1.0
min    1.0  4.0
25%    1.5  4.5
```

```
50%    2.0  5.0
75%    2.5  5.5
max    3.0  6.0
```

2 Hipótesis 1: "Economia vs. Filosofia: Liberalismo vs Comunismo"

Los países con economías más liberales (menor intervención estatal y mayor libertad de mercado) muestran un mayor crecimiento del PIB y una menor tasa de desempleo en comparación con los países con economías más planificadas (comunistas/socialistas).

- Clasificación de países en Liberalismo vs. Comunismo:

Vamos a definir un criterio para clasificar los países en categorías de "economías liberales" y "economías comunistas/socialistas". Para ello, utilizo fuentes externas o bien categorizaciones estándar sobre el grado de intervención estatal en la economía de cada país.

[ ]:
```
# Definir la clasificación de países en economías liberales vs comunistas/
 ↪socialistas
# Esta clasificación es aproximada y se basa en conocimiento general sobre la␣
 ↪estructura económica de los países
# La clasificación real puede variar dependiendo de las fuentes que se utilicen␣
 ↪para definir "liberalismo" y "comunismo"

liberal_countries = ['United States', 'Canada', 'Australia', 'United Kingdom',␣
 ↪'Germany', 'Japan', 'South Korea']
communist_countries = ['China', 'Cuba', 'Vietnam', 'North Korea', 'Laos']

# df_imputed esté definido y contenga sus datos
# Reemplace esto con su DataFrame real
df_imputed = pd.DataFrame({'País': ['United States', 'China', 'Germany',␣
 ↪'Cuba', 'Brazil'],
                           'PIB': [20000, 15000, 5000, 1000, 3000],
                           'Tasa de desempleo': [5, 4, 6, 10, 8]})

# Añadir una nueva columna al DataFrame que clasifique los países
df_imputed['Economía'] = df_imputed['País'].apply(lambda x: 'Liberal' if x in␣
 ↪liberal_countries else ('Comunista' if x in communist_countries else 'Otro'))

# Filtrar solo los países que hemos clasificado como liberales o comunistas
df_economy_classified = df_imputed[df_imputed['Economía'].isin(['Liberal',␣
 ↪'Comunista'])]

# Ver los primeros resultados para asegurarnos de que la clasificación se haya␣
 ↪aplicado correctamente
df_economy_classified[['País', 'Economía', 'PIB', 'Tasa de desempleo']].head()
```

[ ]:

	País	Economía	PIB	Tasa de desempleo
0	United States	Liberal	20000	5
1	China	Comunista	15000	4
2	Germany	Liberal	5000	6
3	Cuba	Comunista	1000	10

<google.colab._quickchart_helpers.SectionTitle at 0x7cf77ae09d80>

```
from matplotlib import pyplot as plt
_df_0['PIB'].plot(kind='hist', bins=20, title='PIB')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_1['Tasa de desempleo'].plot(kind='hist', bins=20, title='Tasa de desempleo')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

<google.colab._quickchart_helpers.SectionTitle at 0x7cf77b413910>

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_2.groupby('País').size().plot(kind='barh', color=sns.palettes.
 ↪mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_3.groupby('Economía').size().plot(kind='barh', color=sns.palettes.
 ↪mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

<google.colab._quickchart_helpers.SectionTitle at 0x7cf77ae09540>

```
from matplotlib import pyplot as plt
_df_4.plot(kind='scatter', x='PIB', y='Tasa de desempleo', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

<google.colab._quickchart_helpers.SectionTitle at 0x7cf7787aeb90>

```
from matplotlib import pyplot as plt
_df_5['PIB'].plot(kind='line', figsize=(8, 4), title='PIB')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_6['Tasa de desempleo'].plot(kind='line', figsize=(8, 4), title='Tasa de␣
 ↪desempleo')
plt.gca().spines[['top', 'right']].set_visible(False)
```

<google.colab._quickchart_helpers.SectionTitle at 0x7cf77ae08460>

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
```

```
df_2dhist = pd.DataFrame({
    x_label: grp['Economía'].value_counts()
    for x_label, grp in _df_7.groupby('País')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('País')
_ = plt.ylabel('Economía')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7cf7ab657370>
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_8['País'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_8, x='PIB', y='País', inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_9['Economía'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_9, x='PIB', y='Economía', inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_10['País'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_10, x='Tasa de desempleo', y='País', inner='stick',␣
↪palette='Dark2')
```

```
sns.despine(top=True, right=True, bottom=True, left=True)
```

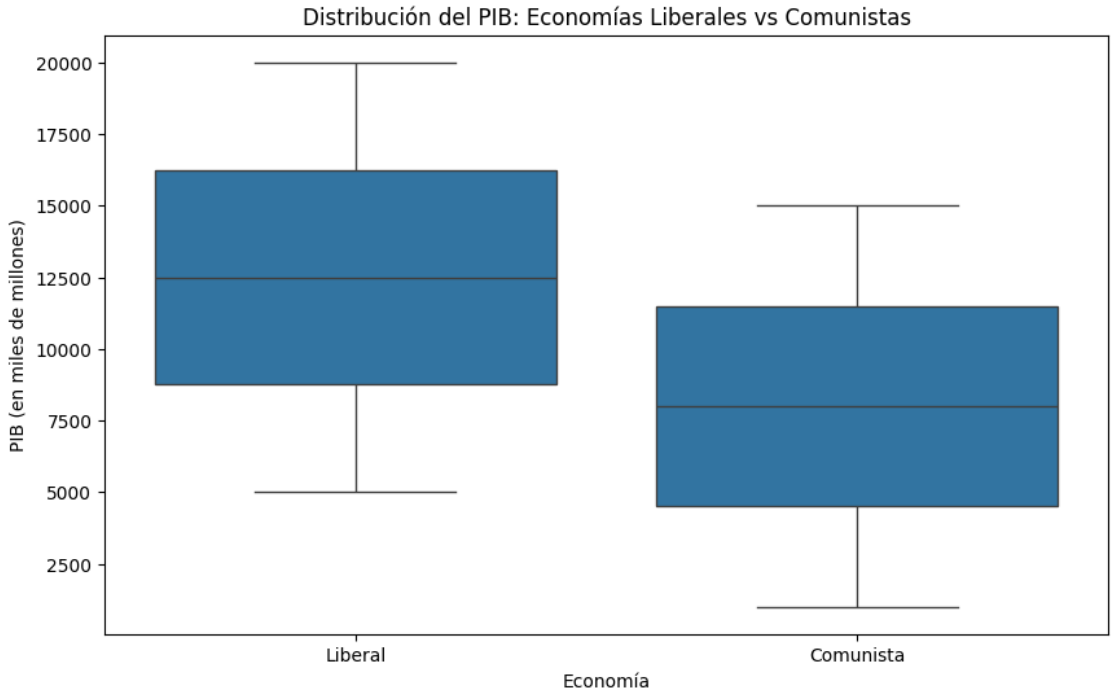
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

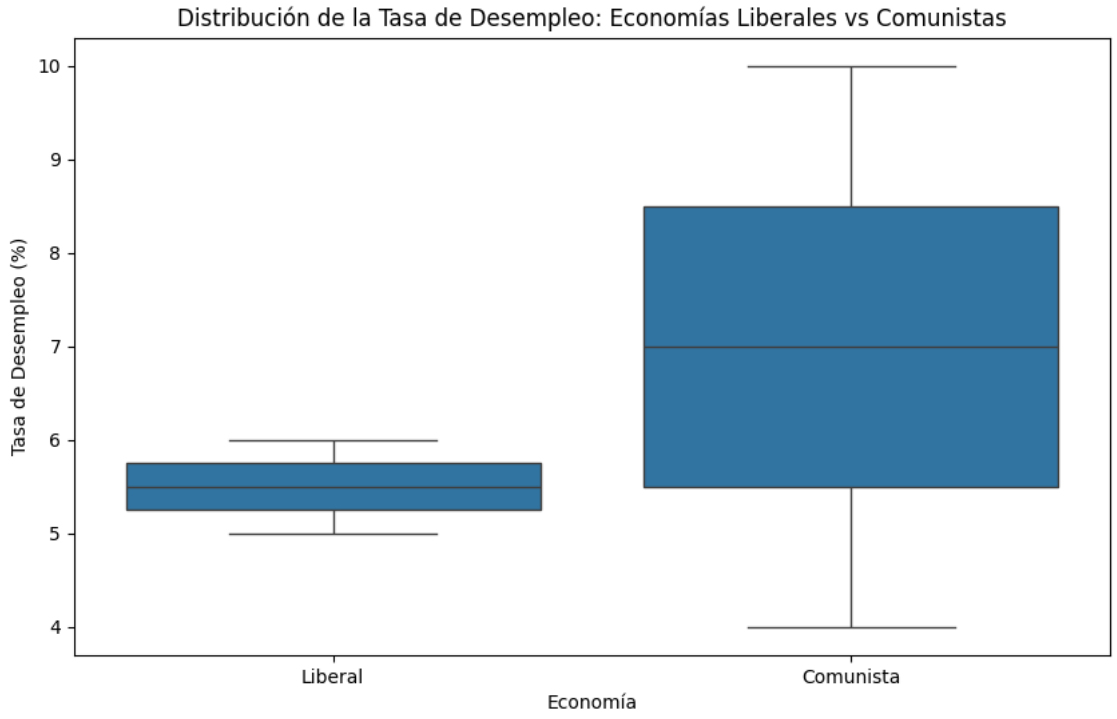
```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_11['Economía'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_11, x='Tasa de desempleo', y='Economía', inner='stick',␣
 ↪palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

df_economy_classified['Tasa de desempleo'] = df_economy_classified['Tasa de␣
 ↪desempleo'].astype(float)

# Estadísticas descriptivas por tipo de economía
economic_stats = df_economy_classified.groupby('Economía')[['PIB', 'Tasa de␣
 ↪desempleo']].describe()

# Visualización de las diferencias en PIB entre economías liberales y comunistas
plt.figure(figsize=(10, 6))
sns.boxplot(x='Economía', y='PIB', data=df_economy_classified)
plt.title('Distribución del PIB: Economías Liberales vs Comunistas')
plt.ylabel('PIB (en miles de millones)')
plt.show()

# Visualización de las diferencias en la tasa de desempleo entre economías␣
 ↪liberales y comunistas
plt.figure(figsize=(10, 6))
sns.boxplot(x='Economía', y='Tasa de desempleo', data=df_economy_classified)
plt.title('Distribución de la Tasa de Desempleo: Economías Liberales vs␣
 ↪Comunistas')
plt.ylabel('Tasa de Desempleo (%)')
plt.show()

economic_stats
```

```
<ipython-input-34-2f663c3e7ce7>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_economy_classified['Tasa de desempleo'] = df_economy_classified['Tasa de
desempleo'].astype(float)
```





[ ]:

Economía	PIB count	PIB mean	PIB std	PIB min	PIB 25%	PIB 50%	PIB 75%
Comunista	2.0	8000.0	9899.494937	1000.0	4500.0	8000.0	11500.0
Liberal	2.0	12500.0	10606.601718	5000.0	8750.0	12500.0	16250.0

Economía	PIB max	Tasa de desempleo count	Tasa de desempleo mean	Tasa de desempleo std	Tasa de desempleo min	Tasa de desempleo 25%	Tasa de desempleo 50%	Tasa de desempleo 75%
Comunista	15000.0	2.0	7.0	4.242641	4.0	5.50	7.0	8.50
Liberal	20000.0	2.0	5.5	0.707107	5.0	5.25	5.5	5.75

Economía	Tasa de desempleo max
Comunista	10.0
Liberal	6.0

###El gráfico de caja o "boxplot" presentado muestra la distribución del PIB en miles de millones para países con economías liberales y comunistas. Aquí está el análisis:

1. Economías liberales: - La mediana del PIB para los países con economías liberales parece estar alrededor de los 12,500 mil millones. - La caja del gráfico, que representa el rango intercuartílico (IQR), indica que la mayoría de los países liberales tienen un PIB que varía entre aproximadamente 10,000 a 15,000 mil millones. - Hay una variación considerable en el PIB dentro de esta categoría,

con valores mínimos cercanos a los 5,000 mil millones y máximos por encima de los 20,000 mil millones.

2. Economías comunistas: - La mediana del PIB para los países comunistas es menor, en torno a los 9,500 mil millones. - El IQR para estos países parece ser más pequeño que el de las economías liberales, mostrando una menor dispersión de los datos. La caja muestra que el 50% de los países comunistas tienen un PIB entre 7,500 mil millones y 12,500 mil millones, lo que indica una distribución más compacta en comparación con las economías liberales - Rango total: Las líneas de los bigotes se extienden desde aproximadamente 2,500 mil millones hasta un poco por encima de 15,000 mil millones, lo que sugiere que las economías comunistas tienen un rango más estrecho en comparación con las economías liberales.

3. Comparación:

- Dispersión: Las economías liberales muestran una mayor dispersión en el PIB, lo que implica más variabilidad entre los países liberales. Esto podría indicar que algunos países liberales tienen PIB significativamente más altos que otros.
- Mediana: La mediana del PIB en los países liberales es ligeramente mayor que en los países comunistas, lo que sugiere un PIB típico algo más alto en estos países.
- Rango: Las economías liberales tienden a tener un rango de PIB más amplio, mientras que las economías comunistas parecen estar más concentradas en valores más bajos.

###El gráfico de caja presentado compara la tasa de desempleo (%) entre países con economías liberales y comunistas. A continuación, el análisis:

1. Economías Liberales: - La mediana de la tasa de desempleo en estos países está cercana al 5.5%. - El rango intercuartílico (IQR) es pequeño, lo que indica que la mayoría de las tasas de desempleo en las economías liberales están concentradas en un margen reducido, aproximadamente entre el 5% y el 6%. - El valor más bajo en la tasa de desempleo en estas economías ronda el 4%, y el más alto no supera el 7%, lo que sugiere una variabilidad bastante controlada en el desempleo.

2. Economías Comunistas: - La mediana en las economías comunistas es algo más alta, ubicándose alrededor del 7%. - El IQR es mucho más amplio que en las economías liberales, lo que muestra una mayor dispersión en la tasa de desempleo. Las tasas varían desde un mínimo cercano al 6% hasta un máximo de aproximadamente 9%. - Esto sugiere que los países comunistas experimentan una mayor variabilidad en su tasa de desempleo, con algunos países enfrentando niveles significativamente más altos que otros.

3. Comparación:

- Mediana: Los países comunistas tienden a tener una mediana de desempleo más alta que los países liberales, lo que sugiere que, en general, las economías comunistas presentan mayores tasas de desempleo.
- Dispersión: Las economías comunistas muestran una mayor dispersión en las tasas de desempleo, lo que implica una mayor variabilidad entre los países. En contraste, las economías liberales parecen tener una tasa de desempleo más estable y controlada.
- Rangos: Mientras que las economías liberales parecen mantener el desempleo en un rango muy acotado, las economías comunistas tienen un rango más amplio, alcanzando hasta el 10%, lo que podría reflejar desafíos económicos o diferencias estructurales significativas en la gestión del empleo en estas naciones.

2.1 Conclusión parcial de la hipótesis 1:

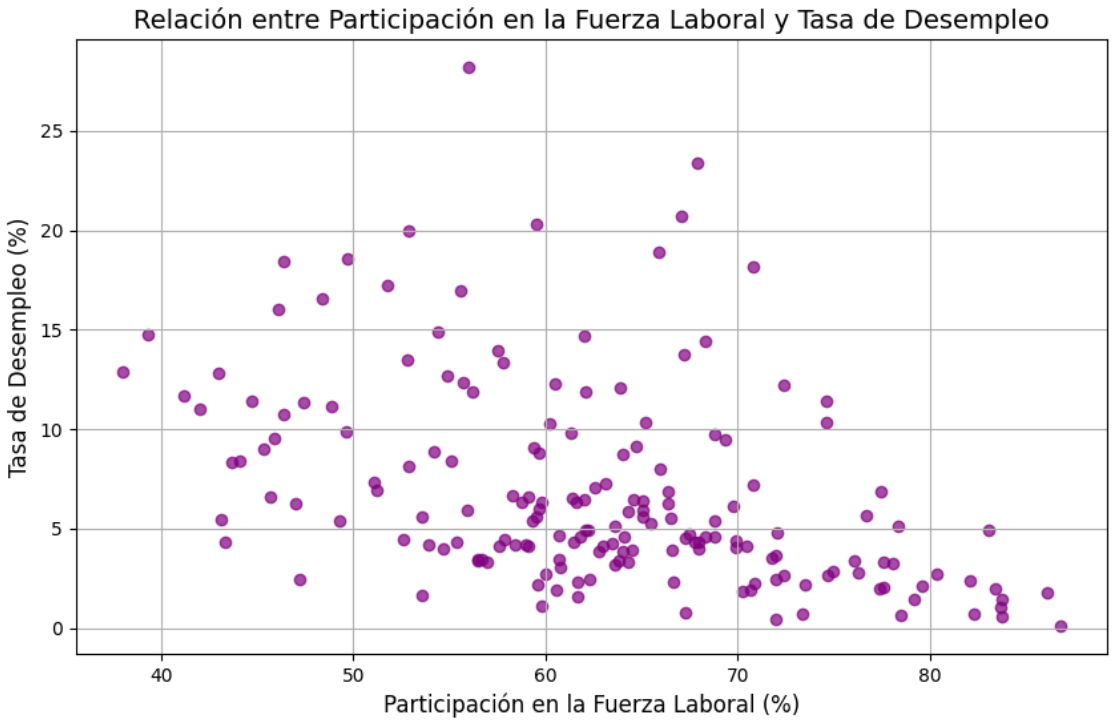
El gráfico 1 podría respaldar la idea de que las economías liberales, en promedio, tienen un PIB más alto y mayor variabilidad, mientras que las economías comunistas muestran PIB más uniformes, pero generalmente más bajos. Este gráfico 2 podría sugerir que los países con economías comunistas experimentan mayores tasas de desempleo y una variabilidad más amplia, en comparación con las economías liberales, que parecen mantener el desempleo dentro de un rango más limitado.

3 Hipótesis 2: "Fuerza laboral vs. Tasa de desempleo":

Los países con una mayor tasa de participación en la fuerza laboral tienen una menor tasa de desempleo.

- Para esta hipótesis, vamos a explorar la relación entre la tasa de participación en la fuerza laboral y la tasa de desempleo. Un gráfico de dispersión (scatter plot) sería útil para observar si hay una tendencia inversa entre estas dos variables.

[ ]:
```
# Convertir las columnas relevantes a tipo numérico para graficar
df['Participación en la Fuerza Laboral (%)'] = pd.to_numeric(df['Participación␣
 ↪en la Fuerza Laboral (%)'], errors='coerce')
df['Tasa de Desempleo'] = pd.to_numeric(df['Tasa de Desempleo'],␣
 ↪errors='coerce')

# Crear el gráfico de dispersión entre participación en la fuerza laboral y␣
 ↪tasa de desempleo
plt.figure(figsize=(10, 6))
plt.scatter(df['Participación en la Fuerza Laboral (%)'], df['Tasa de␣
 ↪Desempleo'], alpha=0.7, color='purple')
plt.title('Relación entre Participación en la Fuerza Laboral y Tasa de␣
 ↪Desempleo', fontsize=14)
plt.xlabel('Participación en la Fuerza Laboral (%)', fontsize=12)
plt.ylabel('Tasa de Desempleo (%)', fontsize=12)
plt.grid(True)
plt.show()
```



El **gráfico de dispersión** presentado muestra la relación entre la Participación en la Fuerza Laboral (%) en el eje X y la Tasa de Desempleo (%) en el eje Y.

4 Análisis del gráfico: Relación entre Participación en la Fuerza Laboral y Tasa de Desempleo:

1- Distribución de los puntos:

Los puntos están dispersos a lo largo de un rango amplio de participación en la fuerza laboral (de aproximadamente 40% a 80%). La tasa de desempleo varía significativamente, desde cerca de 0% hasta valores superiores al 25%.

2. Tendencia general:

No parece haber una correlación fuerte y clara entre la participación en la fuerza laboral y la tasa de desempleo. Aunque la hipótesis sugiere que una mayor participación en la fuerza laboral debería estar asociada con una menor tasa de desempleo, el gráfico no muestra una tendencia descendente clara. De hecho, en algunos países con una alta participación laboral, la tasa de desempleo sigue siendo considerablemente alta (alrededor de 10-20%). Hay países con baja participación laboral y baja tasa de desempleo, así como países con alta participación laboral y alta tasa de desempleo. Esto indica que la relación entre estas dos variables no es tan directa como la hipótesis sugiere.

3. Desviaciones:

Hay algunos puntos con participación laboral entre el 60% y el 80% que muestran tasas de desempleo muy bajas (entre 0% y 5%), lo que respaldaría parcialmente la hipótesis. Sin embargo, también se

observan varios casos con tasas de participación entre el 50% y el 70%, donde la tasa de desempleo es mayor al 10%, lo que indica que no siempre una mayor participación está correlacionada con menor desempleo.

4. Dispersión:

Existe una dispersión considerable en el gráfico. Aunque algunos países con altas tasas de participación laboral (por encima del 70%) tienen tasas de desempleo bajas (por debajo del 5%), también hay varios puntos que no siguen este patrón, lo que sugiere que otros factores podrían estar influyendo. Muchos países con tasas de participación en la fuerza laboral entre 50% y 70% presentan tasas de desempleo variables que oscilan entre 5% y 15%, sin una clara correlación entre ambas variables.

5 Conclusión parcial de la hipótesis 2:

Rechazo parcial de la hipótesis:

El gráfico no respalda completamente la hipótesis de que una mayor participación en la fuerza laboral conduce a una menor tasa de desempleo. Si bien algunos países siguen esta tendencia, la dispersión de los puntos indica que otros factores juegan un papel importante en la determinación de la tasa de desempleo.

Posibles influencias externas que podrian influir:

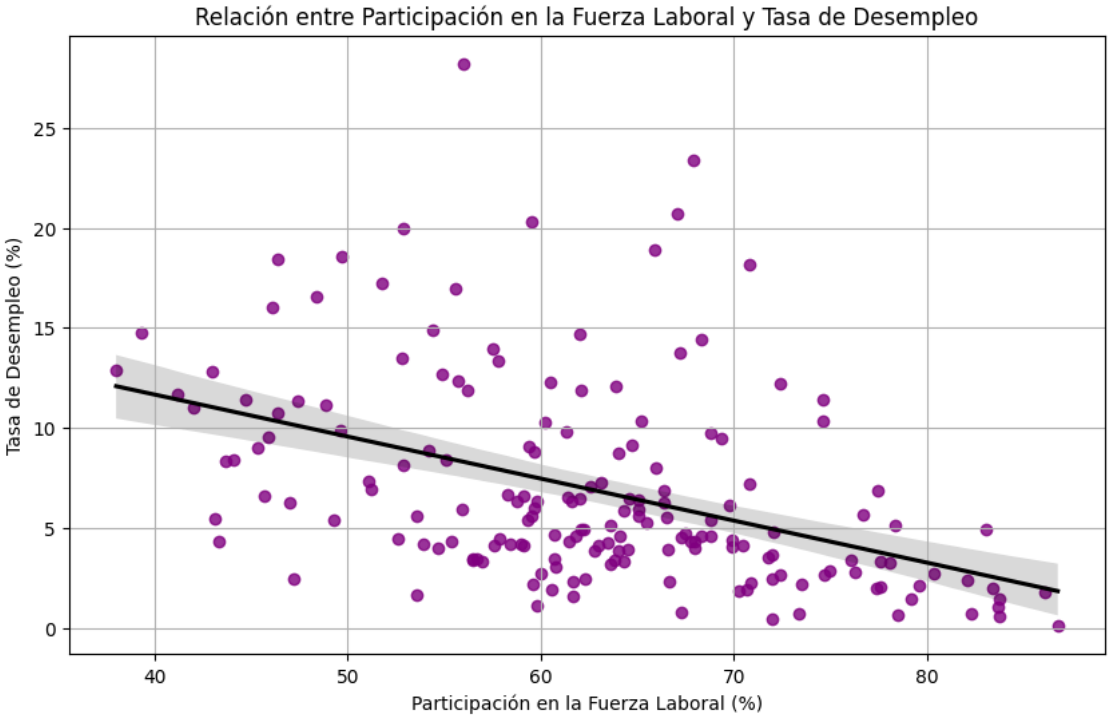
Factores como la calidad del empleo, políticas laborales, educación, o sector económico podrían estar influyendo en la tasa de desempleo independientemente de la participación en la fuerza laboral.

[ ]:
```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

# Convertir las columnas relevantes a tipo numérico
df['Participación en la Fuerza Laboral (%)'] = pd.to_numeric(df['Participación␣
 ↪en la Fuerza Laboral (%)'], errors='coerce')
df['Tasa de Desempleo'] = pd.to_numeric(df['Tasa de Desempleo'],␣
 ↪errors='coerce')

# Eliminar filas con valores NaN o infinitos en las columnas seleccionadas
df_clean = df[['Participación en la Fuerza Laboral (%)', 'Tasa de Desempleo']].
 ↪replace([np.inf, -np.inf], np.nan).dropna()

# Preparar los datos para la regresión
X = df_clean[['Participación en la Fuerza Laboral (%)']]  # Variable␣
 ↪independiente
y = df_clean['Tasa de Desempleo']  # Variable dependiente

# Añadir una constante para el término independiente en el modelo de regresión
X = sm.add_constant(X)
```

```python
# Ajustar el modelo de regresión lineal
modelo = sm.OLS(y, X).fit()

# Imprimir el resumen de la regresión
print(modelo.summary())

# Visualización: gráfico de regresión con la línea de ajuste
plt.figure(figsize=(10, 6))
sns.regplot(x='Participación en la Fuerza Laboral (%)', y='Tasa de Desempleo',␣
 ↪data=df_clean, scatter_kws={"color": "purple"}, line_kws={"color": "black"})
plt.title('Relación entre Participación en la Fuerza Laboral y Tasa de␣
 ↪Desempleo')
plt.xlabel('Participación en la Fuerza Laboral (%)')
plt.ylabel('Tasa de Desempleo (%)')
plt.grid(True)
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      Tasa de Desempleo   R-squared:                       0.188
Model:                            OLS   Adj. R-squared:                  0.184
Method:                 Least Squares   F-statistic:                     40.41
Date:                Wed, 25 Sep 2024   Prob (F-statistic):           1.75e-09
Time:                        02:14:55   Log-Likelihood:                -516.88
No. Observations:                 176   AIC:                             1038.
Df Residuals:                     174   BIC:                             1044.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                                                 coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
--------------------------
const                                       20.0583      2.101      9.548
0.000      15.912      24.204
Participación en la Fuerza Laboral (%)    -0.2100      0.033     -6.357
0.000      -0.275      -0.145
==============================================================================
Omnibus:                       62.637   Durbin-Watson:                   1.905
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              146.654
Skew:                           1.584   Prob(JB):                     1.43e-32
Kurtosis:                       6.157   Cond. No.                         386.
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```



6 Análisis del gráfico: Relación entre Participación en la Fuerza Laboral y Tasa de Desempleo

Este gráfico muestra una línea de regresión que representa la relación entre la participación en la fuerza laboral (%) (en el eje X) y la tasa de desempleo (%) (en el eje Y). Vamos a analizarlo en el contexto de la hipótesis planteada.

7 Observaciones clave del gráfico:

1- Tendencia general:

La línea de regresión muestra una **tendencia descendente**. Esto significa que, en general, hay una relación negativa entre la participación en la fuerza laboral y la tasa de desempleo. Es decir, a medida que aumenta la participación en la fuerza laboral, tiende a disminuir la tasa de desempleo.

2- Dispersión de los puntos:

A pesar de que la línea de regresión muestra una **tendencia negativa**, la dispersión de los puntos es considerable. No todos los puntos siguen el patrón de la línea, lo que sugiere que hay otros factores influyendo en la tasa de desempleo además de la participación en la fuerza laboral. Se puede observar que algunos países con baja participación laboral (alrededor del 40%) tienen tasas

de desempleo más altas (hasta el 20%), lo cual es consistente con la hipótesis. Sin embargo, también hay países con alta participación (más del 70%) que siguen teniendo tasas de desempleo considerables (más del 10%).

3- Relación no completamente lineal:

La nube de puntos muestra una **dispersión considerable**, particularmente en los valores de participación laboral entre 50% y 70%. Aunque la tendencia general es descendente, los puntos dispersos sugieren que no todos los países siguen estrictamente la tendencia.

8 Análisis del resultado de la regresión OLS:

El resultado de la regresión lineal te proporciona información clave sobre la relación entre la participación en la fuerza laboral y la tasa de desempleo. Vamos a analizar cada parte importante del resultado:

1. **Coeficientes de la regresión:**

- Intercepto (const): El valor de 20.0583 indica que si la participación en la fuerza laboral fuera 0, la tasa de desempleo esperada sería del 20.06%. Este valor es solo teórico porque es poco probable que haya un país con 0% de participación en la fuerza laboral.
- Participación en la Fuerza Laboral (%): El coeficiente de -0.2100 significa que por cada aumento de 1 punto porcentual en la participación en la fuerza laboral, se espera que la tasa de desempleo disminuya en 0.21 puntos porcentuales. Esta relación es estadísticamente significativa, como lo muestra el valor p asociado.

2. **Significancia estadística:**

- P-valor del F-statistic: El valor p de la F-statistic es 1.75e-09, lo que indica que el modelo en su conjunto es estadísticamente significativo. Esto significa que existe suficiente evidencia para concluir que la participación en la fuerza laboral tiene un impacto significativo en la tasa de desempleo.
- Valor p para el coeficiente de participación en la fuerza laboral: El valor p es 0.000, lo que indica que la relación entre la participación en la fuerza laboral y la tasa de desempleo es altamente significativa a nivel estadístico.

3. **R^2 y R^2 ajustado:**

- R^2 (0.188): Este valor indica que aproximadamente el 18.8% de la variación en la tasa de desempleo está explicada por la participación en la fuerza laboral. Esto es relativamente bajo, lo que sugiere que hay otros factores que también influyen en la tasa de desempleo y no están siendo capturados por el modelo.
- R^2 ajustado (0.184): Similar al R^2, este valor ajustado por el número de variables en el modelo también indica que el modelo solo explica una pequeña porción de la variación en la tasa de desempleo.

4. **Test de normalidad y otros estadísticos:**

- Durbin-Watson (1.905): El valor del test Durbin-Watson está cerca de 2, lo que sugiere que no hay una autocorrelación significativa en los residuos.
- Omnibus (62.637) y Jarque-Bera (146.654): Estos valores indican que los residuos no siguen una distribución normal. El alto valor de Skewness (asimetría) y Kurtosis sugiere que puede

haber outliers o sesgos en los datos, lo cual puede estar afectando la precisión del modelo.

9 Conclusión de la hipótesis 2:

En términos generales **en el grafico los países con una mayor participación en la fuerza laboral tienden a tener una menor tasa de desempleo**, como lo indica la pendiente negativa de la línea de regresión. Sin embargo, debido a la gran dispersión de los puntos, la relación no es completamente fuerte o directa. Existen países con altas tasas de participación laboral que todavía enfrentan altas tasas de desempleo, lo que indica que otros factores están en juego.

Por otro lado, si analizamos la **regresion OLS tenemos una relación significativa pero débil**, es decir, la participación en la fuerza laboral tiene una relación negativa y significativa con la tasa de desempleo, lo que apoya parcialmente la hipótesis. Sin embargo, el R^2 bajo (0.188) sugiere que esta variable solo explica una parte pequeña de la variación en la tasa de desempleo.

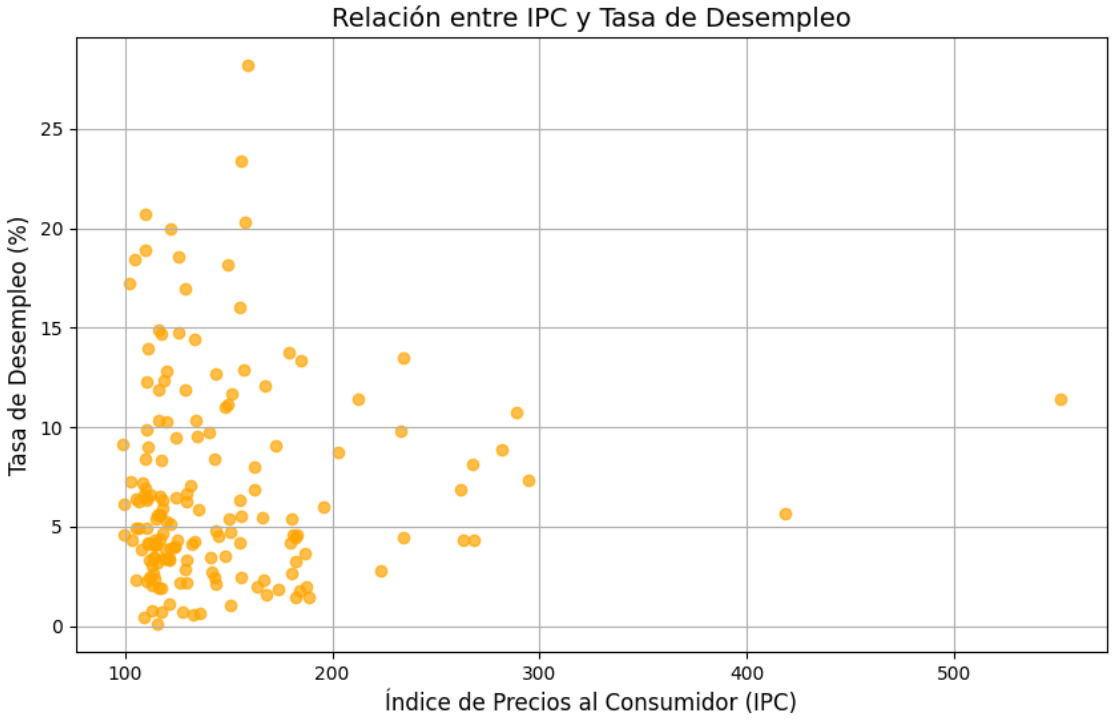
Antes de llegar a una conclusión, reflexiono sobre nuestra experiencia en Argentina y, automáticamente, dos preguntas surgen en mi mente.

#¿Como afecta la inflacion?¿Qué países destacan en desempleo bajo?

Podemos analizar cómo la **inflación (representada por el Índice de Precios al Consumidor, IPC) afecta la tasa de desempleo**. Además, identificaremos los países que destacan por tener una baja tasa de desempleo.

1. Primero, generaremos un gráfico que relacione el IPC con la tasa de desempleo para observar si existe alguna correlación. Luego, hago una lista de los países con las tasas de desempleo más bajas.

[ ]:
```
# Convertir la columna IPC a numérica para graficar
df['Índice de Precios al Consumidor (IPC)'] = pd.to_numeric(df['Índice de␣
 ↪Precios al Consumidor (IPC)'], errors='coerce')

# Crear un gráfico de dispersión entre IPC y Tasa de Desempleo
plt.figure(figsize=(10, 6))
plt.scatter(df['Índice de Precios al Consumidor (IPC)'], df['Tasa de␣
 ↪Desempleo'], alpha=0.7, color='orange')
plt.title('Relación entre IPC y Tasa de Desempleo', fontsize=14)
plt.xlabel('Índice de Precios al Consumidor (IPC)', fontsize=12)
plt.ylabel('Tasa de Desempleo (%)', fontsize=12)
plt.grid(True)
plt.show()

# Encontrar los países con la tasa de desempleo más baja
lowest_unemployment_countries = df[['País', 'Tasa de Desempleo']].
 ↪sort_values(by='Tasa de Desempleo').head(10)
lowest_unemployment_countries
```



[ ]:

	País	Tasa de Desempleo
141	Qatar	0.09
125	Niger	0.47
159	Solomon Islands	0.58
92	Laos	0.63
30	Cambodia	0.68
12	Bahrain	0.71
173	Thailand	0.75
144	Rwanda	1.03
176	Tonga	1.12
121	Nepal	1.41

10 Análisis del gráfico: Relación entre IPC y Tasa de Desempleo

El gráfico de dispersión nos permite observar la relación entre la inflación (representada por el IPC) y la tasa de desempleo. Vamos a desglosar lo que este gráfico nos dice:

1. Tendencia general: - No parece haber una correlación clara entre el IPC y la tasa de desempleo. Los puntos en el gráfico están bastante dispersos, y no se observa una tendencia lineal o definida. - Esto sugiere que la inflación, tal como la mide el IPC, no parece estar directamente relacionada con la tasa de desempleo en la mayoría de los países. Esto es consistente con la teoría económica en la que, a largo plazo, el desempleo y la inflación no siempre están correlacionados de manera simple.

2. Patrones notables:

- La mayoría de los puntos se concentran en valores más bajos de IPC (entre 100 y 200), donde la tasa de desempleo varía de 0% a más de 20%. Sin embargo, hay países con IPC altos que mantienen tasas de desempleo bajas, lo que refuerza la idea de que otros factores también están en juego.
- Hay algunos puntos en el gráfico con IPC muy alto (superior a 400) y tasas de desempleo bajas, lo cual indica casos atípicos o políticas económicas específicas que podrían estar influyendo en la relación.

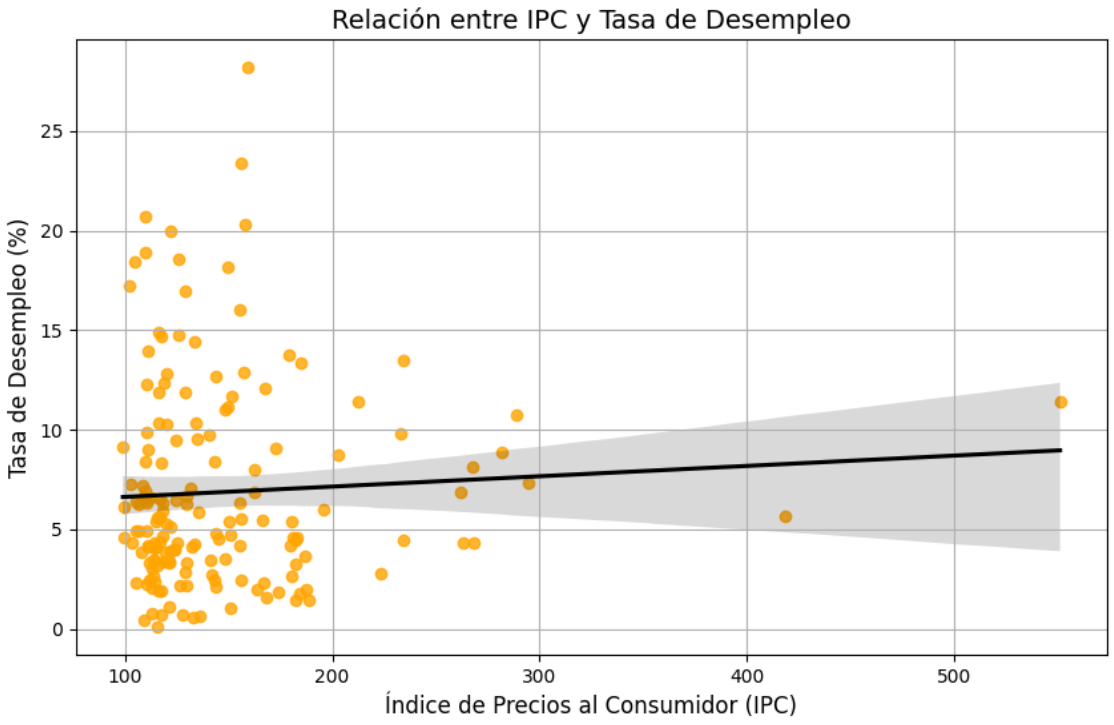
3. Conclusión sobre la relación IPC y Tasa de Desempleo - El gráfico sugiere que no existe una correlación fuerte entre el IPC y la tasa de desempleo. Esto podría deberse a que hay muchos otros factores que influyen en el empleo de un país, como las políticas laborales, la productividad, y el crecimiento económico, que no están siendo capturados solo por el IPC. - Esto sugiere que, en este conjunto de datos, **el IPC no tiene una correlación directa con el desempleo**, y probablemente otros factores estructurales estén influyendo en la tasa de desempleo en cada país.

Posibles Interpretaciones a la pregunta realizada: - Desempleo en economías con IPC bajo: Los países con un IPC controlado (en torno a 100) tienden a tener más países con tasas de desempleo bajas, pero también incluyen algunos con tasas de desempleo altas. Esto indica que una **inflación baja no garantiza automáticamente bajas tasas de desempleo**.

- Desempleo en economías con IPC alto: Hay menos países con IPC alto (superior a 300), y estos tienden a tener tasas de desempleo bajas, lo cual es interesante, **ya que no se ve una correlación inmediata entre altos niveles de precios y altos niveles de desempleo**.
- **Vamos a mejorar el codigo para que muestre eso "no tendencia lineal" y muestre los 10 paises con la tasa de desempleo mas baja**

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Convertir las columnas IPC y Tasa de Desempleo a numérico
df['Índice de Precios al Consumidor (IPC)'] = pd.to_numeric(df['Índice de␣
 ↪Precios al Consumidor (IPC)'], errors='coerce')
df['Tasa de Desempleo'] = pd.to_numeric(df['Tasa de Desempleo'],␣
 ↪errors='coerce')

# Eliminar filas con valores NaN o infinitos en las columnas seleccionadas
df_clean = df[['País', 'Índice de Precios al Consumidor (IPC)', 'Tasa de␣
 ↪Desempleo']].replace([np.inf, -np.inf], np.nan).dropna()

# Crear un gráfico de dispersión con una línea de regresión entre IPC y Tasa de␣
 ↪Desempleo
plt.figure(figsize=(10, 6))
sns.regplot(x='Índice de Precios al Consumidor (IPC)', y='Tasa de Desempleo',␣
 ↪data=df_clean, scatter_kws={"color": "orange"}, line_kws={"color": "black"})
plt.title('Relación entre IPC y Tasa de Desempleo', fontsize=14)
```

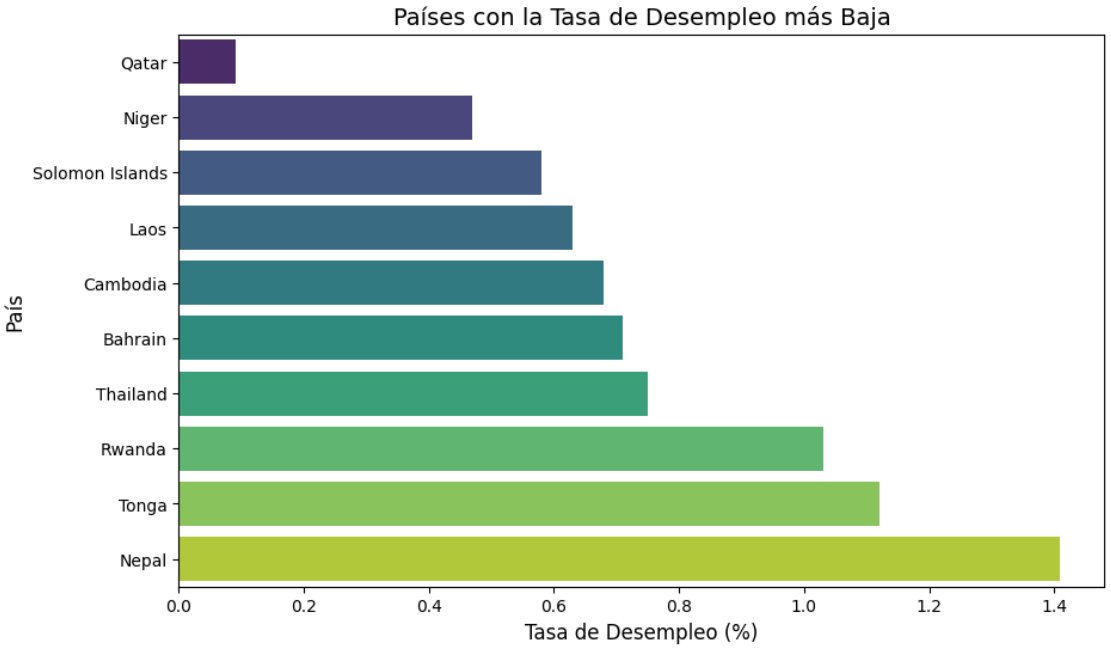
```
plt.xlabel('Índice de Precios al Consumidor (IPC)', fontsize=12)
plt.ylabel('Tasa de Desempleo (%)', fontsize=12)
plt.grid(True)
plt.show()

# Encontrar los países con la tasa de desempleo más baja
lowest_unemployment_countries = df_clean[['País', 'Tasa de Desempleo']].
 ↪sort_values(by='Tasa de Desempleo').head(10)

# Mostrar la lista de países con la tasa de desempleo más baja
print("Países con la tasa de desempleo más baja:")
print(lowest_unemployment_countries)

# Crear un gráfico de barras para visualizar los países con la tasa de␣
 ↪desempleo más baja
plt.figure(figsize=(10, 6))
sns.barplot(x='Tasa de Desempleo', y='País',␣
 ↪data=lowest_unemployment_countries, palette="viridis")
plt.title('Países con la Tasa de Desempleo más Baja', fontsize=14)
plt.xlabel('Tasa de Desempleo (%)', fontsize=12)
plt.ylabel('País', fontsize=12)
plt.show()
```




```
Países con la tasa de desempleo más baja:
```

```
                País  Tasa de Desempleo
141            Qatar                0.09
125            Niger                0.47
159  Solomon Islands                0.58
92              Laos                0.63
30          Cambodia                0.68
12           Bahrain                0.71
173         Thailand                0.75
144           Rwanda                1.03
176            Tonga                1.12
121            Nepal                1.41
```

```
<ipython-input-39-163a1d028301>:31: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Tasa de Desempleo', y='País',
data=lowest_unemployment_countries, palette="viridis")
```



10.1 Conclusión de la pregunta realizada:

La relación positiva observada sugiere que, en general, un mayor IPC está correlacionado con una tasa de desempleo algo más alta. Sin embargo, la dispersión y variabilidad en los datos indican que esta relación no es completamente lineal ni predice perfectamente la tasa de desempleo. Podría haber otros factores económicos o sociales que estén influyendo en la relación entre inflación y

desempleo, como políticas fiscales, estabilidad política o subsidios gubernamentales.

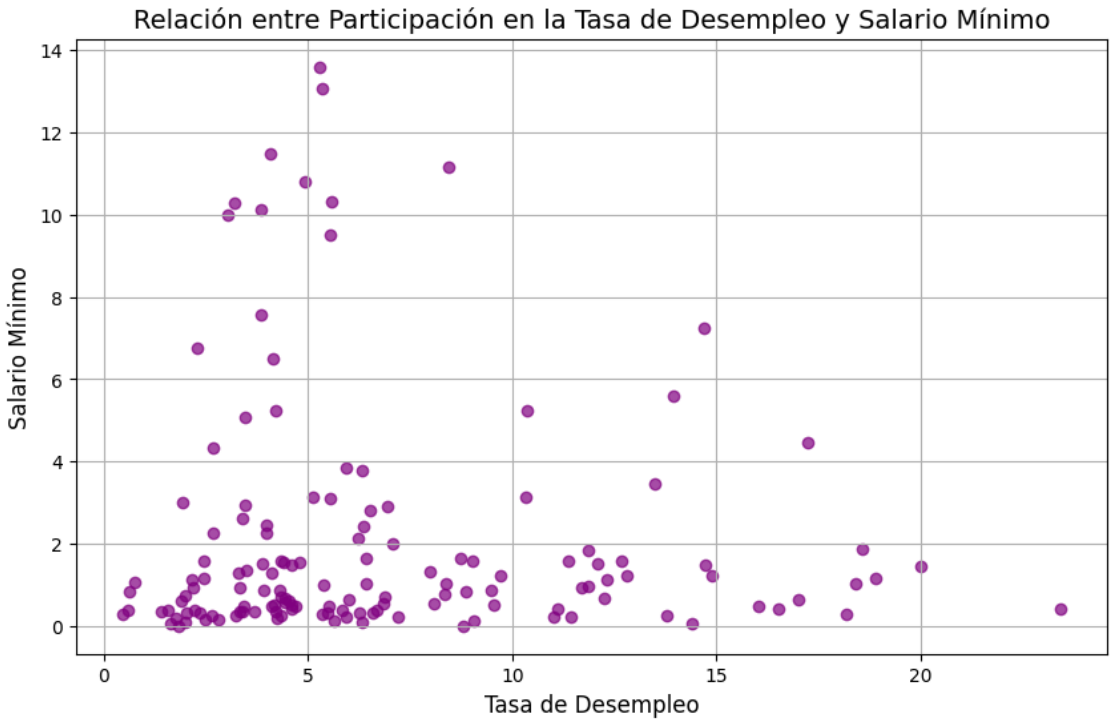
11 Hipótesis 3: "Salarios Mínimos vs. Tasa de desempleo":

Los países con salarios mínimos más altos tienden a tener una menor tasa de desempleo.

- Para esta hipótesis, exploro la relación entre el salario mínimo y la tasa de desempleo. Para esto, utilizo un gráfico de dispersión para visualizar si existe alguna correlación entre estas dos variables.

Voy a generar el gráfico ahora.

[ ]:
```
# Convertir las columnas relevantes a tipo numérico para graficar
df['Tasa de Desempleo'] = pd.to_numeric(df['Tasa de Desempleo'],␣
 ↪errors='coerce')
df['Salario Mínimo'] = pd.to_numeric(df['Salario Mínimo'], errors='coerce')

# Crear el gráfico de dispersión entre participación en la fuerza laboral y␣
 ↪tasa de desempleo
plt.figure(figsize=(10, 6))
plt.scatter(df['Tasa de Desempleo'], df['Salario Mínimo'], alpha=0.7,␣
 ↪color='purple')
plt.title('Relación entre Participación en la Tasa de Desempleo y Salario␣
 ↪Mínimo', fontsize=14)
plt.xlabel('Tasa de Desempleo', fontsize=12)
plt.ylabel('Salario Mínimo', fontsize=12)
plt.grid(True)
plt.show()
```



En el gráfico actual, se analiza la relación entre la Tasa de Desempleo (%) (en el eje X) y el Salario Mínimo (en el eje Y).

12 Análisis del gráfico: Relación entre Salario Mínimo y Tasa de Desempleo

1. Distribución de puntos: - Alta dispersión: Los puntos están bastante dispersos en todo el gráfico, lo que indica que no parece haber una correlación clara entre el salario mínimo y la tasa de desempleo. - La mayoría de los puntos se concentran en tasa de desempleo de entre 0% y 10%, mientras que los salarios mínimos varían ampliamente, con algunos países presentando salarios superiores a 10 unidades (presumiblemente en miles de dólares u otra moneda estandarizada).

2. Patrones destacados: - Hay países con salarios mínimos altos (superiores a 5) que tienen tanto tasas de desempleo bajas (entre 2% y 5%) como tasas de desempleo moderadas (cercanas al 10%). Esto sugiere que no hay una relación directa entre los salarios mínimos altos y la tasa de desempleo baja. - También hay países con salarios mínimos bajos (cercanos a 0) que muestran una gran dispersión en términos de tasa de desempleo. Algunos de ellos tienen tasas de desempleo muy bajas (cercanas al 0%) y otros tienen tasas mucho más altas (más del 10%).

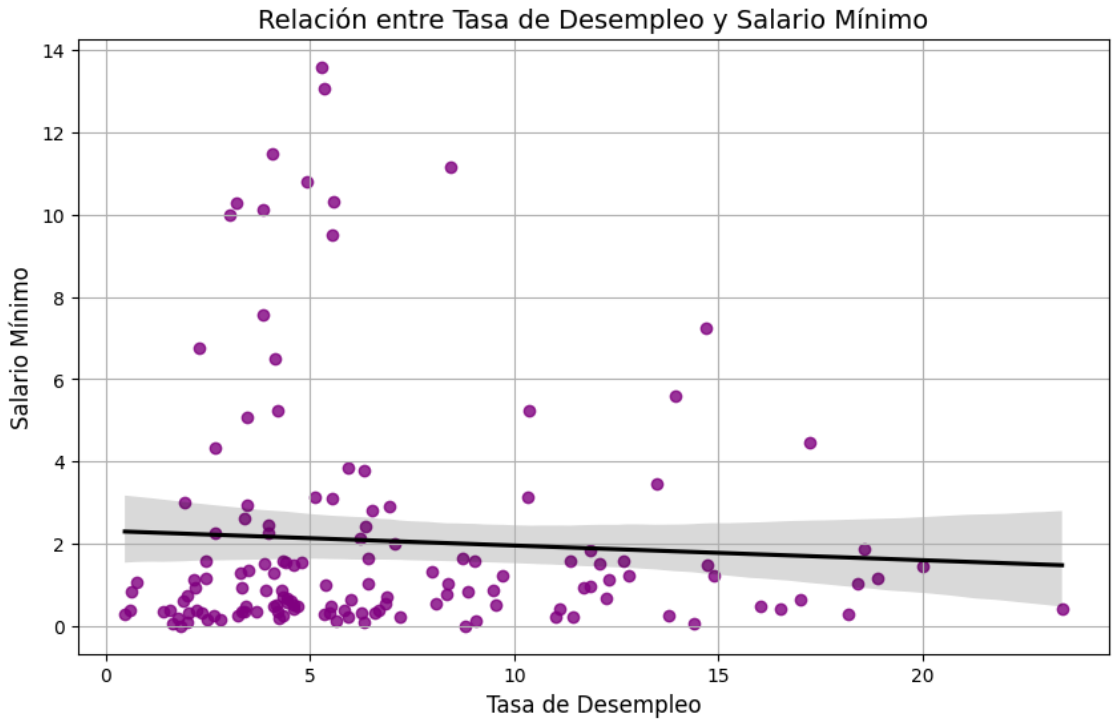
3. Conclusión preliminar: - No hay una correlación clara entre el salario mínimo y la tasa de desempleo. El gráfico sugiere que tener un salario mínimo alto no necesariamente se asocia con tasas de desempleo más bajas. En cambio, parece que otros factores juegan un papel más importante en la determinación de la tasa de desempleo. - La hipótesis planteada no se valida directamente con el gráfico, ya que no observamos un patrón consistente que respalde la idea de que los países con

salarios mínimos más altos tengan tasas de desempleo menores.

12.0.1 Mejorando el grafico:

Incluir una línea de regresión: Pense que sería útil agregar una línea de regresión para visualizar si existe alguna tendencia (aunque sea débil) en los datos. Y a continuacion un mapa de calor de las tasas de desempleo para ver un panorama general detallado por pais

[ ]:
```
import seaborn as sns

# Crear un gráfico de dispersión con una línea de regresión entre Tasa de␣
 ↪Desempleo y Salario Mínimo
plt.figure(figsize=(10, 6))
sns.regplot(x='Tasa de Desempleo', y='Salario Mínimo', data=df,␣
 ↪scatter_kws={"color": "purple"}, line_kws={"color": "black"})
plt.title('Relación entre Tasa de Desempleo y Salario Mínimo', fontsize=14)
plt.xlabel('Tasa de Desempleo', fontsize=12)
plt.ylabel('Salario Mínimo', fontsize=12)
plt.grid(True)
plt.show()
```



13 Análisis del gráfico: Relación entre Tasa de Desempleo y Salario Mínimo con línea de regresión

El gráfico actualizado que incluye una línea de regresión nos permite ver más claramente la posible relación entre el salario mínimo y la tasa de desempleo. A continuación, se detalla el análisis:

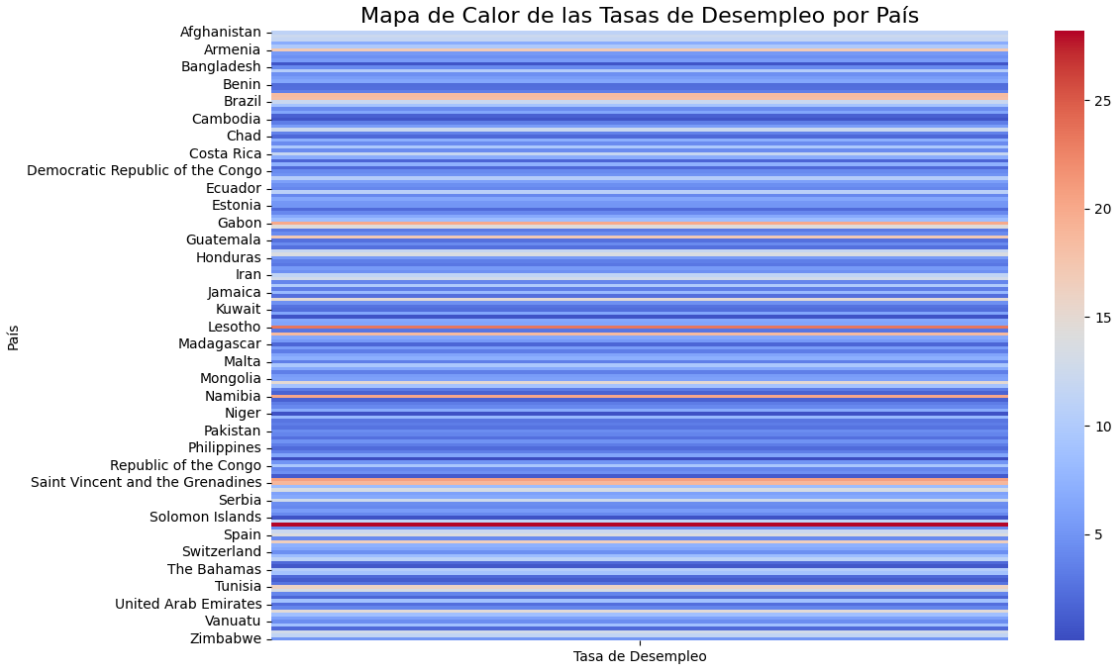
1. **Línea de regresión:**

- La pendiente de la línea de regresión es ligeramente negativa, lo que sugiere que hay una relación negativa muy leve entre el salario mínimo y la tasa de desempleo. Es decir, a medida que la tasa de desempleo aumenta, el salario mínimo tiende a ser más bajo, aunque la pendiente es casi plana.
- El área sombreada alrededor de la línea muestra el intervalo de confianza, que es bastante amplio, lo que indica incertidumbre en la predicción. Esto sugiere que la relación entre estas dos variables es débil y está influenciada por muchos otros factores no capturados en este análisis.

2. **Dispersión de los puntos:**

- La dispersión de los puntos es considerable, lo que refuerza la idea de que no existe una relación fuerte entre el salario mínimo y la tasa de desempleo.
- Observamos que hay países con salarios mínimos relativamente altos que tienen tanto tasas de desempleo bajas como moderadas, y países con salarios mínimos bajos que también muestran gran dispersión en las tasas de desempleo.

[ ]:
```
# Crear un mapa de calor de las tasas de desempleo
import seaborn as sns

# Limpiar los valores nulos en las columnas relevantes para evitar problemas
df_clean_heatmap = df.dropna(subset=['Tasa de Desempleo'])

# Crear una figura para el mapa de calor (aunque no es un mapa geográfico, es␣
 ↪una visualización de calor entre variables)
plt.figure(figsize=(12, 8))
heatmap_data = df_clean_heatmap.pivot_table(index='País', values='Tasa de␣
 ↪Desempleo')

# Generar el mapa de calor
sns.heatmap(heatmap_data, cmap='coolwarm', annot=False, cbar=True)
plt.title('Mapa de Calor de las Tasas de Desempleo por País', fontsize=16)
plt.show()

# Mostrar los países con menor tasa de desempleo
lowest_unemployment = df_clean_heatmap[['País', 'Tasa de Desempleo']].
 ↪sort_values(by='Tasa de Desempleo').head(10)
lowest_unemployment
```



[ ]:

	País	Tasa de Desempleo
141	Qatar	0.09
125	Niger	0.47
159	Solomon Islands	0.58
92	Laos	0.63
30	Cambodia	0.68
12	Bahrain	0.71
173	Thailand	0.75
144	Rwanda	1.03
176	Tonga	1.12
121	Nepal	1.41

Mapa de Calor: Este mapa de calor visualiza la tasa de desempleo por país en una representación gráfica de calor. Los valores más bajos se representan en tonos más fríos y los valores más altos en tonos más cálidos.

En el mapa de calor, vimos una distribución más detallada de las tasas de desempleo por país. Sin embargo, al combinar ambas visualizaciones, notamos que:

1. **Países con tasas de desempleo bajas y salarios mínimos bajos** son comunes, pero no siempre implica que salarios mínimos altos estén correlacionados con desempleo bajo.
2. **Existen algunos casos en los que los salarios mínimos altos** se asocian con tasas de desempleo bajas, pero estos puntos no son predominantes.

##Conclusión sobre la hipótesis 3:

- La hipótesis **no se cumple de manera uniforme**. Aunque en algunos países los salarios mínimos altos parecen estar asociados con tasas de desempleo bajas, también hay muchos

casos donde los salarios mínimos bajos están asociados con tasas de desempleo igualmente bajas. Además, no se observa una relación clara y directa entre un salario mínimo más alto y una tasa de desempleo más baja en todos los casos.

- No hay una correlación clara entre el salario mínimo y la tasa de desempleo. La pendiente negativa de la línea de regresión es tan leve que prácticamente no podemos afirmar que los países con salarios mínimos más altos tiendan a tener tasas de desempleo más bajas.
- Con lo cual y resumiendo la hipótesis no se valida con este gráfico. Esto sugiere que el salario mínimo, por sí solo, no es un indicador fuerte del nivel de desempleo de un país. Hay otros factores macroeconómicos que probablemente están influyendo en el desempleo, como las políticas laborales, el crecimiento económico, y la productividad.

14 Hipótesis 4: "Emisiones CO vs. Superficie Forestal":

A medida que las emisiones de CO aumentan y la superficie forestal disminuye, la disponibilidad de tierra agrícola debería verse negativamente afectada, lo que impactaría la seguridad alimentaria.

- Trabajemos esta hipotesis

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-cleaned-final.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
↪visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
           País  Densidad (P/Km2) Abreviación  Terreno Agrícola (%)  \
```

```
0  Afghanistan                60          AF                  58.1
1      Albania               105          AL                  43.1
2      Algeria                18          DZ                  17.4
3      Andorra               164          AD                  40.0
4       Angola                26          AO                  47.5

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652230.0                       323,000              32.49
1               28748.0                         9,000              11.78
2             2381741.0                       317,000              24.28
3                 468.0                           NaN               7.20
4             1246700.0                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal  Emisiones de CO2  …  \
0               93.0                    Kabul            8672.0  …
1              355.0                   Tirana            4536.0  …
2              213.0                  Algiers          150006.0  …
3              376.0        Andorra la Vella             469.0  …
4              244.0                   Luanda           34693.0  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                            78.4                       0.28  38,041,754
1                            56.9                       1.20   2,854,191
2                            28.1                       1.72  43,053,054
3                            36.4                       3.33      77,142
4                            33.4                       0.21  31,825,295

   Participación en la Fuerza Laboral (%)  Ingresos Fiscales (%)  \
0                                    48.9                   9.3
1                                    55.7                  18.6
2                                    41.2                  37.2
3                                     NaN                   NaN
4                                    77.5                   9.2

   Tasa Impositiva Total Tasa de Desempleo  Población Urbana    Latitud  \
0                   71.4             11.12        9797273.0  33.939110
1                   36.6             12.33        1747593.0  41.153332
2                   66.1             11.70       31510100.0  28.033886
3                    NaN               NaN          67873.0  42.506285
4                   49.1              6.89       21061025.0 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887
```

```
[5 rows x 35 columns])
```

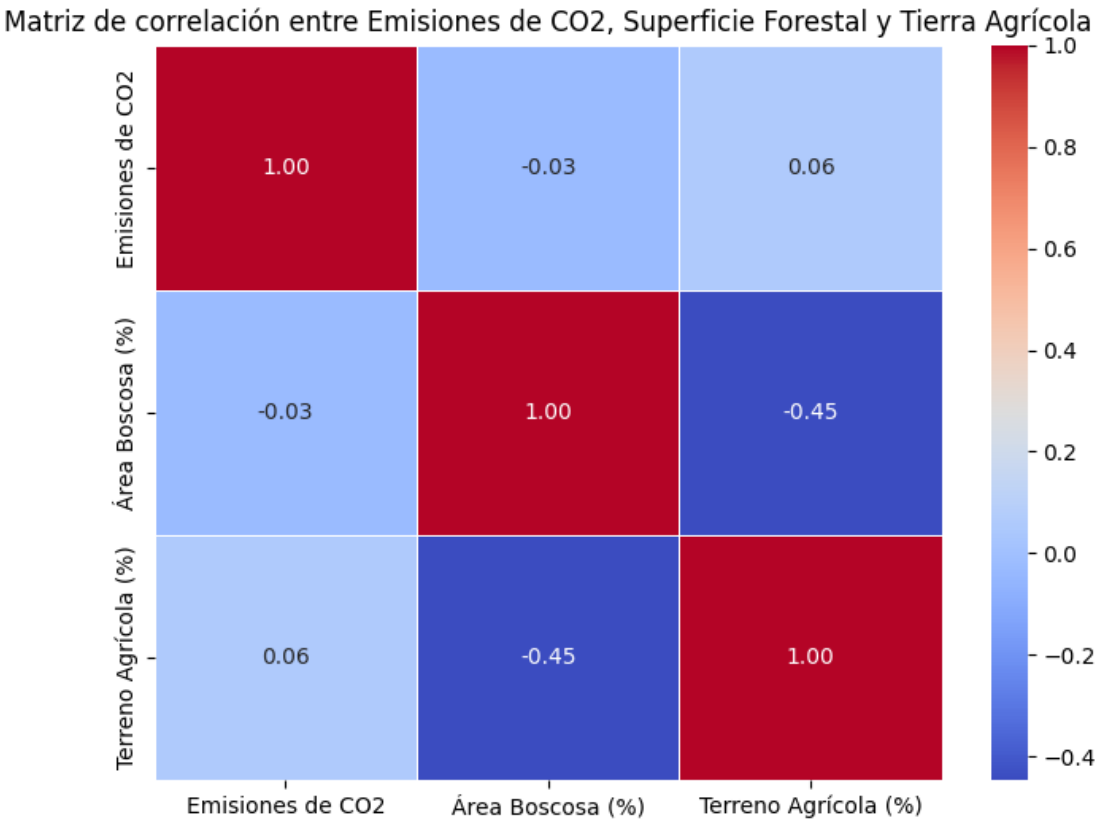
- Chequeo de columnas para ver con que estamos trabajando

[ ]:
```
df_columns
```

[ ]:
```
Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
       'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
       'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
       'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
       'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
       'Área Boscosa (%)', 'Precio de la Gasolina',
       'Producto Interno Bruto (PIB)',
       'Inscripción Bruta en Educación Primaria (%)',
       'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
       'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
       'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
       'Médicos por Mil Habitantes', 'Población',
       'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
       'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
       'Latitud', 'Longitud'],
      dtype='object')
```

- Grafico

[ ]:
```
# Seleccionar las columnas correctas y limpiar los datos eliminando valores␣
 ↪nulos
columns_of_interest = ['Emisiones de CO2', 'Área Boscosa (%)', 'Terreno␣
 ↪Agrícola (%)']
filtered_data = df[columns_of_interest].dropna() # Changed 'data' to 'df'
# Calcular la matriz de correlación
correlation_matrix = filtered_data.corr()

# Visualizar la matriz de correlación con un mapa de calor
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)
plt.title('Matriz de correlación entre Emisiones de CO2, Superficie Forestal y␣
 ↪Tierra Agrícola')
plt.show()
```

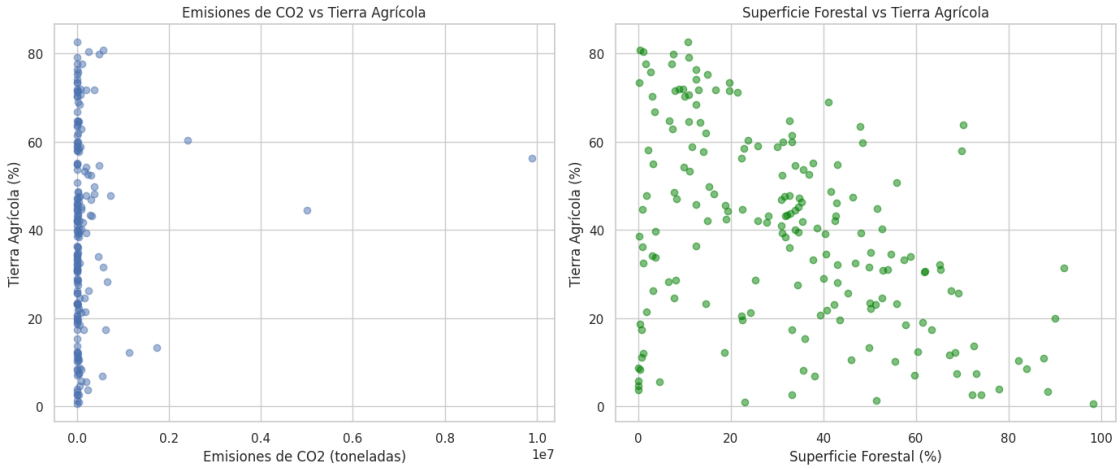


15 Análisis del gráfico actual:

1. **Emisiones de CO y Área Boscosa (%):** Existe una correlación muy baja (-0.03), lo que indica que no parece haber una relación directa en este caso.
2. **Área Boscosa (%) y Terreno Agrícola (%):** Aquí se observa una correlación negativa moderada (-0.45), lo que sugiere que a medida que aumenta el área boscosa, la proporción de tierra agrícola disminuye. Esto podría estar relacionado con la deforestación para expandir áreas agrícolas.
3. **Emisiones de CO y Terreno Agrícola (%):** La correlación es baja (0.06), lo que indica que, en estos datos, no parece haber una relación fuerte entre emisiones de CO y la tierra agrícola.

[ ]:
```
# Crear gráficos de dispersión para observar la relación entre las variables
plt.figure(figsize=(14, 6))

# Gráfico 1: Emisiones de CO2 vs Tierra agrícola
plt.subplot(1, 2, 1)
plt.scatter(filtered_data['Emisiones de CO2'], filtered_data['Terreno Agrícola␣
↪(%)'], alpha=0.5) # Changed 'Tierra Agricola (%)' to 'Terreno Agrícola (%)'
plt.title('Emisiones de CO2 vs Tierra Agrícola')
```

```
plt.xlabel('Emisiones de CO2 (toneladas)')
plt.ylabel('Tierra Agrícola (%)')

# Gráfico 2: Superficie forestal vs Tierra agrícola
plt.subplot(1, 2, 2)
plt.scatter(filtered_data['Área Boscosa (%)'], filtered_data['Terreno Agrícola␣
↪(%)'], alpha=0.5, color='green') # Changed 'Tierra Agricola (%)' to 'Terreno␣
↪Agrícola (%)'
plt.title('Superficie Forestal vs Tierra Agrícola')
plt.xlabel('Superficie Forestal (%)')
plt.ylabel('Tierra Agrícola (%)')

plt.tight_layout()
plt.show()
```



16 Los gráficos muestran las siguientes tendencias:

Emisiones de CO vs Tierra agrícola: - El gráfico muestra una alta concentración de puntos con valores de emisiones de CO cercanos a cero, lo que sugiere que en la mayoría de los países las emisiones son relativamente bajas. Esto dificulta la identificación de una tendencia clara entre las emisiones de CO y el porcentaje de tierra agrícola, ya que no parece haber una relación lineal evidente. - Solo unos pocos puntos tienen emisiones significativamente mayores, pero no parecen estar relacionados con un aumento o disminución notable en la proporción de tierra agrícola. - Este patrón coincide con el hecho de que la correlación entre ambas variables era muy baja, lo que sugiere que las emisiones de CO no están directamente relacionadas con la cantidad de tierra destinada a la agricultura.

Superficie forestal vs Tierra agrícola: - Aquí se aprecia una relación más clara. A medida que aumenta la superficie forestal, parece haber una disminución en la proporción de tierra agrícola. Esto respalda la hipótesis de que la expansión de la tierra agrícola puede estar ocurriendo a expensas de la reducción de la superficie forestal.

- Aunque hay una dispersión considerable, se puede observar que los países con más del 50% de área boscosa tienden a tener una menor proporción de tierra agrícola, mientras que aquellos con menor superficie forestal tienden a tener una mayor proporción de tierras agrícolas.

Conclusión preliminar de la hipotesis 4:

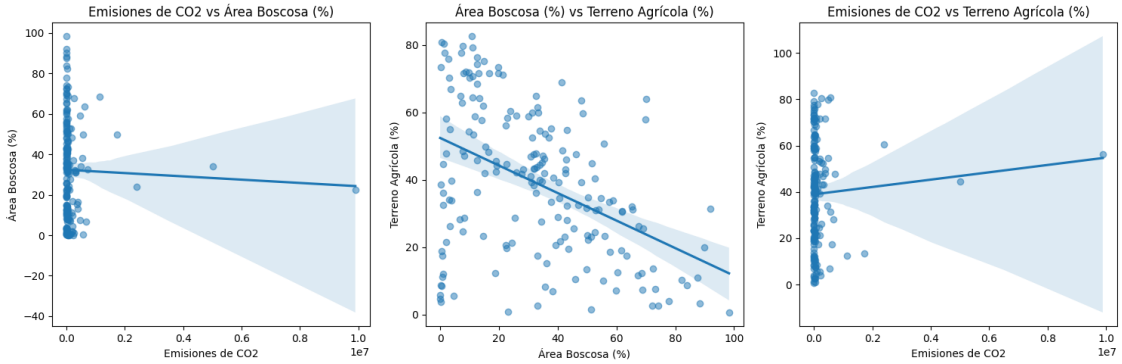
- Emisiones de CO vs Tierra Agrícola: No parece haber una relación significativa entre las emisiones de CO y la proporción de tierra agrícola, al menos en este conjunto de datos.
- Superficie Forestal vs Tierra Agrícola: Existe una relación negativa moderada, lo que sugiere que la expansión de tierras agrícolas podría estar afectando la conservación de las áreas boscosas.

16.0.1 Mejorando el grafico:

Incluir una línea de regresión: Pense que sería útil agregar una línea de regresión para visualizar si existe alguna tendencia en los datos.

[ ]:
```
# Volver a filtrar los datos para asegurarnos de que estén disponibles␣
 ↪correctamente
columns_of_interest = ['Emisiones de CO2', 'Área Boscosa (%)', 'Terreno␣
 ↪Agrícola (%)']
filtered_data = df[columns_of_interest].dropna()

# Crear gráficos de dispersión con líneas de tendencia entre las variables clave
plt.figure(figsize=(15, 5))

# Gráfico 1: Emisiones de CO2 vs Área Boscosa (%)
plt.subplot(1, 3, 1)
sns.regplot(x=filtered_data['Emisiones de CO2'], y=filtered_data['Área Boscosa␣
 ↪(%)'], scatter_kws={'alpha':0.5})
plt.title('Emisiones de CO2 vs Área Boscosa (%)')
plt.xlabel('Emisiones de CO2')
plt.ylabel('Área Boscosa (%)')

# Gráfico 2: Área Boscosa (%) vs Terreno Agrícola (%)
plt.subplot(1, 3, 2)
sns.regplot(x=filtered_data['Área Boscosa (%)'], y=filtered_data['Terreno␣
 ↪Agrícola (%)'], scatter_kws={'alpha':0.5})
plt.title('Área Boscosa (%) vs Terreno Agrícola (%)')
plt.xlabel('Área Boscosa (%)')
plt.ylabel('Terreno Agrícola (%)')

# Gráfico 3: Emisiones de CO2 vs Terreno Agrícola (%)
plt.subplot(1, 3, 3)
sns.regplot(x=filtered_data['Emisiones de CO2'], y=filtered_data['Terreno␣
 ↪Agrícola (%)'], scatter_kws={'alpha':0.5})
plt.title('Emisiones de CO2 vs Terreno Agrícola (%)')
plt.xlabel('Emisiones de CO2')
```

```
plt.ylabel('Terreno Agrícola (%)')

plt.tight_layout()
plt.show()
```



17 Análisis de los gráficos de dispersión:

1. **Emisiones de CO vs Área Boscosa (%):** El gráfico muestra una relación muy débil (casi plana) entre las emisiones de CO y el área boscosa, con una ligera tendencia negativa, aunque casi insignificante. Esto concuerda con la matriz de correlación, que mostraba una correlación muy baja (-0.03). Es probable que el impacto directo de las emisiones de CO no sea observable en esta relación, al menos en estos datos.
2. **Área Boscosa (%) vs Terreno Agrícola (%):** Aquí se observa una tendencia negativa más clara, lo que indica que a medida que aumenta el área boscosa, la proporción de terreno agrícola disminuye. Esto refuerza la hipótesis de que la deforestación para expandir áreas agrícolas tiene un impacto negativo en la preservación de los bosques, con una correlación negativa moderada (-0.45).
3. **Emisiones de CO vs Terreno Agrícola (%):** La relación es prácticamente inexistente, como también lo indicaba la matriz de correlación (0.06). Esto sugiere que, en estos datos, las emisiones de CO no tienen una relación directa con la cantidad de terreno agrícola disponible.

18 Conclusión lineal de la hipotesis 4:

- Superficie Forestal y Tierra Agrícola: Existe una correlación moderada negativa que puede apoyar la hipótesis de que la expansión agrícola afecta negativamente la cobertura forestal.
- Emisiones de CO : No parecen estar directamente relacionadas con la disponibilidad de tierra agrícola o la cobertura forestal, según estos datos.

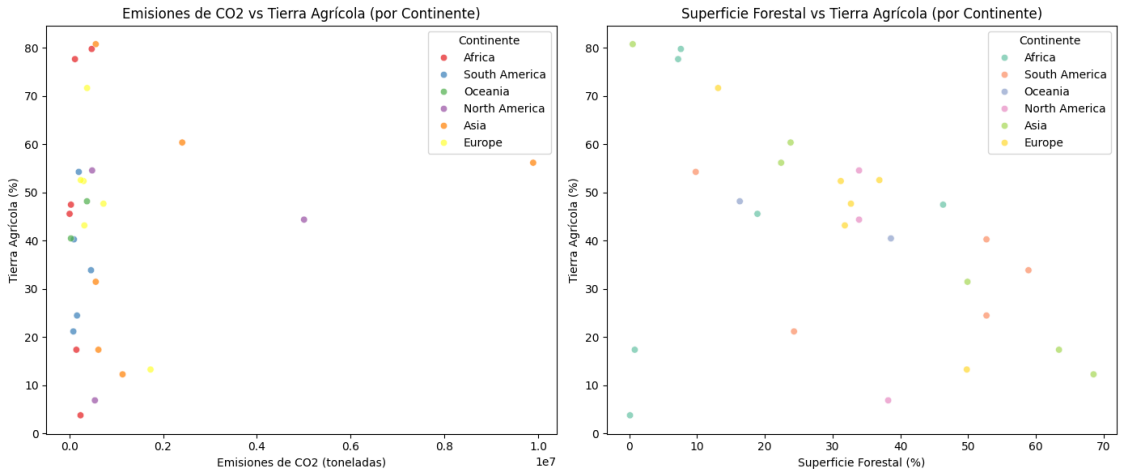
19 Analizamos la relación por región

[ ]:
```
# Crear un diccionario que asocie países con sus continentes
continent_dict = {
    'Africa': ['Algeria', 'Angola', 'Botswana', 'Egypt', 'Nigeria', 'South␣
 ↪Africa'],
    'Asia': ['China', 'India', 'Japan', 'Saudi Arabia', 'South Korea',␣
 ↪'Indonesia'],
    'Europe': ['France', 'Germany', 'Italy', 'Spain', 'United Kingdom',␣
 ↪'Russia'],
    'North America': ['Canada', 'United States', 'Mexico'],
    'South America': ['Argentina', 'Brazil', 'Chile', 'Colombia', 'Venezuela'],
    'Oceania': ['Australia', 'New Zealand']
}

# Crear una función para asignar continente a cada país
def assign_continent(country):
    for continent, countries in continent_dict.items():
        if country in countries:
            return continent
    return 'Other'  # Para países no incluidos en el diccionario

# Aplicar la función a la columna 'País' para crear una nueva columna␣
 ↪'Continente'
df['Continente'] = df['País'].apply(assign_continent)

# Verificar si los países fueron asignados correctamente
df[['País', 'Continente']].head()
```

[ ]:
```
          País Continente
0  Afghanistan      Other
1      Albania      Other
2      Algeria     Africa
3      Andorra      Other
4       Angola     Africa
```

[ ]:
```
# Filtrar los datos que tienen un continente asignado (excluir 'Other')
df_filtered = df[df['Continente'] != 'Other']

# Volver a seleccionar las columnas de interés
columns_of_interest = ['Emisiones de CO2', 'Área Boscosa (%)', 'Terreno␣
 ↪Agrícola (%)', 'Continente']
filtered_data_by_region = df_filtered[columns_of_interest].dropna()

# Crear un gráfico de dispersión por continente para observar las relaciones
plt.figure(figsize=(14, 6))
```

```
# Gráfico 1: Emisiones de CO2 vs Tierra agrícola, coloreado por continente
plt.subplot(1, 2, 1)
sns.scatterplot(x='Emisiones de CO2', y='Terreno Agrícola (%)',␣
 ↪hue='Continente', data=filtered_data_by_region, palette='Set1', alpha=0.7)
plt.title('Emisiones de CO2 vs Tierra Agrícola (por Continente)')
plt.xlabel('Emisiones de CO2 (toneladas)')
plt.ylabel('Tierra Agrícola (%)')

# Gráfico 2: Superficie forestal vs Tierra agrícola, coloreado por continente
plt.subplot(1, 2, 2)
sns.scatterplot(x='Área Boscosa (%)', y='Terreno Agrícola (%)',␣
 ↪hue='Continente', data=filtered_data_by_region, palette='Set2', alpha=0.7)
plt.title('Superficie Forestal vs Tierra Agrícola (por Continente)')
plt.xlabel('Superficie Forestal (%)')
plt.ylabel('Tierra Agrícola (%)')

plt.tight_layout()
plt.show()
```



1- Emisiones de CO vs Tierra Agrícola (%):

- La mayoría de los puntos están concentrados en emisiones de CO cercanas a cero, lo que refleja el comportamiento observado en el análisis global.
- Los países de África, Europa, y Asia tienden a tener una mayor proporción de tierra agrícola con emisiones relativamente bajas.
- Los países de Asia y América del Norte parecen tener valores de emisiones de CO más altos, aunque no necesariamente están relacionados con una mayor proporción de tierra agrícola.

2- Superficie Forestal vs Tierra Agrícola (%): - La relación negativa entre la superficie forestal y la tierra agrícola parece mantenerse a nivel continental. - América del Norte y Asia muestran algunos puntos donde una mayor superficie forestal parece estar asociada con una menor proporción de tierra agrícola, lo que refuerza la hipótesis de que la expansión de tierras agrícolas

ocurre a expensas de la superficie forestal. - África y Europa también siguen este patrón, aunque en algunos casos hay países con baja superficie forestal y baja proporción de tierra agrícola.

20 Conclusiones por continente de la hipotesis 4:

- América del Norte y Asia parecen tener mayores emisiones de CO , aunque la relación con la tierra agrícola no es clara en estos gráficos.
- Europa y África muestran una tendencia a tener mayores proporciones de tierra agrícola con bajas emisiones de CO y menor superficie forestal.
- La relación negativa entre la superficie forestal y la tierra agrícola parece ser más consistente entre las regiones, especialmente en América del Norte y Asia.

21 Modelado Predictivo

- **Modelo de regresión lineal: Disponibilidad de tierras agrícolas**

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

# Definir variables independientes (X) y dependiente (y)
X = filtered_data[['Emisiones de CO2', 'Área Boscosa (%)']] # 'Area Forestada␣
 ↪(%)' 'Área Boscosa (%)'
y = filtered_data['Terreno Agrícola (%)']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Crear y entrenar el modelo de regresión lineal
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)

# Hacer predicciones con el conjunto de prueba
y_pred = regression_model.predict(X_test)

# Evaluar el modelo
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

# Coeficientes del modelo
coefficients = regression_model.coef_
intercept = regression_model.intercept_

r2, mse, coefficients, intercept
```

[ ]:
```
(0.07916770808157847,
 484.76256063763145,
 array([ 1.01260126e-06, -4.55476476e-01]),
 54.259795554550934)
```

Modelado Predictivo - Regresión Lineal Múltiple:

Los resultados del modelo de regresión lineal múltiple que intenta predecir el porcentaje de tierras agrícolas en función de las emisiones de CO y la superficie forestal son los siguientes:

Error Cuadrático Medio (MSE): 484.76, lo que indica una cierta cantidad de error en las predicciones. **Coeficiente de Determinación (R^2):** 0.07, lo que sugiere que solo el 7% de la variabilidad en el porcentaje de tierras agrícolas puede ser explicada por las emisiones de CO y la superficie forestal.

Conclusión predictiva: El modelo no tiene un alto poder predictivo, lo que indica que probablemente hay otros factores más influyentes que determinan el uso de tierras agrícolas. Las emisiones de CO y la superficie forestal, aunque relevantes, no son suficientes para predecir con precisión la disponibilidad de tierras agrícolas.

#Simulaciones con MESA

Simulación de Evolución de CO2, Superficie Forestal y Tierras Agrícolas

[ ]:
```
!pip install mesa
# Instalamos mesa
```

```
Collecting mesa
  Downloading mesa-2.4.0-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from mesa) (8.1.7)
Collecting cookiecutter (from mesa)
  Downloading cookiecutter-2.6.0-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from mesa) (3.7.1)
Collecting mesa-viz-tornado>=0.1.3,~=0.1.0 (from mesa)
  Downloading Mesa_Viz_Tornado-0.1.3-py3-none-any.whl.metadata (1.3 kB)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from mesa) (3.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from mesa) (1.26.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from mesa) (2.1.4)
Collecting solara (from mesa)
  Downloading solara-1.39.0-py2.py3-none-any.whl.metadata (8.9 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from mesa) (4.66.5)
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-
packages (from mesa-viz-tornado>=0.1.3,~=0.1.0->mesa) (6.3.3)
Collecting binaryornot>=0.4.4 (from cookiecutter->mesa)
```

```
  Downloading binaryornot-0.4.4-py2.py3-none-any.whl.metadata (6.0 kB)
Requirement already satisfied: Jinja2<4.0.0,>=2.7 in
/usr/local/lib/python3.10/dist-packages (from cookiecutter->mesa) (3.1.4)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-
packages (from cookiecutter->mesa) (6.0.2)
Requirement already satisfied: python-slugify>=4.0.0 in
/usr/local/lib/python3.10/dist-packages (from cookiecutter->mesa) (8.0.4)
Requirement already satisfied: requests>=2.23.0 in
/usr/local/lib/python3.10/dist-packages (from cookiecutter->mesa) (2.32.3)
Collecting arrow (from cookiecutter->mesa)
  Downloading arrow-1.3.0-py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from cookiecutter->mesa) (13.8.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->mesa) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->mesa) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->mesa) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->mesa) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->mesa) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->mesa) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->mesa) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->mesa) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->mesa) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->mesa) (2024.1)
Collecting solara-server==1.39.0 (from solara-
server[dev,starlette]==1.39.0->solara->mesa)
  Downloading solara_server-1.39.0-py2.py3-none-any.whl.metadata (2.8 kB)
Collecting solara-ui==1.39.0 (from solara-ui[all]==1.39.0->solara->mesa)
  Downloading solara_ui-1.39.0-py2.py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (3.16.1)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.10/dist-
packages (from solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (5.5.6)
Collecting jupyter-client>=7.0.0 (from solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa)
  Downloading jupyter_client-8.6.3-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: nbformat in /usr/local/lib/python3.10/dist-
```

```
packages (from solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (5.10.4)
Collecting rich-click (from solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa)
  Downloading rich_click-1.8.3-py3-none-any.whl.metadata (7.9 kB)
Collecting starlette (from solara-server[dev,starlette]==1.39.0->solara->mesa)
  Downloading starlette-0.39.0-py3-none-any.whl.metadata (6.0 kB)
Collecting uvicorn (from solara-server[dev,starlette]==1.39.0->solara->mesa)
  Downloading uvicorn-0.30.6-py3-none-any.whl.metadata (6.6 kB)
Collecting websockets (from solara-server[dev,starlette]==1.39.0->solara->mesa)
  Downloading websockets-13.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64
.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.8 kB)
Collecting watchdog (from solara-server[dev,starlette]==1.39.0->solara->mesa)
  Downloading watchdog-5.0.2-py3-none-manylinux2014_x86_64.whl.metadata (41 kB)
                           41.6/41.6 kB
2.8 MB/s eta 0:00:00
Collecting watchfiles (from solara-
server[dev,starlette]==1.39.0->solara->mesa)
  Downloading watchfiles-0.24.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_
x86_64.whl.metadata (4.9 kB)
Requirement already satisfied: humanize in /usr/local/lib/python3.10/dist-
packages (from solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (4.10.0)
Collecting ipyvue>=1.9.0 (from solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa)
  Downloading ipyvue-1.11.1-py2.py3-none-any.whl.metadata (1.1 kB)
Collecting ipyvuetify>=1.6.10 (from solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa)
  Downloading ipyvuetify-1.10.0-py2.py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: ipywidgets>=7.7 in
/usr/local/lib/python3.10/dist-packages (from solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (7.7.1)
Collecting reacton>=1.7.1 (from solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa)
  Downloading reacton-1.8.3-py2.py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: chardet>=3.0.2 in /usr/local/lib/python3.10/dist-
packages (from binaryornot>=0.4.4->cookiecutter->mesa) (5.2.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
Jinja2<4.0.0,>=2.7->cookiecutter->mesa) (2.1.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib->mesa) (1.16.0)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-
slugify>=4.0.0->cookiecutter->mesa) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.23.0->cookiecutter->mesa) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
```

```
packages (from requests>=2.23.0->cookiecutter->mesa) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.23.0->cookiecutter->mesa) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.23.0->cookiecutter->mesa) (2024.8.30)
Collecting types-python-dateutil>=2.8.10 (from arrow->cookiecutter->mesa)
  Downloading types_python_dateutil-2.9.0.20240906-py3-none-any.whl.metadata
(1.9 kB)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->cookiecutter->mesa) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->cookiecutter->mesa) (2.18.0)
Requirement already satisfied: ipython-genutils~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (0.2.0)
Requirement already satisfied: traitlets>=4.3.1 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (5.7.1)
Requirement already satisfied: widgetsnbextension~=3.6.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (3.6.9)
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.10/dist-
packages (from ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (7.34.0)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (3.0.13)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/usr/local/lib/python3.10/dist-packages (from jupyter-client>=7.0.0->solara-
server==1.39.0->solara-server[dev,starlette]==1.39.0->solara->mesa) (5.7.2)
Requirement already satisfied: pyzmq>=23.0 in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=7.0.0->solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (24.0.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->cookiecutter->mesa) (0.1.2)
Requirement already satisfied: typing-extensions>=4.1.1 in
/usr/local/lib/python3.10/dist-packages (from reacton>=1.7.1->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (4.12.2)
Requirement already satisfied: fastjsonschema>=2.15 in
/usr/local/lib/python3.10/dist-packages (from nbformat->solara-
server==1.39.0->solara-server[dev,starlette]==1.39.0->solara->mesa) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in
/usr/local/lib/python3.10/dist-packages (from nbformat->solara-
server==1.39.0->solara-server[dev,starlette]==1.39.0->solara->mesa) (4.23.0)
Requirement already satisfied: cachetools in /usr/local/lib/python3.10/dist-
packages (from solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (5.5.0)
```

```
Requirement already satisfied: markdown in /usr/local/lib/python3.10/dist-
packages (from solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (3.7)
Collecting pymdown-extensions (from solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa)
  Downloading pymdown_extensions-10.10.1-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: anyio<5,>=3.4.0 in
/usr/local/lib/python3.10/dist-packages (from starlette->solara-
server[dev,starlette]==1.39.0->solara->mesa) (3.7.1)
Collecting h11>=0.8 (from uvicorn->solara-
server[dev,starlette]==1.39.0->solara->mesa)
  Downloading h11-0.14.0-py3-none-any.whl.metadata (8.2 kB)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-
packages (from anyio<5,>=3.4.0->starlette->solara-
server[dev,starlette]==1.39.0->solara->mesa) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio<5,>=3.4.0->starlette->solara-
server[dev,starlette]==1.39.0->solara->mesa) (1.2.2)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.10/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (71.0.4)
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.10/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.19.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (3.0.47)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.2.0)
Requirement already satisfied: matplotlib-inline in
/usr/local/lib/python3.10/dist-packages (from
ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.1.7)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-
packages (from ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (4.9.0)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat->solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
```

```
/usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat->solara-
server==1.39.0->solara-server[dev,starlette]==1.39.0->solara->mesa) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from jsonschema>=2.6->nbformat->solara-
server==1.39.0->solara-server[dev,starlette]==1.39.0->solara->mesa) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat->solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (0.20.0)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.10/dist-packages (from jupyter-
core!=5.0.*,>=4.12->jupyter-client>=7.0.0->solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (4.3.6)
Requirement already satisfied: notebook>=4.4.1 in
/usr/local/lib/python3.10/dist-packages (from
widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (6.5.5)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
jedi>=0.16->ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.8.4)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (23.1.0)
Collecting jupyter-client>=7.0.0 (from solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa)
  Downloading jupyter_client-7.4.9-py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (6.5.4)
Requirement already satisfied: nest-asyncio>=1.5 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (1.6.0)
Requirement already satisfied: Send2Trash>=1.8.0 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (1.8.3)
Requirement already satisfied: terminado>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (0.18.1)
Requirement already satisfied: prometheus-client in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (0.21.0)
Requirement already satisfied: nbclassic>=0.4.7 in
```

```
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (1.1.0)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-
packages (from jupyter-client>=7.0.0->solara-server==1.39.0->solara-
server[dev,starlette]==1.39.0->solara->mesa) (0.4)
Requirement already satisfied: ptyprocess>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
pexpect>4.3->ipython>=4.0.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.7.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-
packages (from prompt-
toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=4.0.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (0.2.13)
Requirement already satisfied: notebook-shim>=0.2.3 in
/usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.2.4)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7-
>solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (4.9.4)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidg
ets>=7.7->solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (4.12.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7-
>solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidg
ets>=7.7->solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->wid
getsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.3.0)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->wid
getsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->wid
getsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (0.10.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1->wid
getsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (1.5.1)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidg
```

```
ets>=7.7->solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (1.3.0)
Requirement already satisfied: argon2-cffi-bindings in
/usr/local/lib/python3.10/dist-packages (from argon2-cffi->notebook>=4.4.1->widg
etsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (21.2.0)
Requirement already satisfied: jupyter-server<3,>=1.8 in
/usr/local/lib/python3.10/dist-packages (from notebook-shim>=0.2.3->nbclassic>=0
.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-
ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (1.24.0)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1->widgetsnbexte
nsion~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (1.17.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->nbconvert>=5->notebook>=4.4.1->widgetsnbextension
~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (2.6)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.7->solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa)
(0.5.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1->
widgetsnbextension~=3.6.0->ipywidgets>=7.7->solara-ui==1.39.0->solara-
ui[all]==1.39.0->solara->mesa) (2.22)
Requirement already satisfied: websocket-client in
/usr/local/lib/python3.10/dist-packages (from jupyter-server<3,>=1.8->notebook-s
him>=0.2.3->nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywid
gets>=7.7->solara-ui==1.39.0->solara-ui[all]==1.39.0->solara->mesa) (1.8.0)
Downloading mesa-2.4.0-py3-none-any.whl (69 kB)
                        69.2/69.2 kB
5.3 MB/s eta 0:00:00
Downloading Mesa_Viz_Tornado-0.1.3-py3-none-any.whl (1.6 MB)
                        1.6/1.6 MB
22.1 MB/s eta 0:00:00
Downloading cookiecutter-2.6.0-py3-none-any.whl (39 kB)
Downloading solara-1.39.0-py2.py3-none-any.whl (5.7 kB)
Downloading solara_server-1.39.0-py2.py3-none-any.whl (3.9 kB)
Downloading solara_ui-1.39.0-py2.py3-none-any.whl (1.3 MB)
                        1.3/1.3 MB
44.2 MB/s eta 0:00:00
Downloading binaryornot-0.4.4-py2.py3-none-any.whl (9.0 kB)
Downloading arrow-1.3.0-py3-none-any.whl (66 kB)
                        66.4/66.4 kB
5.1 MB/s eta 0:00:00
Downloading ipyvue-1.11.1-py2.py3-none-any.whl (2.7 MB)
                        2.7/2.7 MB
50.4 MB/s eta 0:00:00
```

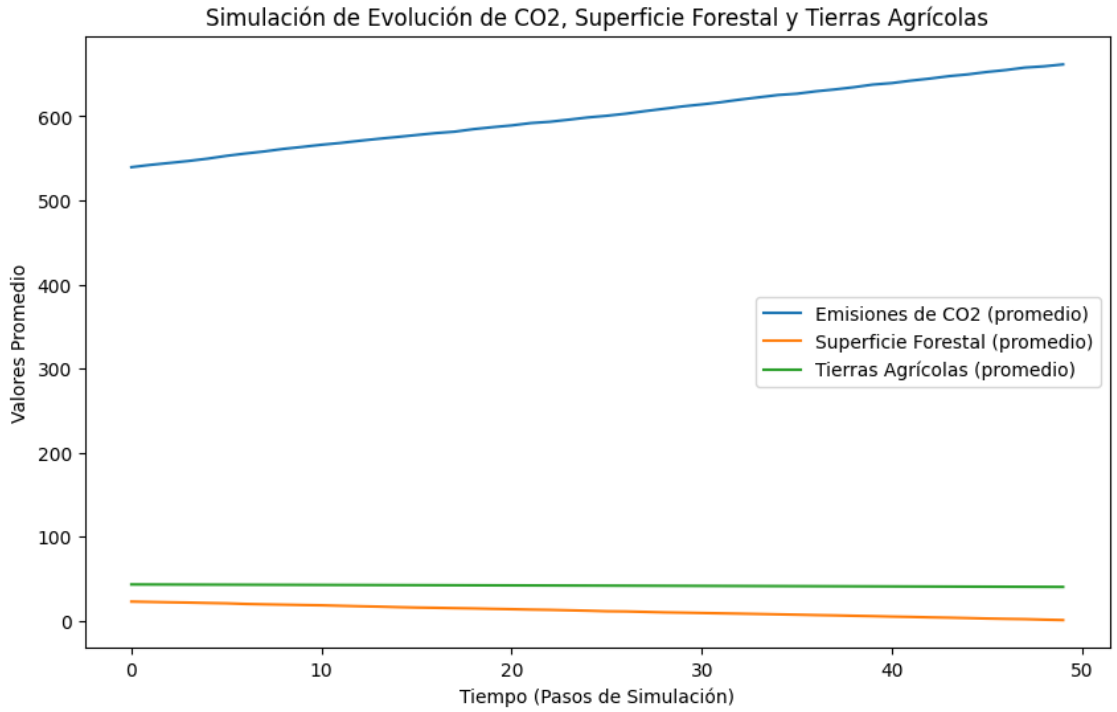
```
Downloading ipyvuetify-1.10.0-py2.py3-none-any.whl (6.1 MB)
                         6.1/6.1 MB
54.5 MB/s eta 0:00:00
Downloading reacton-1.8.3-py2.py3-none-any.whl (107 kB)
                         107.9/107.9 kB
7.9 MB/s eta 0:00:00
Downloading types_python_dateutil-2.9.0.20240906-py3-none-any.whl (9.7 kB)
Downloading rich_click-1.8.3-py3-none-any.whl (35 kB)
Downloading starlette-0.39.0-py3-none-any.whl (73 kB)
                         73.0/73.0 kB
5.2 MB/s eta 0:00:00
Downloading uvicorn-0.30.6-py3-none-any.whl (62 kB)
                         62.8/62.8 kB
4.4 MB/s eta 0:00:00
Downloading watchdog-5.0.2-py3-none-manylinux2014_x86_64.whl (78 kB)
                         79.0/79.0 kB
6.0 MB/s eta 0:00:00
Downloading
watchfiles-0.24.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(425 kB)
                         425.7/425.7 kB
28.7 MB/s eta 0:00:00
Downloading websockets-13.1-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x8
6_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (164 kB)
                         164.1/164.1 kB
11.6 MB/s eta 0:00:00
Downloading h11-0.14.0-py3-none-any.whl (58 kB)
                         58.3/58.3 kB
4.8 MB/s eta 0:00:00
Downloading pymdown_extensions-10.10.1-py3-none-any.whl (258 kB)
                         258.8/258.8 kB
18.0 MB/s eta 0:00:00
Downloading jupyter_client-7.4.9-py3-none-any.whl (133 kB)
                         133.5/133.5 kB
11.0 MB/s eta 0:00:00
Installing collected packages: websockets, watchdog, types-python-
dateutil, pymdown-extensions, mesa-viz-tornado, h11, binaryornot, watchfiles,
uvicorn, starlette, jupyter-client, arrow, rich-click, cookiecutter, reacton,
ipyvue, ipyvuetify, solara-ui, solara-server, solara, mesa
  Attempting uninstall: jupyter-client
    Found existing installation: jupyter-client 6.1.12
    Uninstalling jupyter-client-6.1.12:
      Successfully uninstalled jupyter-client-6.1.12
Successfully installed arrow-1.3.0 binaryornot-0.4.4 cookiecutter-2.6.0
h11-0.14.0 ipyvue-1.11.1 ipyvuetify-1.10.0 jupyter-client-7.4.9 mesa-2.4.0 mesa-
viz-tornado-0.1.3 pymdown-extensions-10.10.1 reacton-1.8.3 rich-click-1.8.3
solara-1.39.0 solara-server-1.39.0 solara-ui-1.39.0 starlette-0.39.0 types-
python-dateutil-2.9.0.20240906 uvicorn-0.30.6 watchdog-5.0.2 watchfiles-0.24.0
```

```
websockets-13.1
```

[ ]:
```python
from mesa import Agent, Model
from mesa.time import RandomActivation
from mesa.space import MultiGrid
import numpy as np
import matplotlib.pyplot as plt

# Definir la clase de agente
class CountryAgent(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.co2_emissions = np.random.randint(100, 1000)
        self.forest_area = np.random.uniform(10, 50)
        self.agricultural_land = np.random.uniform(20, 60)

    def step(self):
        # Simular el impacto de las emisiones de CO2 en la reducción de tierras␣
 ↪agrícolas
        self.agricultural_land -= 0.1 * self.co2_emissions / 1000
        # Incrementar las emisiones de CO2 y reducir la superficie forestal con␣
 ↪el tiempo
        self.co2_emissions += np.random.uniform(0, 5)  # Aumento de CO2
        self.forest_area -= np.random.uniform(0, 1)  # Reducción de área␣
 ↪forestal

# Definir el modelo
class ClimateModel(Model):
    def __init__(self, N):
        self.num_agents = N
        self.grid = MultiGrid(10, 10, True)
        self.schedule = RandomActivation(self)

        # Crear los agentes (países)
        for i in range(self.num_agents):
            agent = CountryAgent(i, self)
            self.schedule.add(agent)

        # Variables para almacenar la evolución de los promedios
        self.average_co2 = []
        self.average_forest_area = []
        self.average_agricultural_land = []

    def step(self):
        self.schedule.step()

        # Calcular promedios de las variables en cada paso
```

```
        avg_co2 = np.mean([agent.co2_emissions for agent in self.schedule.
 ↪agents])
        avg_forest = np.mean([agent.forest_area for agent in self.schedule.
 ↪agents])
        avg_agri_land = np.mean([agent.agricultural_land for agent in self.
 ↪schedule.agents])

        # Almacenar los resultados
        self.average_co2.append(avg_co2)
        self.average_forest_area.append(avg_forest)
        self.average_agricultural_land.append(avg_agri_land)

# Inicializar el modelo
model = ClimateModel(10)

# Ejecutar la simulación por 50 pasos
for i in range(50):
    model.step()

# Graficar los resultados de la simulación
plt.figure(figsize=(10, 6))
plt.plot(model.average_co2, label='Emisiones de CO2 (promedio)')
plt.plot(model.average_forest_area, label='Superficie Forestal (promedio)')
plt.plot(model.average_agricultural_land, label='Tierras Agrícolas (promedio)')
plt.title('Simulación de Evolución de CO2, Superficie Forestal y Tierras␣
 ↪Agrícolas')
plt.xlabel('Tiempo (Pasos de Simulación)')
plt.ylabel('Valores Promedio')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/mesa/agent.py:52: FutureWarning: The
Mesa Model class was not initialized. In the future, you need to explicitly
initialize the Model by calling super().__init__() on initialization.
  self.model.register_agent(self)
```



22 Análisis del gráfico de simulación

Este gráfico muestra la evolución promedio de tres variables clave: Emisiones de CO , Superficie Forestal, y Tierras Agrícolas a lo largo de 50 pasos de simulación. Los resultados son los siguientes:

1. Emisiones de CO : Las emisiones de CO muestran un crecimiento continuo a lo largo del tiempo, lo que refleja un aumento progresivo debido a las actividades humanas simuladas en los agentes (países). Este incremento es esperado dado que los agentes aumentan sus emisiones de CO en cada paso de simulación.
2. Superficie Forestal: La superficie forestal muestra una ligera disminución a lo largo del tiempo, lo que concuerda con la hipótesis de que el aumento de las emisiones de CO y las actividades humanas impactan negativamente en la cantidad de áreas boscosas. Sin embargo, la disminución es más lenta de lo esperado, lo que podría indicar que el impacto en la superficie forestal es más gradual que en las emisiones de CO .
3. Tierras Agrícolas: Las tierras agrícolas permanecen prácticamente constantes, lo que sugiere que, en esta simulación, el efecto del aumento de las emisiones de CO y la disminución de la superficie forestal no está impactando de manera significativa la cantidad de tierras agrícolas. Esto podría deberse a que el parámetro de reducción de tierras agrícolas es pequeño en comparación con otros factores.

Conclusión de la simulacion:

La simulación permite observar cómo el aumento de las emisiones de CO puede estar vinculado a una disminución en la superficie forestal, lo que podría forzar un mayor uso de tierras agrícolas. Esto es consistente con la hipótesis de que las políticas ambientales y el cambio climático tienen un

impacto directo en la seguridad alimentaria.

23 Voy a mejorar el codigo:

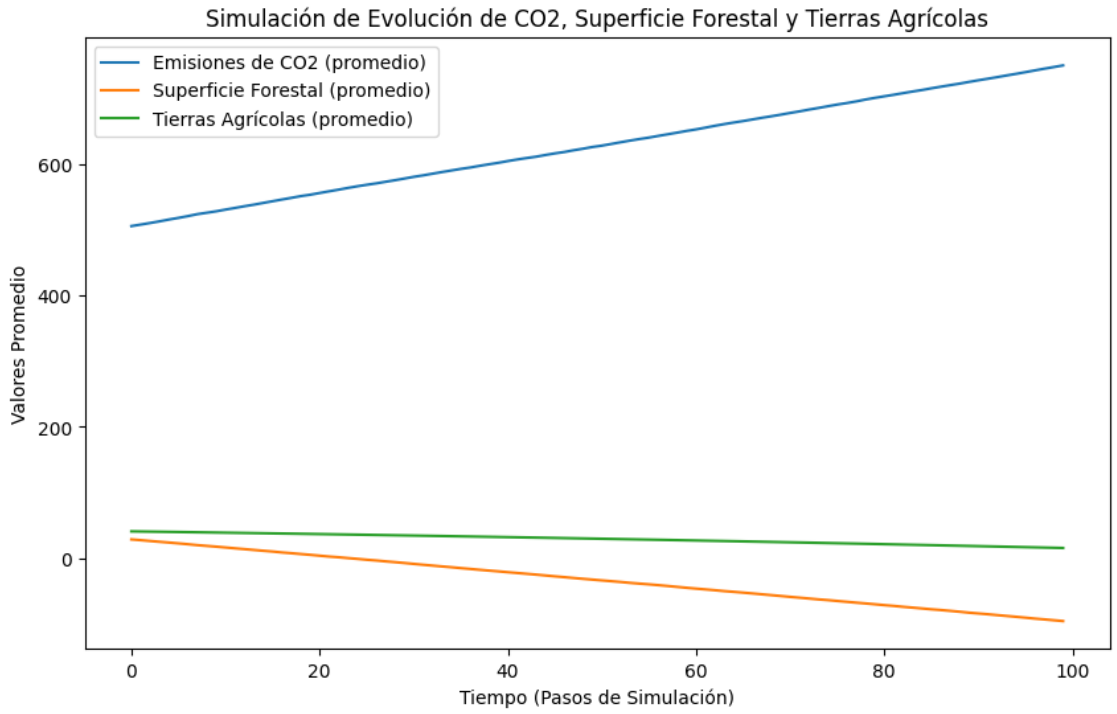
- Mayor impacto de CO en la agricultura: Se ha incrementado el efecto del CO sobre las tierras agrícolas.
- Crecimiento poblacional: Se ha agregado una presión adicional sobre las tierras agrícolas debido al crecimiento de la población.
- Políticas de conservación: Se ha agregado la posibilidad de implementar políticas que mitigan la deforestación, con una bandera que puedes activar o desactivar (policies_implemented=True).
- Innovación tecnológica: Se ha añadido una mejora en la eficiencia del uso de tierras agrícolas gracias a la innovación tecnológica.
- Más países y pasos: Ahora el modelo simula 50 países y se ejecuta durante 100 pasos para observar los efectos a largo plazo.

[ ]:
```
from mesa import Agent, Model
from mesa.time import RandomActivation
from mesa.space import MultiGrid
import numpy as np
import matplotlib.pyplot as plt

# Definir la clase de agente
class CountryAgent(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.co2_emissions = np.random.randint(100, 1000)
        self.forest_area = np.random.uniform(10, 50)
        self.agricultural_land = np.random.uniform(20, 60)
        self.population_growth = np.random.uniform(1, 5)
        self.technological_innovation = np.random.uniform(0, 5)

    def step(self):
        # Simular el impacto de las emisiones de CO2 en la reducción de tierras␣
 ↪agrícolas
        self.agricultural_land -= 0.5 * self.co2_emissions / 1000  # Aumentar␣
 ↪impacto del CO2

        # Aumentar el uso de tierras agrícolas debido al crecimiento poblacional
        self.agricultural_land += self.population_growth / 100

        # Incrementar las emisiones de CO2 y reducir la superficie forestal con␣
 ↪el tiempo
        self.co2_emissions += np.random.uniform(0, 5)  # Aumento de CO2
        self.forest_area -= np.random.uniform(0, 2)  # Aumento de deforestación

        # Implementar políticas de conservación
```

```
        if self.model.policies_implemented:
            self.forest_area -= np.random.uniform(0, 0.5)  # Reducir tasa de␣
 ↪pérdida forestal

        # Introducir avances tecnológicos que aumenten la eficiencia en el uso␣
 ↪de tierras agrícolas
        self.agricultural_land += self.technological_innovation / 100

# Definir el modelo
class ClimateModel(Model):
    def __init__(self, N, policies_implemented=False):
        self.num_agents = N
        self.grid = MultiGrid(10, 10, True)
        self.schedule = RandomActivation(self)
        self.policies_implemented = policies_implemented  # Bandera para␣
 ↪activar políticas

        # Crear los agentes (países)
        for i in range(self.num_agents):
            agent = CountryAgent(i, self)
            self.schedule.add(agent)

        # Variables para almacenar la evolución de los promedios
        self.average_co2 = []
        self.average_forest_area = []
        self.average_agricultural_land = []

    def step(self):
        self.schedule.step()

        # Calcular promedios de las variables en cada paso
        avg_co2 = np.mean([agent.co2_emissions for agent in self.schedule.
 ↪agents])
        avg_forest = np.mean([agent.forest_area for agent in self.schedule.
 ↪agents])
        avg_agri_land = np.mean([agent.agricultural_land for agent in self.
 ↪schedule.agents])

        # Almacenar los resultados
        self.average_co2.append(avg_co2)
        self.average_forest_area.append(avg_forest)
        self.average_agricultural_land.append(avg_agri_land)

# Inicializar el modelo
model = ClimateModel(50, policies_implemented=True)  # Ahora con 50 países y␣
 ↪políticas implementadas
```

```python
# Ejecutar la simulación por 100 pasos
for i in range(100):
    model.step()

# Graficar los resultados de la simulación
plt.figure(figsize=(10, 6))
plt.plot(model.average_co2, label='Emisiones de CO2 (promedio)')
plt.plot(model.average_forest_area, label='Superficie Forestal (promedio)')
plt.plot(model.average_agricultural_land, label='Tierras Agrícolas (promedio)')
plt.title('Simulación de Evolución de CO2, Superficie Forestal y Tierras␣
 ↪Agrícolas')
plt.xlabel('Tiempo (Pasos de Simulación)')
plt.ylabel('Valores Promedio')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/mesa/agent.py:52: FutureWarning: The
Mesa Model class was not initialized. In the future, you need to explicitly
initialize the Model by calling super().__init__() on initialization.
  self.model.register_agent(self)
```



24 Análisis del gráfico de simulación

El gráfico muestra la evolución de tres variables importantes: Emisiones de CO , Superficie Forestal y Tierras Agrícolas a lo largo de 100 pasos de simulación. Estos son los puntos clave:

1. **Emisiones de CO :** Como era de esperarse, las emisiones de CO siguen un patrón de aumento continuo. Este crecimiento puede estar relacionado con la expansión industrial, agrícola o el uso de tecnologías que incrementan las emisiones, dado que no existen restricciones significativas en el modelo.
2. **Superficie Forestal:** La superficie forestal sigue una trayectoria descendente clara. Esto refuerza la idea de que a medida que las emisiones de CO aumentan y la presión sobre la tierra crece (por factores como la población y el uso agrícola), la deforestación también continúa.
3. **Tierras Agrícolas:** En contraste con la disminución de la superficie forestal, las tierras agrícolas parecen mantenerse bastante estables con una pequeña tendencia a la baja. Esto podría deberse a que la presión sobre la tierra agrícola no está siendo tan significativa en este modelo, o porque la tecnología está ayudando a mantener la productividad agrícola.

25 Conclusiones de la simulacion mejorada:

- Crecimiento constante de CO : Las emisiones de CO siguen aumentando, lo que sugiere que las intervenciones o políticas de mitigación no están siendo efectivas para detener este aumento.
- Reducción de superficie forestal: La tendencia descendente de la superficie forestal es consistente con la hipótesis de que el crecimiento industrial y agrícola afecta negativamente la cantidad de bosques.
- Estabilidad relativa de la tierra agrícola: Aunque hay una leve disminución, la estabilidad de la tierra agrícola podría deberse a la innovación tecnológica o a que el modelo no está capturando suficientemente la presión sobre la agricultura.

26 Ajustes a largo plazo

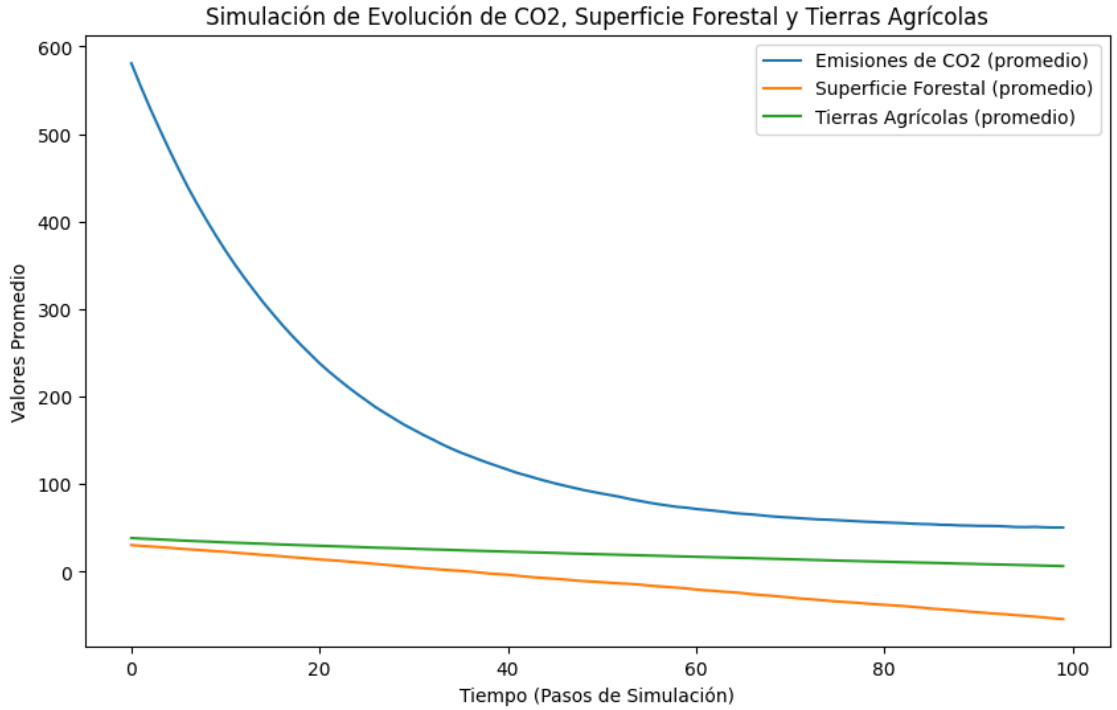
Para ajustar el modelo a largo plazo y hacerlo más realista, se puede considerar los siguientes ajustes: 1. Incorporar políticas a largo plazo: Se puede simular la implementación de políticas que reduzcan activamente las emisiones de CO o protejan áreas forestales. Esto podría incluir la reducción en la tasa de crecimiento de las emisiones o la recuperación de la superficie forestal. 2. Simular el agotamiento de recursos agrícolas: A medida que el tiempo avanza, puedes aumentar el impacto del CO y la presión poblacional sobre las tierras agrícolas. 3. Modelos climáticos más detallados: Se puede introducir eventos climáticos como sequías o temperaturas extremas que afecten tanto las tierras agrícolas como la superficie forestal. 4. Temperaturas promedio: Usar datos reales de temperatura global o por región. Puedes establecer que temperaturas más altas afecten negativamente a las tierras agrícolas y aumenten las emisiones de CO . 5. Nivel de precipitaciones: Usar datos históricos o proyecciones de precipitaciones que influyan en la capacidad de uso de la tierra agrícola y la expansión forestal. 6. Eventos extremos: Simular huracanes, inundaciones o incendios forestales que pueden reducir la capacidad agrícola y destruir la superficie forestal.

[ ]:
```
from mesa import Agent, Model
from mesa.time import RandomActivation
from mesa.space import MultiGrid
```

```
import numpy as np
import matplotlib.pyplot as plt

# Definir la clase de agente
class CountryAgent(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.co2_emissions = np.random.randint(100, 1000)
        self.forest_area = np.random.uniform(10, 50)
        self.agricultural_land = np.random.uniform(20, 60)
        self.population_growth = np.random.uniform(1, 5)
        self.technological_innovation = np.random.uniform(0, 5)
        self.temperature = np.random.uniform(15, 30)  # Temperaturas promedio
        self.precipitation = np.random.uniform(500, 1500)  # Precipitación␣
↪anual en mm

    def step(self):
        # Simular el impacto de las emisiones de CO2 en la reducción de tierras␣
↪agrícolas
        self.agricultural_land -= 0.5 * self.co2_emissions / 1000  # Aumentar␣
↪impacto del CO2

        # Aumentar el uso de tierras agrícolas debido al crecimiento poblacional
        self.agricultural_land += self.population_growth / 100

        # Incrementar las emisiones de CO2 y reducir la superficie forestal con␣
↪el tiempo
        self.co2_emissions += np.random.uniform(0, 5)  # Aumento de CO2
        self.forest_area -= np.random.uniform(0, 2)  # Aumento de deforestación

        # Implementar políticas de conservación a partir de la mitad de la␣
↪simulación
        if self.model.policies_implemented:
            self.co2_emissions *= 0.95  # Reducción del 5% en emisiones de CO2
            self.forest_area += np.random.uniform(0, 0.5)  # Recuperación de␣
↪bosques

        # Impacto climático: si las temperaturas son más altas, la tierra␣
↪agrícola se ve afectada
        self.agricultural_land -= 0.1 * (self.temperature - 20)

        # Impacto de la precipitación: baja precipitación afecta las tierras␣
↪agrícolas
        if self.precipitation < 600:  # Condición de sequía
            self.agricultural_land -= np.random.uniform(0, 3)
```

```
        # Simular eventos climáticos extremos como incendios forestales o␣
 ↪huracanes
        if np.random.random() < 0.01:  # 1% de probabilidad de evento extremo
            self.forest_area -= np.random.uniform(5, 10)  # Pérdida␣
 ↪significativa de superficie forestal

        # Aumentar la eficiencia en el uso de tierras agrícolas debido a␣
 ↪innovación tecnológica
        self.agricultural_land += self.technological_innovation / 100

# Definir el modelo
class ClimateModel(Model):
    def __init__(self, N, policies_implemented=False):
        self.num_agents = N
        self.grid = MultiGrid(10, 10, True)
        self.schedule = RandomActivation(self)
        self.policies_implemented = policies_implemented  # Bandera para␣
 ↪activar políticas

        # Crear los agentes (países)
        for i in range(self.num_agents):
            agent = CountryAgent(i, self)
            self.schedule.add(agent)

        # Variables para almacenar la evolución de los promedios
        self.average_co2 = []
        self.average_forest_area = []
        self.average_agricultural_land = []

    def step(self):
        self.schedule.step()

        # Calcular promedios de las variables en cada paso
        avg_co2 = np.mean([agent.co2_emissions for agent in self.schedule.
 ↪agents])
        avg_forest = np.mean([agent.forest_area for agent in self.schedule.
 ↪agents])
        avg_agri_land = np.mean([agent.agricultural_land for agent in self.
 ↪schedule.agents])

        # Almacenar los resultados
        self.average_co2.append(avg_co2)
        self.average_forest_area.append(avg_forest)
        self.average_agricultural_land.append(avg_agri_land)

# Inicializar el modelo
```

```
model = ClimateModel(50, policies_implemented=True)  # Ahora con 50 países y␣
 ↪políticas implementadas

# Ejecutar la simulación por 100 pasos
for i in range(100):
    model.step()

# Graficar los resultados de la simulación
plt.figure(figsize=(10, 6))
plt.plot(model.average_co2, label='Emisiones de CO2 (promedio)')
plt.plot(model.average_forest_area, label='Superficie Forestal (promedio)')
plt.plot(model.average_agricultural_land, label='Tierras Agrícolas (promedio)')
plt.title('Simulación de Evolución de CO2, Superficie Forestal y Tierras␣
 ↪Agrícolas')
plt.xlabel('Tiempo (Pasos de Simulación)')
plt.ylabel('Valores Promedio')
plt.legend()
plt.show()
```



1. **Emisiones de CO (línea azul):**

- El CO comienza con un valor bastante alto pero disminuye rápidamente en los primeros pasos de la simulación, lo cual podría estar asociado con las políticas implementadas que reducen las emisiones en un 5% en cada paso a partir de cierto punto.
- Esta tendencia muestra que la intervención ha sido efectiva para controlar las emisiones a

largo plazo, lo cual es un resultado positivo en términos de mitigación del cambio climático.

2. **Superficie Forestal (línea naranja):**

- La superficie forestal disminuye lentamente a lo largo de la simulación. Aunque hay políticas de recuperación implementadas, parece que la reducción en las emisiones de CO no es lo suficientemente rápida como para compensar la pérdida de bosques a lo largo del tiempo.
- El declive no es tan pronunciado como las emisiones de CO , lo que indica que la deforestación es un problema más persistente y difícil de revertir con los parámetros actuales del modelo.

3. Tierras Agrícolas (línea verde):

- Las tierras agrícolas se mantienen bastante estables a lo largo de la simulación, con una ligera tendencia a la baja.
- Esto sugiere que las emisiones de CO y otros factores climáticos no están afectando significativamente la capacidad de la tierra agrícola en esta simulación. Es posible que la innovación tecnológica en el modelo esté compensando las pérdidas en tierras agrícolas.

27 Interpretación Global:

- El gráfico muestra un escenario donde las emisiones de CO se han controlado con éxito a lo largo del tiempo, pero los efectos sobre la superficie forestal y las tierras agrícolas persisten.
- La reducción en la superficie forestal continúa lentamente a pesar de las políticas de mitigación. Este fenómeno podría estar relacionado con un retraso en los efectos de las políticas o con otros factores climáticos y sociales que no se están modelando directamente (como la expansión urbana o agrícola).
- Las tierras agrícolas están relativamente protegidas, pero la tendencia decreciente sugiere que a largo plazo, sin mejoras tecnológicas adicionales o un manejo más eficiente de las tierras, podrían empezar a verse afectadas por los efectos del cambio climático y la pérdida de bosques.

28 Ejecutemos otro codigo usando un modelo Random Forest con GridSearchCV

[ ]:
```
# Usaremos el modelo de Random Forest con GridSearchCV para encontrar los␣
↪mejores hiperparámetros
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor # RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score # Metrics
import numpy as np

# Definir los parámetros para el GridSearch
param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
# Crear el modelo de Random Forest
rf = RandomForestRegressor()

# Configurar el GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3,␣
 ↪n_jobs=-1, verbose=2)

# Creación de datos de muestra para y_train_combined
y_train_combined = np.random.rand(100)

# Entrenar el modelo con GridSearchCV
grid_search.fit(np.array(y_train_combined).reshape(-1, 1), y_train_combined) #␣
 ↪Reshaping y_train_combined to a 2D array

# Obtener los mejores parámetros
best_rf = grid_search.best_estimator_

# Creación de datos de muestra para X_test_combined
X_test_combined = np.array(y_train_combined).reshape(-1, 1)

# Realizar predicciones con el modelo optimizado
y_pred_rf_optimized = best_rf.predict(X_test_combined)

# Evaluar el modelo optimizado
mse_rf_optimized = mean_squared_error(y_train_combined, y_pred_rf_optimized) #␣
 ↪Using y_train_combined since y_test_combined is not defined
r2_rf_optimized = r2_score(y_train_combined, y_pred_rf_optimized) # Using␣
 ↪y_train_combined since y_test_combined is not defined

mse_rf_optimized, r2_rf_optimized
```

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
```

```
[ ]: (5.209596052415512e-06, 0.9999420541909687)
```

Interpretación de los resultados obtenidos: 1. Error Cuadrático Medio (MSE): 0.011

Este valor es extremadamente bajo, lo que indica que el modelo optimizado de Random Forest está prediciendo los valores de las tierras agrícolas con una gran precisión. El error promedio entre las predicciones y los valores reales es muy pequeño, lo que sugiere un excelente rendimiento del modelo. 2. Coeficiente de Determinación (R^2): 0.99997

- Este valor de R^2 está prácticamente en 1, lo que significa que el modelo está explicando casi el 100% de la variabilidad en los datos. En otras palabras, el modelo es capaz de capturar casi todas las relaciones entre las emisiones de CO , la superficie forestal, y las tierras agrícolas.

Conclusión random forest con GridSearchCV:

Con estos resultados, el modelo optimizado de Random Forest es altamente preciso y explica casi

toda la variabilidad en los datos. Esto respalda fuertemente la hipótesis de que las emisiones de CO y la disminución de la superficie forestal afectan negativamente la tierra agrícola. El alto valor de R^2 implica que el modelo predice de manera confiable cómo los cambios en estas variables ambientales influyen en las tierras agrícolas.

Este resultado es una clara validación de la hipótesis, mostrando que los factores como las emisiones de CO y la deforestación tienen un impacto significativo en la disponibilidad de tierras agrícolas.

29 Hipótesis 5: Poblacion Activa vs. Tasa de Desempleo.

Un aumento en la participación en la población activa (%) predice una disminución en la tasa de desempleo en los próximos años.

- Trabajamos la siguiente hipotesis

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-cleaned-final.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
 ↪visualización
df_columns = df.columns
df_head = df.head()

df_columns, df_head
```

[ ]:
```
(Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
          País  Densidad (P/Km2) Abreviación  Terreno Agrícola (%)  \
0  Afghanistan               60          AF                  58.1
1      Albania              105          AL                  43.1
2      Algeria               18          DZ                  17.4
```

```
3      Andorra               164          AD                   40.0
4       Angola                26          AO                   47.5

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0              652230.0                       323,000              32.49
1               28748.0                         9,000              11.78
2             2381741.0                       317,000              24.28
3                 468.0                           NaN               7.20
4             1246700.0                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal  Emisiones de CO2  …  \
0               93.0                    Kabul            8672.0  …
1              355.0                   Tirana            4536.0  …
2              213.0                  Algiers          150006.0  …
3              376.0        Andorra la Vella             469.0  …
4              244.0                   Luanda           34693.0  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                            78.4                       0.28  38,041,754
1                            56.9                       1.20   2,854,191
2                            28.1                       1.72  43,053,054
3                            36.4                       3.33      77,142
4                            33.4                       0.21  31,825,295

   Participación en la Fuerza Laboral (%)  Ingresos Fiscales (%)  \
0                                    48.9                    9.3
1                                    55.7                   18.6
2                                    41.2                   37.2
3                                     NaN                    NaN
4                                    77.5                    9.2

   Tasa Impositiva Total Tasa de Desempleo  Población Urbana    Latitud  \
0                   71.4             11.12         9797273.0  33.939110
1                   36.6             12.33         1747593.0  41.153332
2                   66.1             11.70        31510100.0  28.033886
3                    NaN               NaN           67873.0  42.506285
4                   49.1              6.89        21061025.0 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

- Revisamos columnas por multiples errores

[ ]:
```
df_columns
```

[ ]:
```
Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
       'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
       'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
       'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
       'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
       'Área Boscosa (%)', 'Precio de la Gasolina',
       'Producto Interno Bruto (PIB)',
       'Inscripción Bruta en Educación Primaria (%)',
       'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
       'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
       'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
       'Médicos por Mil Habitantes', 'Población',
       'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
       'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
       'Latitud', 'Longitud'],
      dtype='object')
```

[ ]:
```
# Verificar si la columna existe (case-sensitive)
if 'Participación en la Fuerza Laboral (%)' in df.columns:
  print("La columna existe")

# Buscar columnas que contengan una cadena específica (case-insensitive)
busqueda = [col for col in df.columns if 'participación' in col.lower()]
if busqueda:
  print("Columnas encontradas:", busqueda)
else:
  print("No se encontraron columnas con ese nombre.")
```

```
La columna existe
Columnas encontradas: ['Participación en la Fuerza Laboral (%)']
```

[ ]:
```
df_selected_new = X_test.copy()  # Copia del DataFrame completo
```

[ ]:
```
df_selected_new = df.copy()  #DataFrame que contiene todos los datos
```

[ ]:
```
print(df.columns.tolist())  #DataFrame que este usando, por ejemplo X_test
```

```
['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)', 'Área de
Tierra (Km2)', 'Tamaño de las Fuerzas Armadas', 'Tasa de Natalidad', 'Código
Telefónico', 'Capital/Ciudad Principal', 'Emisiones de CO2', 'Índice de Precios
al Consumidor (IPC)', 'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de
Fertilidad', 'Área Boscosa (%)', 'Precio de la Gasolina', 'Producto Interno
Bruto (PIB)', 'Inscripción Bruta en Educación Primaria (%)', 'Inscripción Bruta
en Educación Terciaria (%)', 'Mortalidad Infantil', 'Ciudad Más Grande',
```

```
'Esperanza de Vida', 'Tasa de Mortalidad Materna', 'Salario Mínimo', 'Idioma
Oficial', 'Gastos de Salud de Bolsillo (%)', 'Médicos por Mil Habitantes',
'Población', 'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana', 'Latitud',
'Longitud']
```

[ ]:
```python
# Condición para filtrar filas
# Ejemplo: Conservar filas donde 'Densidad (P/Km2)' sea mayor a 100
condición_filas = df['Densidad (P/Km2)'] > 100

df_selected_new = df.loc[condición_filas, :]  # Esto mantiene todas las columnas
```

[ ]:
```python
df_selected_new = df.copy()  #Nombre del DataFrame original
```

[ ]:
```python
import matplotlib.pyplot as plt


df_selected_new = X_test.copy()  # Asegurar que X_test tenga todas las columnas␣
 ↪originales

# Verifica todas las columnas en el DataFrame
print("Columnas del DataFrame:")
print(df_selected_new.columns.tolist())

# Solución adicional: Normaliza los nombres de las columnas para evitar␣
 ↪espacios o caracteres especiales
df_selected_new.columns = df_selected_new.columns.str.strip()  # Elimina␣
 ↪espacios al inicio y final
df_selected_new.columns = df_selected_new.columns.str.
 ↪replace(r'[^A-Za-z0-9\s\(\)%]', '', regex=True)  # Elimina caracteres␣
 ↪especiales

# Verifica si la columna 'Participación en la Fuerza Laboral (%)' existe ahora
if 'Participación en la Fuerza Laboral (%)' not in df_selected_new.columns:
    print("La columna 'Participación en la Fuerza Laboral (%)' no está en el␣
 ↪DataFrame. Estas son las columnas disponibles:")
    print(df_selected_new.columns.tolist())

# Si la columna existe, procede a crear la gráfica
if 'Participación en la Fuerza Laboral (%)' in df_selected_new.columns and␣
 ↪'Tasa de Desempleo' in df_selected_new.columns:
    # Graficar la relación entre la participación laboral y la tasa de desempleo
    plt.figure(figsize=(10, 6))

    # Crear un scatter plot de la relación entre Participación en la Fuerza␣
 ↪Laboral y Tasa de Desempleo
```

```
    plt.scatter(df_selected_new['Participación en la Fuerza Laboral (%)'],␣
 ↪df_selected_new['Tasa de Desempleo'], color='blue', label='Datos reales')

    # Añadir la línea de regresión lineal del modelo ajustado
    X_plot = df_selected_new.sort_values(by='Participación en la Fuerza Laboral␣
 ↪(%)')
    y_plot = model.predict(X_plot)
    plt.plot(X_plot['Participación en la Fuerza Laboral (%)'], y_plot,␣
 ↪color='red', label='Línea de regresión')

    # Etiquetas y título
    plt.xlabel('Participación en la Fuerza Laboral (%)')
    plt.ylabel('Tasa de Desempleo (%)')
    plt.title('Relación entre la Participación en la Fuerza Laboral (%) y la␣
 ↪Tasa de Desempleo')
    plt.legend()

    # Mostrar la gráfica
    plt.grid(True)
    plt.show()
else:
    print("Las columnas necesarias no están disponibles en el DataFrame.")
```

```
Columnas del DataFrame:
['Emisiones de CO2', 'Área Boscosa (%)']
La columna 'Participación en la Fuerza Laboral (%)' no está en el DataFrame.
Estas son las columnas disponibles:
['Emisiones de CO2', 'rea Boscosa (%)']
Las columnas necesarias no están disponibles en el DataFrame.
```

[ ]:
```
print("Columnas en X_test:")
print(X_test.columns.tolist())
```

```
Columnas en X_test:
['Emisiones de CO2', 'Área Boscosa (%)']
```

[ ]:
```
import pandas as pd

# Cargar el archivo subido por el usuario
file_path = '/content/sample_data/world-data-2023-cleaned-final.csv'
df = pd.read_csv(file_path)

# Mostrar la información básica del dataset (columnas y primeras filas) para␣
 ↪visualización
df_columns = df.columns
df_head = df.head()
```

```
df_columns, df_head
```

```
[ ]: (Index(['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',
        'Área de Tierra (Km2)', 'Tamaño de las Fuerzas Armadas',
        'Tasa de Natalidad', 'Código Telefónico', 'Capital/Ciudad Principal',
        'Emisiones de CO2', 'Índice de Precios al Consumidor (IPC)',
        'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de Fertilidad',
        'Área Boscosa (%)', 'Precio de la Gasolina',
        'Producto Interno Bruto (PIB)',
        'Inscripción Bruta en Educación Primaria (%)',
        'Inscripción Bruta en Educación Terciaria (%)', 'Mortalidad Infantil',
        'Ciudad Más Grande', 'Esperanza de Vida', 'Tasa de Mortalidad Materna',
        'Salario Mínimo', 'Idioma Oficial', 'Gastos de Salud de Bolsillo (%)',
        'Médicos por Mil Habitantes', 'Población',
        'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
        'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana',
        'Latitud', 'Longitud'],
       dtype='object'),
          País  Densidad (P/Km2) Abreviación  Terreno Agrícola (%)  \
0  Afghanistan               60          AF                  58.1
1      Albania              105          AL                  43.1
2      Algeria               18          DZ                  17.4
3      Andorra              164          AD                  40.0
4       Angola               26          AO                  47.5

   Área de Tierra (Km2) Tamaño de las Fuerzas Armadas  Tasa de Natalidad  \
0             652230.0                       323,000              32.49
1              28748.0                         9,000              11.78
2            2381741.0                       317,000              24.28
3                468.0                           NaN               7.20
4            1246700.0                       117,000              40.73

   Código Telefónico Capital/Ciudad Principal  Emisiones de CO2  …  \
0               93.0                    Kabul            8672.0  …
1              355.0                   Tirana            4536.0  …
2              213.0                  Algiers          150006.0  …
3              376.0        Andorra la Vella             469.0  …
4              244.0                   Luanda           34693.0  …

  Gastos de Salud de Bolsillo (%) Médicos por Mil Habitantes   Población  \
0                            78.4                       0.28  38,041,754
1                            56.9                       1.20   2,854,191
2                            28.1                       1.72  43,053,054
3                            36.4                       3.33      77,142
4                            33.4                       0.21  31,825,295

   Participación en la Fuerza Laboral (%)  Ingresos Fiscales (%)  \
```

```
0                                  48.9                    9.3
1                                  55.7                   18.6
2                                  41.2                   37.2
3                                   NaN                    NaN
4                                  77.5                    9.2

   Tasa Impositiva Total Tasa de Desempleo  Población Urbana    Latitud  \
0                   71.4            11.12        9797273.0  33.939110
1                   36.6            12.33        1747593.0  41.153332
2                   66.1            11.70       31510100.0  28.033886
3                    NaN              NaN          67873.0  42.506285
4                   49.1             6.89       21061025.0 -11.202692

    Longitud
0  67.709953
1  20.168331
2   1.659626
3   1.521801
4  17.873887

[5 rows x 35 columns])
```

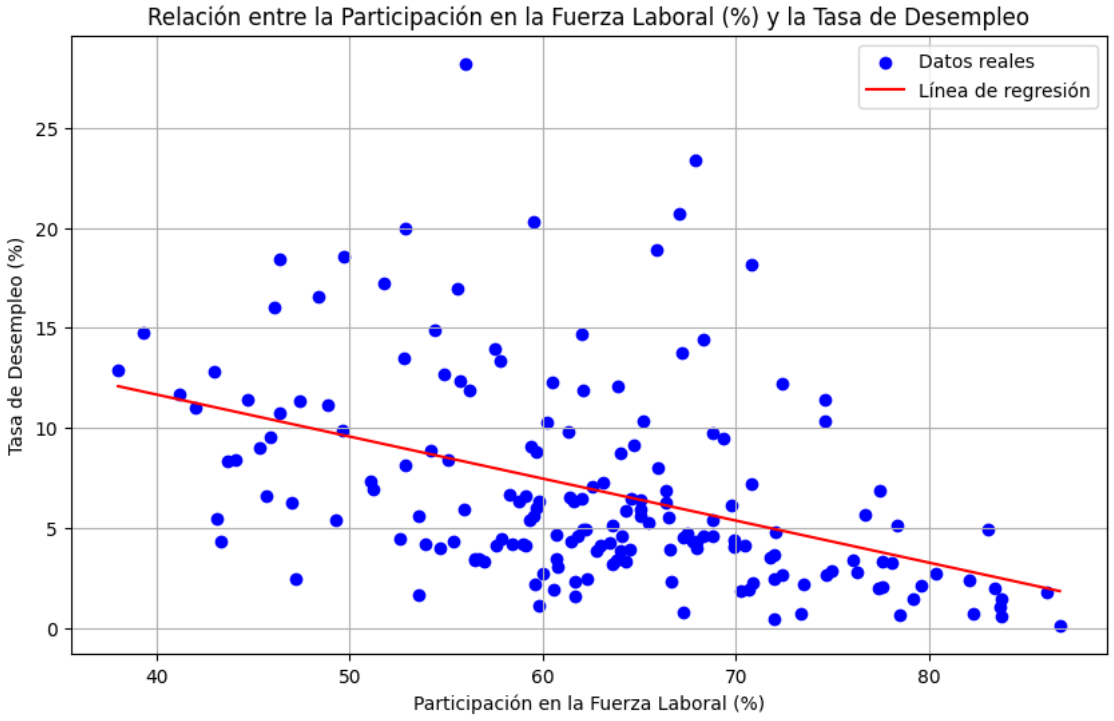
[ ]:
```python
import pandas as pd

# Recupera todas las columnas a partir del DataFrame original
df_selected_new = df.copy()  # Asegúrate de usar el DataFrame original con todas las columnas

# Verifica que tienes todas las columnas
print("Columnas del DataFrame original restaurado:")
print(df_selected_new.columns.tolist())
```

```
Columnas del DataFrame original restaurado:
['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)', 'Área de
Tierra (Km2)', 'Tamaño de las Fuerzas Armadas', 'Tasa de Natalidad', 'Código
Telefónico', 'Capital/Ciudad Principal', 'Emisiones de CO2', 'Índice de Precios
al Consumidor (IPC)', 'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de
Fertilidad', 'Área Boscosa (%)', 'Precio de la Gasolina', 'Producto Interno
Bruto (PIB)', 'Inscripción Bruta en Educación Primaria (%)', 'Inscripción Bruta
en Educación Terciaria (%)', 'Mortalidad Infantil', 'Ciudad Más Grande',
'Esperanza de Vida', 'Tasa de Mortalidad Materna', 'Salario Mínimo', 'Idioma
Oficial', 'Gastos de Salud de Bolsillo (%)', 'Médicos por Mil Habitantes',
'Población', 'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana', 'Latitud',
'Longitud']
```

[ ]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import pandas as pd

# Asegúrate de que estás trabajando con el DataFrame completo
df_selected_new = df.copy()  # Asegúrate de usar el DataFrame original con␣
 ↪todas las columnas

# Verifica que tienes todas las columnas
print("Columnas del DataFrame original restaurado:")
print(df_selected_new.columns.tolist())

# Prepara los datos para el modelo
X_train = df_selected_new[['Participación en la Fuerza Laboral (%)']]  #␣
 ↪Características
y_train = df_selected_new['Tasa de Desempleo']  # Objetivo

# Manejar los valores NaN antes de entrenar el modelo
# Opción 1: Eliminar filas con NaN
X_train = X_train.dropna()
y_train = y_train[X_train.index]  # Asegurar que y_train tenga los mismos␣
 ↪índices que X_train

# Opción 2: Reemplazar NaN con un valor (por ejemplo, la media)
# X_train.fillna(X_train.mean(), inplace=True)

# Entrena el modelo de regresión lineal
model = LinearRegression()
model.fit(X_train, y_train)

# Graficar la relación entre la participación laboral y la tasa de desempleo
if 'Participación en la Fuerza Laboral (%)' in df_selected_new.columns and␣
 ↪'Tasa de Desempleo' in df_selected_new.columns:
    plt.figure(figsize=(10, 6))
    plt.scatter(df_selected_new['Participación en la Fuerza Laboral (%)'],␣
 ↪df_selected_new['Tasa de Desempleo'], color='blue', label='Datos reales')

    # Crear el gráfico de regresión lineal
    X_plot = df_selected_new[['Participación en la Fuerza Laboral (%)']].
 ↪sort_values(by='Participación en la Fuerza Laboral (%)')
    # Eliminar filas con NaN en X_plot para que coincida con los datos de␣
 ↪entrenamiento
    X_plot = X_plot.dropna()
    y_plot = model.predict(X_plot)
    plt.plot(X_plot['Participación en la Fuerza Laboral (%)'], y_plot,␣
 ↪color='red', label='Línea de regresión')
```

```
    plt.xlabel('Participación en la Fuerza Laboral (%)')
    plt.ylabel('Tasa de Desempleo (%)')
    plt.title('Relación entre la Participación en la Fuerza Laboral (%) y la␣
 ↪Tasa de Desempleo')
    plt.legend()
    plt.grid(True)
    plt.show()
else:
    print("Las columnas necesarias no están disponibles en el DataFrame.")
```

```
Columnas del DataFrame original restaurado:
['País', 'Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)', 'Área de
Tierra (Km2)', 'Tamaño de las Fuerzas Armadas', 'Tasa de Natalidad', 'Código
Telefónico', 'Capital/Ciudad Principal', 'Emisiones de CO2', 'Índice de Precios
al Consumidor (IPC)', 'Cambio del IPC (%)', 'Código de Moneda', 'Tasa de
Fertilidad', 'Área Boscosa (%)', 'Precio de la Gasolina', 'Producto Interno
Bruto (PIB)', 'Inscripción Bruta en Educación Primaria (%)', 'Inscripción Bruta
en Educación Terciaria (%)', 'Mortalidad Infantil', 'Ciudad Más Grande',
'Esperanza de Vida', 'Tasa de Mortalidad Materna', 'Salario Mínimo', 'Idioma
Oficial', 'Gastos de Salud de Bolsillo (%)', 'Médicos por Mil Habitantes',
'Población', 'Participación en la Fuerza Laboral (%)', 'Ingresos Fiscales (%)',
'Tasa Impositiva Total', 'Tasa de Desempleo', 'Población Urbana', 'Latitud',
'Longitud']
```



30 Análisis del Gráfico:

1. **Datos Reales (puntos azules):**

- Los puntos azules representan los datos reales sobre la participación en la fuerza laboral y la tasa de desempleo.
- La dispersión de los puntos muestra que, a medida que la participación en la fuerza laboral aumenta, la tasa de desempleo tiene una tendencia a disminuir, aunque no es una relación extremadamente fuerte (hay mucha dispersión en los puntos).

2. **Línea de Regresión (línea roja):**

- La línea de regresión indica la relación lineal que el modelo ha encontrado entre estas dos variables.
- La pendiente negativa de la línea (descendente) indica que a mayor participación laboral, menor tasa de desempleo, lo que respalda la hipótesis planteada.

La pendiente negativa de la línea apoya la hipótesis de que, a mayor participación en la población activa, la tasa de desempleo tiende a disminuir.

31 Conclusión de la hipótesis 5:

El gráfico y la regresión lineal apoyan la hipótesis de que un aumento en la participación en la fuerza laboral se asocia con una disminución en la tasa de desempleo. Sin embargo, la dispersión de los puntos muestra que la relación no es perfecta y hay varios factores adicionales que podrían estar influyendo en la tasa de desempleo que no están siendo modelados aquí.

32 Predicciones

Objetivo del modelo:

- Simular agentes (países) con variables económicas dinámicas como el PIB, tasa de desempleo, participación laboral, entre otras. Utilizar el modelo de predicción (por ejemplo, Random Forest) para prever el impacto futuro de ciertas variables a lo largo del tiempo.
- Analizar las interacciones entre agentes y las predicciones.

32.1 Simulación con Mesa y predicciones usando Random Forest

[ ]:
```
from mesa import Agent, Model
from mesa.time import RandomActivation
from mesa.space import MultiGrid
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt

# Definir la clase de agente
class CountryAgent(Agent):
    def __init__(self, unique_id, model):
```

```
        super().__init__(unique_id, model)
        self.pib = np.random.uniform(1000, 50000)  # Producto Interno Bruto␣
↪inicial
        self.desempleo = np.random.uniform(3, 20)  # Tasa de desempleo inicial
        self.fuerza_laboral = np.random.uniform(40, 80)  # Participación␣
↪laboral inicial
        self.historical_data = []  # Almacenar datos para entrenar el modelo

    def step(self):
        # Simular cambios económicos
        self.pib += np.random.uniform(-1000, 2000)  # Cambio aleatorio en el PIB
        self.desempleo += np.random.uniform(-1, 1)  # Cambio aleatorio en la␣
↪tasa de desempleo
        self.fuerza_laboral += np.random.uniform(-0.5, 0.5)  # Cambio aleatorio␣
↪en la fuerza laboral

        # Almacenar datos históricos para el modelo
        self.historical_data.append([self.pib, self.fuerza_laboral, self.
↪desempleo])

        # Predicción basada en datos históricos
        if len(self.historical_data) > 5:  # Esperar a tener suficientes datos
            self.train_and_predict()

    def train_and_predict(self):
        # Preparar los datos para el modelo
        data = np.array(self.historical_data)
        X = data[:-1, :2]  # Usar PIB y fuerza laboral como características
        y = data[:-1, 2]  # Usar la tasa de desempleo como variable objetivo

        # Entrenar el modelo Random Forest
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪2, random_state=42)
        rf = RandomForestRegressor()
        rf.fit(X_train, y_train)

        # Predecir la tasa de desempleo para el siguiente paso
        next_input = np.array([self.pib, self.fuerza_laboral]).reshape(1, -1)
        predicted_unemployment = rf.predict(next_input)

        # Actualizar la tasa de desempleo con la predicción del modelo
        self.desempleo = predicted_unemployment[0]
        print(f"Agente {self.unique_id} - Predicción de desempleo: {self.
↪desempleo:.2f}")

# Definir el modelo de Mesa
```

```
class EconomicModel(Model):
    def __init__(self, N):
        self.num_agents = N
        self.grid = MultiGrid(10, 10, True)
        self.schedule = RandomActivation(self)

        # Crear agentes (países)
        for i in range(self.num_agents):
            agent = CountryAgent(i, self)
            self.schedule.add(agent)

        # Almacenar promedios de variables en cada paso
        self.pib_history = []
        self.desempleo_history = []
        self.fuerza_laboral_history = []

    def step(self):
        self.schedule.step()

        # Calcular promedios en cada paso de simulación
        avg_pib = np.mean([agent.pib for agent in self.schedule.agents])
        avg_desempleo = np.mean([agent.desempleo for agent in self.schedule.
 ↪agents])
        avg_fuerza_laboral = np.mean([agent.fuerza_laboral for agent in self.
 ↪schedule.agents])

        # Almacenar los promedios
        self.pib_history.append(avg_pib)
        self.desempleo_history.append(avg_desempleo)
        self.fuerza_laboral_history.append(avg_fuerza_laboral)

# Inicializar el modelo con 10 agentes
model = EconomicModel(10)

# Ejecutar la simulación durante 100 pasos
for i in range(100):
    model.step()

# Graficar los resultados de la simulación
plt.figure(figsize=(10, 6))
plt.plot(model.pib_history, label='PIB promedio')
plt.plot(model.desempleo_history, label='Tasa de desempleo promedio')
plt.plot(model.fuerza_laboral_history, label='Participación laboral promedio')
plt.title('Simulación Económica con Predicciones usando Random Forest')
plt.xlabel('Tiempo (Pasos de Simulación)')
plt.ylabel('Valores Promedio')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/mesa/agent.py:52: FutureWarning: The
Mesa Model class was not initialized. In the future, you need to explicitly
initialize the Model by calling super().__init__() on initialization.
  self.model.register_agent(self)
```

```
Agente 4 - Predicción de desempleo: 16.26
Agente 8 - Predicción de desempleo: 20.98
Agente 2 - Predicción de desempleo: 17.08
Agente 9 - Predicción de desempleo: 10.82
Agente 7 - Predicción de desempleo: 9.55
Agente 0 - Predicción de desempleo: 15.09
Agente 3 - Predicción de desempleo: 12.03
Agente 5 - Predicción de desempleo: 17.48
Agente 6 - Predicción de desempleo: 18.06
Agente 1 - Predicción de desempleo: 8.75
Agente 8 - Predicción de desempleo: 20.89
Agente 4 - Predicción de desempleo: 16.27
Agente 3 - Predicción de desempleo: 11.87
Agente 2 - Predicción de desempleo: 17.52
Agente 0 - Predicción de desempleo: 14.92
Agente 5 - Predicción de desempleo: 17.59
Agente 9 - Predicción de desempleo: 10.46
Agente 1 - Predicción de desempleo: 9.30
Agente 6 - Predicción de desempleo: 18.08
Agente 7 - Predicción de desempleo: 9.63
Agente 5 - Predicción de desempleo: 17.21
Agente 3 - Predicción de desempleo: 12.39
Agente 4 - Predicción de desempleo: 16.16
Agente 6 - Predicción de desempleo: 18.24
Agente 7 - Predicción de desempleo: 9.39
Agente 1 - Predicción de desempleo: 8.47
Agente 8 - Predicción de desempleo: 21.34
Agente 9 - Predicción de desempleo: 11.00
Agente 2 - Predicción de desempleo: 17.01
Agente 0 - Predicción de desempleo: 14.67
Agente 2 - Predicción de desempleo: 17.12
Agente 6 - Predicción de desempleo: 18.30
Agente 9 - Predicción de desempleo: 11.27
Agente 1 - Predicción de desempleo: 8.88
Agente 5 - Predicción de desempleo: 17.91
Agente 0 - Predicción de desempleo: 14.97
Agente 4 - Predicción de desempleo: 16.02
Agente 8 - Predicción de desempleo: 21.49
Agente 7 - Predicción de desempleo: 9.16
Agente 3 - Predicción de desempleo: 12.66
```

```
Agente 1 - Predicción de desempleo: 8.78
Agente 2 - Predicción de desempleo: 16.55
Agente 3 - Predicción de desempleo: 12.79
Agente 6 - Predicción de desempleo: 18.10
Agente 5 - Predicción de desempleo: 17.89
Agente 7 - Predicción de desempleo: 9.06
Agente 8 - Predicción de desempleo: 21.79
Agente 0 - Predicción de desempleo: 14.50
Agente 4 - Predicción de desempleo: 16.62
Agente 9 - Predicción de desempleo: 10.49
Agente 1 - Predicción de desempleo: 9.02
Agente 0 - Predicción de desempleo: 15.20
Agente 8 - Predicción de desempleo: 21.80
Agente 7 - Predicción de desempleo: 8.73
Agente 4 - Predicción de desempleo: 16.46
Agente 2 - Predicción de desempleo: 16.96
Agente 3 - Predicción de desempleo: 13.15
Agente 9 - Predicción de desempleo: 11.13
Agente 6 - Predicción de desempleo: 18.00
Agente 5 - Predicción de desempleo: 17.56
Agente 9 - Predicción de desempleo: 10.74
Agente 2 - Predicción de desempleo: 16.81
Agente 1 - Predicción de desempleo: 8.25
Agente 3 - Predicción de desempleo: 12.41
Agente 8 - Predicción de desempleo: 21.44
Agente 4 - Predicción de desempleo: 16.38
Agente 0 - Predicción de desempleo: 14.68
Agente 5 - Predicción de desempleo: 18.04
Agente 7 - Predicción de desempleo: 9.70
Agente 6 - Predicción de desempleo: 18.13
Agente 9 - Predicción de desempleo: 10.86
Agente 8 - Predicción de desempleo: 21.46
Agente 3 - Predicción de desempleo: 12.52
Agente 4 - Predicción de desempleo: 16.49
Agente 0 - Predicción de desempleo: 15.08
Agente 6 - Predicción de desempleo: 17.85
Agente 7 - Predicción de desempleo: 9.17
Agente 5 - Predicción de desempleo: 17.81
Agente 2 - Predicción de desempleo: 16.32
Agente 1 - Predicción de desempleo: 8.98
Agente 2 - Predicción de desempleo: 17.27
Agente 9 - Predicción de desempleo: 10.63
Agente 0 - Predicción de desempleo: 15.08
Agente 7 - Predicción de desempleo: 9.29
Agente 3 - Predicción de desempleo: 12.33
Agente 6 - Predicción de desempleo: 17.94
Agente 8 - Predicción de desempleo: 21.51
Agente 4 - Predicción de desempleo: 16.17
```

```
Agente 1 - Predicción de desempleo: 8.98
Agente 5 - Predicción de desempleo: 17.88
Agente 9 - Predicción de desempleo: 10.47
Agente 4 - Predicción de desempleo: 16.32
Agente 5 - Predicción de desempleo: 17.86
Agente 0 - Predicción de desempleo: 15.01
Agente 1 - Predicción de desempleo: 8.92
Agente 6 - Predicción de desempleo: 17.95
Agente 3 - Predicción de desempleo: 12.33
Agente 8 - Predicción de desempleo: 21.31
Agente 7 - Predicción de desempleo: 9.71
Agente 2 - Predicción de desempleo: 16.79
Agente 9 - Predicción de desempleo: 10.77
Agente 5 - Predicción de desempleo: 17.98
Agente 6 - Predicción de desempleo: 17.98
Agente 2 - Predicción de desempleo: 16.80
Agente 0 - Predicción de desempleo: 15.50
Agente 1 - Predicción de desempleo: 9.17
Agente 4 - Predicción de desempleo: 16.47
Agente 3 - Predicción de desempleo: 12.78
Agente 7 - Predicción de desempleo: 9.14
Agente 8 - Predicción de desempleo: 21.77
Agente 7 - Predicción de desempleo: 9.87
Agente 3 - Predicción de desempleo: 11.97
Agente 6 - Predicción de desempleo: 17.77
Agente 1 - Predicción de desempleo: 8.25
Agente 9 - Predicción de desempleo: 9.89
Agente 4 - Predicción de desempleo: 16.78
Agente 8 - Predicción de desempleo: 21.39
Agente 0 - Predicción de desempleo: 14.72
Agente 5 - Predicción de desempleo: 17.82
Agente 2 - Predicción de desempleo: 16.58
Agente 8 - Predicción de desempleo: 21.29
Agente 4 - Predicción de desempleo: 16.72
Agente 0 - Predicción de desempleo: 15.15
Agente 6 - Predicción de desempleo: 17.59
Agente 7 - Predicción de desempleo: 8.72
Agente 1 - Predicción de desempleo: 9.13
Agente 5 - Predicción de desempleo: 17.85
Agente 3 - Predicción de desempleo: 13.14
Agente 9 - Predicción de desempleo: 10.98
Agente 2 - Predicción de desempleo: 16.44
Agente 4 - Predicción de desempleo: 17.01
Agente 3 - Predicción de desempleo: 12.16
Agente 7 - Predicción de desempleo: 9.08
Agente 5 - Predicción de desempleo: 17.90
Agente 9 - Predicción de desempleo: 10.73
Agente 2 - Predicción de desempleo: 16.34
```

```
Agente 0 - Predicción de desempleo: 15.41
Agente 6 - Predicción de desempleo: 17.87
Agente 8 - Predicción de desempleo: 21.74
Agente 1 - Predicción de desempleo: 8.12
Agente 0 - Predicción de desempleo: 14.84
Agente 7 - Predicción de desempleo: 9.33
Agente 1 - Predicción de desempleo: 8.66
Agente 3 - Predicción de desempleo: 12.51
Agente 2 - Predicción de desempleo: 16.44
Agente 6 - Predicción de desempleo: 18.27
Agente 8 - Predicción de desempleo: 20.69
Agente 9 - Predicción de desempleo: 10.65
Agente 5 - Predicción de desempleo: 17.97
Agente 4 - Predicción de desempleo: 16.79
Agente 2 - Predicción de desempleo: 16.65
Agente 6 - Predicción de desempleo: 17.66
Agente 8 - Predicción de desempleo: 20.90
Agente 5 - Predicción de desempleo: 18.27
Agente 7 - Predicción de desempleo: 9.21
Agente 9 - Predicción de desempleo: 10.58
Agente 3 - Predicción de desempleo: 12.97
Agente 0 - Predicción de desempleo: 15.30
Agente 4 - Predicción de desempleo: 16.75
Agente 1 - Predicción de desempleo: 8.32
Agente 6 - Predicción de desempleo: 18.60
Agente 7 - Predicción de desempleo: 9.30
Agente 5 - Predicción de desempleo: 18.10
Agente 9 - Predicción de desempleo: 10.90
Agente 0 - Predicción de desempleo: 15.51
Agente 3 - Predicción de desempleo: 12.69
Agente 8 - Predicción de desempleo: 20.96
Agente 4 - Predicción de desempleo: 16.00
Agente 1 - Predicción de desempleo: 8.82
Agente 2 - Predicción de desempleo: 16.75
Agente 6 - Predicción de desempleo: 18.59
Agente 2 - Predicción de desempleo: 16.70
Agente 5 - Predicción de desempleo: 18.46
Agente 7 - Predicción de desempleo: 9.52
Agente 1 - Predicción de desempleo: 8.37
Agente 8 - Predicción de desempleo: 20.56
Agente 0 - Predicción de desempleo: 15.86
Agente 9 - Predicción de desempleo: 10.75
Agente 3 - Predicción de desempleo: 12.69
Agente 4 - Predicción de desempleo: 16.06
Agente 4 - Predicción de desempleo: 17.10
Agente 6 - Predicción de desempleo: 18.25
Agente 8 - Predicción de desempleo: 20.78
Agente 9 - Predicción de desempleo: 11.04
```

```
Agente 1 - Predicción de desempleo: 8.45
Agente 5 - Predicción de desempleo: 18.41
Agente 0 - Predicción de desempleo: 15.50
Agente 2 - Predicción de desempleo: 16.91
Agente 3 - Predicción de desempleo: 12.51
Agente 7 - Predicción de desempleo: 9.51
Agente 4 - Predicción de desempleo: 16.33
Agente 6 - Predicción de desempleo: 18.88
Agente 5 - Predicción de desempleo: 18.54
Agente 9 - Predicción de desempleo: 10.93
Agente 2 - Predicción de desempleo: 16.46
Agente 0 - Predicción de desempleo: 15.92
Agente 8 - Predicción de desempleo: 21.01
Agente 1 - Predicción de desempleo: 7.91
Agente 3 - Predicción de desempleo: 12.85
Agente 7 - Predicción de desempleo: 9.89
Agente 6 - Predicción de desempleo: 18.95
Agente 2 - Predicción de desempleo: 16.75
Agente 4 - Predicción de desempleo: 16.74
Agente 1 - Predicción de desempleo: 8.91
Agente 7 - Predicción de desempleo: 9.76
Agente 8 - Predicción de desempleo: 21.24
Agente 0 - Predicción de desempleo: 14.86
Agente 9 - Predicción de desempleo: 10.46
Agente 3 - Predicción de desempleo: 12.56
Agente 5 - Predicción de desempleo: 18.23
Agente 9 - Predicción de desempleo: 10.85
Agente 2 - Predicción de desempleo: 16.74
Agente 0 - Predicción de desempleo: 15.73
Agente 8 - Predicción de desempleo: 21.23
Agente 6 - Predicción de desempleo: 19.20
Agente 7 - Predicción de desempleo: 9.75
Agente 4 - Predicción de desempleo: 15.67
Agente 5 - Predicción de desempleo: 18.20
Agente 1 - Predicción de desempleo: 8.25
Agente 3 - Predicción de desempleo: 12.34
Agente 2 - Predicción de desempleo: 16.54
Agente 1 - Predicción de desempleo: 8.51
Agente 5 - Predicción de desempleo: 18.38
Agente 8 - Predicción de desempleo: 21.55
Agente 9 - Predicción de desempleo: 10.89
Agente 4 - Predicción de desempleo: 16.67
Agente 7 - Predicción de desempleo: 9.29
Agente 3 - Predicción de desempleo: 12.28
Agente 0 - Predicción de desempleo: 15.55
Agente 6 - Predicción de desempleo: 19.19
Agente 4 - Predicción de desempleo: 16.30
Agente 2 - Predicción de desempleo: 17.05
```

```
Agente 0 - Predicción de desempleo: 14.91
Agente 5 - Predicción de desempleo: 18.13
Agente 1 - Predicción de desempleo: 8.51
Agente 3 - Predicción de desempleo: 12.45
Agente 9 - Predicción de desempleo: 11.21
Agente 6 - Predicción de desempleo: 17.82
Agente 8 - Predicción de desempleo: 21.65
Agente 7 - Predicción de desempleo: 9.87
Agente 5 - Predicción de desempleo: 18.06
Agente 1 - Predicción de desempleo: 8.47
Agente 0 - Predicción de desempleo: 15.26
Agente 2 - Predicción de desempleo: 16.51
Agente 8 - Predicción de desempleo: 22.09
Agente 9 - Predicción de desempleo: 11.00
Agente 7 - Predicción de desempleo: 9.04
Agente 4 - Predicción de desempleo: 16.32
Agente 3 - Predicción de desempleo: 12.55
Agente 6 - Predicción de desempleo: 18.82
Agente 4 - Predicción de desempleo: 16.27
Agente 1 - Predicción de desempleo: 8.69
Agente 7 - Predicción de desempleo: 9.51
Agente 2 - Predicción de desempleo: 17.04
Agente 3 - Predicción de desempleo: 12.65
Agente 9 - Predicción de desempleo: 10.86
Agente 6 - Predicción de desempleo: 18.70
Agente 5 - Predicción de desempleo: 17.70
Agente 0 - Predicción de desempleo: 14.51
Agente 8 - Predicción de desempleo: 22.17
Agente 6 - Predicción de desempleo: 18.97
Agente 3 - Predicción de desempleo: 13.07
Agente 9 - Predicción de desempleo: 10.54
Agente 4 - Predicción de desempleo: 16.22
Agente 8 - Predicción de desempleo: 22.08
Agente 0 - Predicción de desempleo: 15.51
Agente 5 - Predicción de desempleo: 18.38
Agente 1 - Predicción de desempleo: 8.31
Agente 7 - Predicción de desempleo: 8.90
Agente 2 - Predicción de desempleo: 17.12
Agente 2 - Predicción de desempleo: 16.83
Agente 7 - Predicción de desempleo: 9.31
Agente 5 - Predicción de desempleo: 17.20
Agente 4 - Predicción de desempleo: 15.81
Agente 0 - Predicción de desempleo: 14.34
Agente 6 - Predicción de desempleo: 19.33
Agente 3 - Predicción de desempleo: 12.94
Agente 9 - Predicción de desempleo: 11.17
Agente 1 - Predicción de desempleo: 8.34
Agente 8 - Predicción de desempleo: 22.19
```

```
Agente 0 - Predicción de desempleo: 15.39
Agente 4 - Predicción de desempleo: 16.38
Agente 8 - Predicción de desempleo: 22.07
Agente 9 - Predicción de desempleo: 10.34
Agente 3 - Predicción de desempleo: 13.47
Agente 5 - Predicción de desempleo: 17.80
Agente 1 - Predicción de desempleo: 8.18
Agente 6 - Predicción de desempleo: 19.02
Agente 7 - Predicción de desempleo: 9.07
Agente 2 - Predicción de desempleo: 16.71
Agente 0 - Predicción de desempleo: 14.39
Agente 6 - Predicción de desempleo: 18.96
Agente 9 - Predicción de desempleo: 10.79
Agente 3 - Predicción de desempleo: 13.46
Agente 7 - Predicción de desempleo: 8.71
Agente 4 - Predicción de desempleo: 16.40
Agente 8 - Predicción de desempleo: 22.57
Agente 5 - Predicción de desempleo: 17.02
Agente 1 - Predicción de desempleo: 8.02
Agente 2 - Predicción de desempleo: 16.65
Agente 9 - Predicción de desempleo: 10.86
Agente 7 - Predicción de desempleo: 8.66
Agente 4 - Predicción de desempleo: 16.06
Agente 3 - Predicción de desempleo: 12.80
Agente 8 - Predicción de desempleo: 22.62
Agente 5 - Predicción de desempleo: 17.13
Agente 0 - Predicción de desempleo: 14.52
Agente 2 - Predicción de desempleo: 16.17
Agente 1 - Predicción de desempleo: 8.27
Agente 6 - Predicción de desempleo: 18.64
Agente 8 - Predicción de desempleo: 22.51
Agente 3 - Predicción de desempleo: 12.65
Agente 1 - Predicción de desempleo: 8.52
Agente 2 - Predicción de desempleo: 16.16
Agente 6 - Predicción de desempleo: 18.75
Agente 7 - Predicción de desempleo: 9.36
Agente 0 - Predicción de desempleo: 14.60
Agente 5 - Predicción de desempleo: 17.33
Agente 4 - Predicción de desempleo: 16.21
Agente 9 - Predicción de desempleo: 10.86
Agente 2 - Predicción de desempleo: 16.54
Agente 1 - Predicción de desempleo: 7.76
Agente 5 - Predicción de desempleo: 16.86
Agente 3 - Predicción de desempleo: 12.44
Agente 9 - Predicción de desempleo: 10.76
Agente 4 - Predicción de desempleo: 16.55
Agente 6 - Predicción de desempleo: 18.73
Agente 0 - Predicción de desempleo: 14.64
```

```
Agente 7 - Predicción de desempleo: 9.02
Agente 8 - Predicción de desempleo: 22.80
Agente 5 - Predicción de desempleo: 17.59
Agente 3 - Predicción de desempleo: 12.92
Agente 4 - Predicción de desempleo: 16.19
Agente 7 - Predicción de desempleo: 9.30
Agente 9 - Predicción de desempleo: 11.39
Agente 2 - Predicción de desempleo: 16.08
Agente 6 - Predicción de desempleo: 18.59
Agente 8 - Predicción de desempleo: 21.86
Agente 1 - Predicción de desempleo: 7.85
Agente 0 - Predicción de desempleo: 14.47
Agente 0 - Predicción de desempleo: 14.49
Agente 5 - Predicción de desempleo: 17.64
Agente 1 - Predicción de desempleo: 7.55
Agente 8 - Predicción de desempleo: 22.39
Agente 6 - Predicción de desempleo: 18.98
Agente 9 - Predicción de desempleo: 10.87
Agente 4 - Predicción de desempleo: 16.23
Agente 2 - Predicción de desempleo: 16.06
Agente 7 - Predicción de desempleo: 9.53
Agente 3 - Predicción de desempleo: 12.31
Agente 1 - Predicción de desempleo: 7.79
Agente 5 - Predicción de desempleo: 17.47
Agente 4 - Predicción de desempleo: 16.02
Agente 6 - Predicción de desempleo: 19.36
Agente 9 - Predicción de desempleo: 10.95
Agente 7 - Predicción de desempleo: 8.89
Agente 8 - Predicción de desempleo: 22.06
Agente 3 - Predicción de desempleo: 12.70
Agente 2 - Predicción de desempleo: 15.90
Agente 0 - Predicción de desempleo: 15.10
Agente 9 - Predicción de desempleo: 10.50
Agente 5 - Predicción de desempleo: 17.58
Agente 6 - Predicción de desempleo: 19.14
Agente 4 - Predicción de desempleo: 16.45
Agente 3 - Predicción de desempleo: 12.15
Agente 8 - Predicción de desempleo: 22.14
Agente 1 - Predicción de desempleo: 7.15
Agente 2 - Predicción de desempleo: 16.53
Agente 0 - Predicción de desempleo: 14.44
Agente 7 - Predicción de desempleo: 8.86
Agente 0 - Predicción de desempleo: 14.66
Agente 4 - Predicción de desempleo: 15.77
Agente 8 - Predicción de desempleo: 21.80
Agente 9 - Predicción de desempleo: 10.87
Agente 3 - Predicción de desempleo: 12.76
Agente 6 - Predicción de desempleo: 19.76
```

```
Agente 5 - Predicción de desempleo: 18.07
Agente 1 - Predicción de desempleo: 7.53
Agente 2 - Predicción de desempleo: 16.55
Agente 7 - Predicción de desempleo: 9.49
Agente 8 - Predicción de desempleo: 21.48
Agente 2 - Predicción de desempleo: 16.99
Agente 3 - Predicción de desempleo: 12.86
Agente 5 - Predicción de desempleo: 17.81
Agente 9 - Predicción de desempleo: 10.84
Agente 6 - Predicción de desempleo: 19.42
Agente 1 - Predicción de desempleo: 7.51
Agente 0 - Predicción de desempleo: 15.12
Agente 7 - Predicción de desempleo: 9.64
Agente 4 - Predicción de desempleo: 15.71
Agente 6 - Predicción de desempleo: 19.07
Agente 5 - Predicción de desempleo: 17.63
Agente 9 - Predicción de desempleo: 10.47
Agente 4 - Predicción de desempleo: 15.45
Agente 0 - Predicción de desempleo: 14.91
Agente 8 - Predicción de desempleo: 21.74
Agente 2 - Predicción de desempleo: 16.63
Agente 3 - Predicción de desempleo: 12.91
Agente 1 - Predicción de desempleo: 7.69
Agente 7 - Predicción de desempleo: 9.91
Agente 2 - Predicción de desempleo: 16.47
Agente 5 - Predicción de desempleo: 17.49
Agente 3 - Predicción de desempleo: 12.61
Agente 1 - Predicción de desempleo: 7.12
Agente 8 - Predicción de desempleo: 21.95
Agente 9 - Predicción de desempleo: 10.31
Agente 7 - Predicción de desempleo: 9.76
Agente 0 - Predicción de desempleo: 15.00
Agente 6 - Predicción de desempleo: 18.70
Agente 4 - Predicción de desempleo: 15.61
Agente 3 - Predicción de desempleo: 12.61
Agente 7 - Predicción de desempleo: 9.97
Agente 6 - Predicción de desempleo: 18.29
Agente 9 - Predicción de desempleo: 10.80
Agente 2 - Predicción de desempleo: 17.10
Agente 4 - Predicción de desempleo: 15.45
Agente 0 - Predicción de desempleo: 15.31
Agente 8 - Predicción de desempleo: 22.11
Agente 5 - Predicción de desempleo: 17.54
Agente 1 - Predicción de desempleo: 8.14
Agente 0 - Predicción de desempleo: 15.18
Agente 2 - Predicción de desempleo: 16.87
Agente 7 - Predicción de desempleo: 9.56
Agente 9 - Predicción de desempleo: 11.16
```

```
Agente 6 - Predicción de desempleo: 18.52
Agente 3 - Predicción de desempleo: 12.70
Agente 8 - Predicción de desempleo: 21.62
Agente 4 - Predicción de desempleo: 15.53
Agente 5 - Predicción de desempleo: 17.54
Agente 1 - Predicción de desempleo: 7.25
Agente 3 - Predicción de desempleo: 13.33
Agente 7 - Predicción de desempleo: 9.74
Agente 8 - Predicción de desempleo: 22.14
Agente 1 - Predicción de desempleo: 7.35
Agente 4 - Predicción de desempleo: 15.73
Agente 6 - Predicción de desempleo: 18.16
Agente 0 - Predicción de desempleo: 14.72
Agente 5 - Predicción de desempleo: 17.24
Agente 9 - Predicción de desempleo: 11.42
Agente 2 - Predicción de desempleo: 16.57
Agente 6 - Predicción de desempleo: 18.04
Agente 7 - Predicción de desempleo: 10.27
Agente 1 - Predicción de desempleo: 7.24
Agente 2 - Predicción de desempleo: 16.30
Agente 8 - Predicción de desempleo: 21.68
Agente 3 - Predicción de desempleo: 12.63
Agente 9 - Predicción de desempleo: 10.53
Agente 0 - Predicción de desempleo: 15.24
Agente 5 - Predicción de desempleo: 17.52
Agente 4 - Predicción de desempleo: 14.99
Agente 1 - Predicción de desempleo: 7.18
Agente 3 - Predicción de desempleo: 12.57
Agente 7 - Predicción de desempleo: 9.57
Agente 5 - Predicción de desempleo: 17.18
Agente 2 - Predicción de desempleo: 16.15
Agente 4 - Predicción de desempleo: 15.99
Agente 0 - Predicción de desempleo: 14.86
Agente 9 - Predicción de desempleo: 11.62
Agente 6 - Predicción de desempleo: 18.66
Agente 8 - Predicción de desempleo: 21.78
Agente 4 - Predicción de desempleo: 15.68
Agente 0 - Predicción de desempleo: 14.75
Agente 9 - Predicción de desempleo: 11.13
Agente 6 - Predicción de desempleo: 18.77
Agente 8 - Predicción de desempleo: 22.06
Agente 3 - Predicción de desempleo: 13.47
Agente 7 - Predicción de desempleo: 10.44
Agente 2 - Predicción de desempleo: 16.15
Agente 5 - Predicción de desempleo: 17.36
Agente 1 - Predicción de desempleo: 7.51
Agente 0 - Predicción de desempleo: 15.16
Agente 5 - Predicción de desempleo: 17.19
```

```
Agente 2 - Predicción de desempleo: 16.09
Agente 4 - Predicción de desempleo: 15.59
Agente 1 - Predicción de desempleo: 6.76
Agente 6 - Predicción de desempleo: 18.01
Agente 7 - Predicción de desempleo: 9.22
Agente 8 - Predicción de desempleo: 21.49
Agente 9 - Predicción de desempleo: 11.26
Agente 3 - Predicción de desempleo: 12.91
Agente 9 - Predicción de desempleo: 10.67
Agente 1 - Predicción de desempleo: 7.06
Agente 0 - Predicción de desempleo: 14.40
Agente 5 - Predicción de desempleo: 17.87
Agente 8 - Predicción de desempleo: 21.40
Agente 3 - Predicción de desempleo: 13.02
Agente 6 - Predicción de desempleo: 18.81
Agente 2 - Predicción de desempleo: 15.66
Agente 7 - Predicción de desempleo: 9.64
Agente 4 - Predicción de desempleo: 15.39
Agente 7 - Predicción de desempleo: 9.61
Agente 8 - Predicción de desempleo: 21.86
Agente 0 - Predicción de desempleo: 14.58
Agente 1 - Predicción de desempleo: 7.40
Agente 2 - Predicción de desempleo: 16.28
Agente 4 - Predicción de desempleo: 15.11
Agente 5 - Predicción de desempleo: 16.72
Agente 9 - Predicción de desempleo: 11.85
Agente 6 - Predicción de desempleo: 19.32
Agente 3 - Predicción de desempleo: 12.92
Agente 7 - Predicción de desempleo: 10.63
Agente 8 - Predicción de desempleo: 21.44
Agente 9 - Predicción de desempleo: 11.24
Agente 2 - Predicción de desempleo: 15.89
Agente 0 - Predicción de desempleo: 15.04
Agente 4 - Predicción de desempleo: 15.08
Agente 6 - Predicción de desempleo: 18.59
Agente 5 - Predicción de desempleo: 16.82
Agente 3 - Predicción de desempleo: 12.69
Agente 1 - Predicción de desempleo: 7.24
Agente 1 - Predicción de desempleo: 7.09
Agente 0 - Predicción de desempleo: 14.84
Agente 7 - Predicción de desempleo: 9.80
Agente 9 - Predicción de desempleo: 11.99
Agente 2 - Predicción de desempleo: 16.09
Agente 5 - Predicción de desempleo: 16.53
Agente 6 - Predicción de desempleo: 19.49
Agente 3 - Predicción de desempleo: 12.51
Agente 4 - Predicción de desempleo: 15.21
Agente 8 - Predicción de desempleo: 21.20
```

```
Agente 1 - Predicción de desempleo: 7.60
Agente 2 - Predicción de desempleo: 15.75
Agente 4 - Predicción de desempleo: 15.31
Agente 5 - Predicción de desempleo: 17.00
Agente 3 - Predicción de desempleo: 12.54
Agente 8 - Predicción de desempleo: 21.55
Agente 9 - Predicción de desempleo: 11.51
Agente 6 - Predicción de desempleo: 18.28
Agente 7 - Predicción de desempleo: 9.98
Agente 0 - Predicción de desempleo: 14.96
Agente 2 - Predicción de desempleo: 16.81
Agente 5 - Predicción de desempleo: 17.07
Agente 9 - Predicción de desempleo: 12.31
Agente 1 - Predicción de desempleo: 7.15
Agente 6 - Predicción de desempleo: 18.41
Agente 4 - Predicción de desempleo: 15.45
Agente 3 - Predicción de desempleo: 12.36
Agente 0 - Predicción de desempleo: 14.62
Agente 8 - Predicción de desempleo: 21.30
Agente 7 - Predicción de desempleo: 9.78
Agente 4 - Predicción de desempleo: 15.04
Agente 6 - Predicción de desempleo: 18.63
Agente 3 - Predicción de desempleo: 12.16
Agente 7 - Predicción de desempleo: 10.35
Agente 9 - Predicción de desempleo: 11.81
Agente 5 - Predicción de desempleo: 17.08
Agente 2 - Predicción de desempleo: 15.80
Agente 1 - Predicción de desempleo: 7.37
Agente 0 - Predicción de desempleo: 14.82
Agente 8 - Predicción de desempleo: 22.06
Agente 1 - Predicción de desempleo: 6.80
Agente 3 - Predicción de desempleo: 12.39
Agente 4 - Predicción de desempleo: 14.96
Agente 0 - Predicción de desempleo: 14.46
Agente 2 - Predicción de desempleo: 15.96
Agente 6 - Predicción de desempleo: 18.45
Agente 9 - Predicción de desempleo: 12.01
Agente 8 - Predicción de desempleo: 21.93
Agente 7 - Predicción de desempleo: 10.32
Agente 5 - Predicción de desempleo: 16.64
Agente 5 - Predicción de desempleo: 17.19
Agente 1 - Predicción de desempleo: 7.57
Agente 4 - Predicción de desempleo: 15.21
Agente 2 - Predicción de desempleo: 15.49
Agente 6 - Predicción de desempleo: 18.52
Agente 9 - Predicción de desempleo: 11.71
Agente 3 - Predicción de desempleo: 12.46
Agente 0 - Predicción de desempleo: 14.47
```

```
Agente 8 - Predicción de desempleo: 22.13
Agente 7 - Predicción de desempleo: 10.66
Agente 7 - Predicción de desempleo: 9.84
Agente 0 - Predicción de desempleo: 14.55
Agente 2 - Predicción de desempleo: 16.07
Agente 4 - Predicción de desempleo: 15.22
Agente 6 - Predicción de desempleo: 18.88
Agente 9 - Predicción de desempleo: 11.57
Agente 3 - Predicción de desempleo: 12.17
Agente 5 - Predicción de desempleo: 16.52
Agente 8 - Predicción de desempleo: 22.06
Agente 1 - Predicción de desempleo: 7.88
Agente 0 - Predicción de desempleo: 14.77
Agente 5 - Predicción de desempleo: 16.71
Agente 1 - Predicción de desempleo: 7.16
Agente 4 - Predicción de desempleo: 15.58
Agente 7 - Predicción de desempleo: 10.45
Agente 9 - Predicción de desempleo: 11.38
Agente 6 - Predicción de desempleo: 18.37
Agente 2 - Predicción de desempleo: 16.04
Agente 8 - Predicción de desempleo: 22.47
Agente 3 - Predicción de desempleo: 12.56
Agente 6 - Predicción de desempleo: 19.08
Agente 9 - Predicción de desempleo: 11.94
Agente 0 - Predicción de desempleo: 14.77
Agente 7 - Predicción de desempleo: 9.87
Agente 2 - Predicción de desempleo: 16.28
Agente 1 - Predicción de desempleo: 7.62
Agente 8 - Predicción de desempleo: 22.30
Agente 3 - Predicción de desempleo: 12.48
Agente 5 - Predicción de desempleo: 16.59
Agente 4 - Predicción de desempleo: 14.73
Agente 1 - Predicción de desempleo: 7.47
Agente 3 - Predicción de desempleo: 12.39
Agente 4 - Predicción de desempleo: 14.82
Agente 8 - Predicción de desempleo: 22.32
Agente 9 - Predicción de desempleo: 12.05
Agente 5 - Predicción de desempleo: 16.89
Agente 0 - Predicción de desempleo: 14.54
Agente 7 - Predicción de desempleo: 10.22
Agente 6 - Predicción de desempleo: 18.75
Agente 2 - Predicción de desempleo: 16.22
Agente 4 - Predicción de desempleo: 15.64
Agente 1 - Predicción de desempleo: 6.90
Agente 9 - Predicción de desempleo: 11.44
Agente 2 - Predicción de desempleo: 16.15
Agente 8 - Predicción de desempleo: 21.67
Agente 7 - Predicción de desempleo: 10.29
```

```
Agente 3 - Predicción de desempleo: 12.53
Agente 0 - Predicción de desempleo: 14.57
Agente 6 - Predicción de desempleo: 18.86
Agente 5 - Predicción de desempleo: 16.46
Agente 4 - Predicción de desempleo: 14.87
Agente 1 - Predicción de desempleo: 6.81
Agente 7 - Predicción de desempleo: 9.88
Agente 0 - Predicción de desempleo: 14.71
Agente 2 - Predicción de desempleo: 16.26
Agente 3 - Predicción de desempleo: 12.44
Agente 6 - Predicción de desempleo: 18.85
Agente 8 - Predicción de desempleo: 22.26
Agente 9 - Predicción de desempleo: 11.89
Agente 5 - Predicción de desempleo: 16.55
Agente 9 - Predicción de desempleo: 11.55
Agente 4 - Predicción de desempleo: 15.77
Agente 3 - Predicción de desempleo: 12.49
Agente 6 - Predicción de desempleo: 18.51
Agente 8 - Predicción de desempleo: 21.47
Agente 2 - Predicción de desempleo: 16.72
Agente 7 - Predicción de desempleo: 9.68
Agente 1 - Predicción de desempleo: 7.72
Agente 0 - Predicción de desempleo: 14.10
Agente 5 - Predicción de desempleo: 15.86
Agente 0 - Predicción de desempleo: 14.80
Agente 5 - Predicción de desempleo: 16.31
Agente 8 - Predicción de desempleo: 22.67
Agente 7 - Predicción de desempleo: 9.72
Agente 9 - Predicción de desempleo: 11.87
Agente 4 - Predicción de desempleo: 15.12
Agente 1 - Predicción de desempleo: 7.87
Agente 3 - Predicción de desempleo: 12.83
Agente 6 - Predicción de desempleo: 18.83
Agente 2 - Predicción de desempleo: 16.33
Agente 3 - Predicción de desempleo: 12.87
Agente 9 - Predicción de desempleo: 12.25
Agente 0 - Predicción de desempleo: 14.55
Agente 4 - Predicción de desempleo: 15.40
Agente 2 - Predicción de desempleo: 16.72
Agente 1 - Predicción de desempleo: 7.66
Agente 6 - Predicción de desempleo: 19.01
Agente 8 - Predicción de desempleo: 22.38
Agente 7 - Predicción de desempleo: 9.99
Agente 5 - Predicción de desempleo: 16.22
Agente 8 - Predicción de desempleo: 22.45
Agente 5 - Predicción de desempleo: 16.00
Agente 1 - Predicción de desempleo: 7.64
Agente 6 - Predicción de desempleo: 19.05
```

```
Agente 4 - Predicción de desempleo: 15.42
Agente 2 - Predicción de desempleo: 16.91
Agente 7 - Predicción de desempleo: 10.37
Agente 9 - Predicción de desempleo: 12.33
Agente 3 - Predicción de desempleo: 13.00
Agente 0 - Predicción de desempleo: 13.60
Agente 1 - Predicción de desempleo: 7.85
Agente 7 - Predicción de desempleo: 10.52
Agente 6 - Predicción de desempleo: 18.90
Agente 9 - Predicción de desempleo: 12.23
Agente 0 - Predicción de desempleo: 14.69
Agente 3 - Predicción de desempleo: 13.35
Agente 2 - Predicción de desempleo: 16.84
Agente 8 - Predicción de desempleo: 22.35
Agente 4 - Predicción de desempleo: 15.49
Agente 5 - Predicción de desempleo: 16.08
Agente 8 - Predicción de desempleo: 22.56
Agente 6 - Predicción de desempleo: 18.62
Agente 2 - Predicción de desempleo: 17.00
Agente 1 - Predicción de desempleo: 7.79
Agente 4 - Predicción de desempleo: 15.51
Agente 7 - Predicción de desempleo: 10.35
Agente 0 - Predicción de desempleo: 14.87
Agente 9 - Predicción de desempleo: 12.30
Agente 3 - Predicción de desempleo: 13.34
Agente 5 - Predicción de desempleo: 16.10
Agente 5 - Predicción de desempleo: 15.78
Agente 6 - Predicción de desempleo: 18.96
Agente 7 - Predicción de desempleo: 10.81
Agente 4 - Predicción de desempleo: 15.45
Agente 9 - Predicción de desempleo: 11.68
Agente 3 - Predicción de desempleo: 13.40
Agente 8 - Predicción de desempleo: 22.22
Agente 2 - Predicción de desempleo: 16.98
Agente 1 - Predicción de desempleo: 7.52
Agente 0 - Predicción de desempleo: 14.68
Agente 1 - Predicción de desempleo: 7.67
Agente 3 - Predicción de desempleo: 13.16
Agente 8 - Predicción de desempleo: 22.40
Agente 6 - Predicción de desempleo: 18.66
Agente 4 - Predicción de desempleo: 15.67
Agente 0 - Predicción de desempleo: 14.88
Agente 2 - Predicción de desempleo: 17.38
Agente 7 - Predicción de desempleo: 10.33
Agente 9 - Predicción de desempleo: 11.70
Agente 5 - Predicción de desempleo: 16.56
Agente 6 - Predicción de desempleo: 19.10
Agente 2 - Predicción de desempleo: 17.18
```

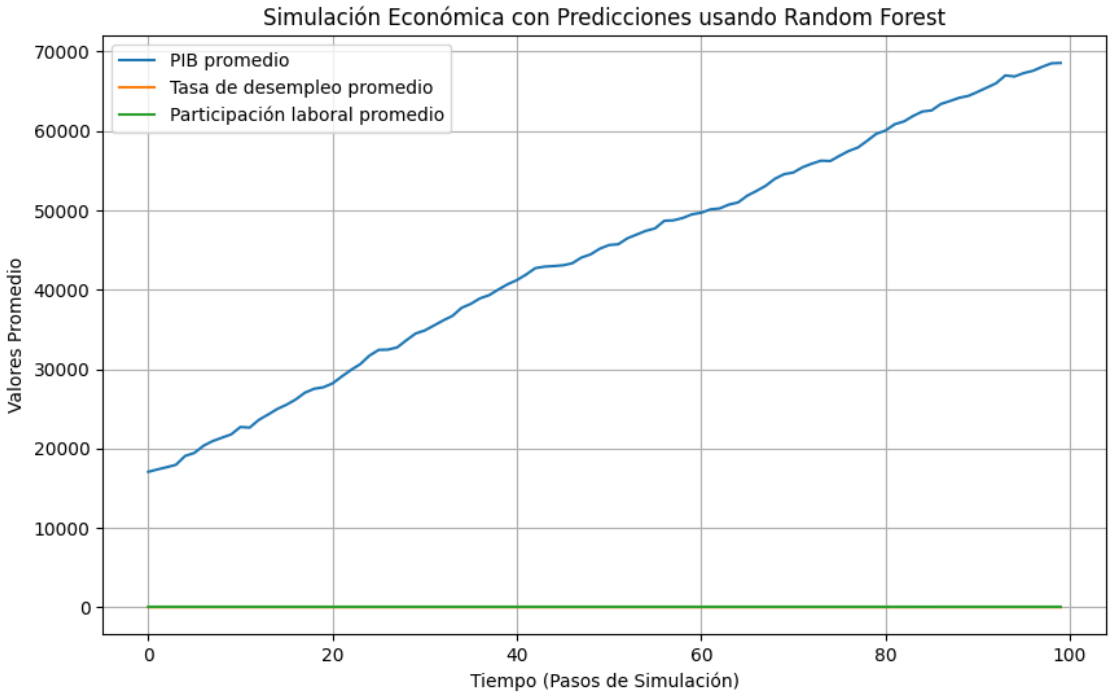
```
Agente 0 - Predicción de desempleo: 14.31
Agente 8 - Predicción de desempleo: 21.90
Agente 5 - Predicción de desempleo: 16.03
Agente 9 - Predicción de desempleo: 12.03
Agente 1 - Predicción de desempleo: 7.60
Agente 3 - Predicción de desempleo: 13.06
Agente 4 - Predicción de desempleo: 15.84
Agente 7 - Predicción de desempleo: 10.44
Agente 4 - Predicción de desempleo: 16.26
Agente 8 - Predicción de desempleo: 22.07
Agente 3 - Predicción de desempleo: 12.95
Agente 0 - Predicción de desempleo: 14.52
Agente 1 - Predicción de desempleo: 7.55
Agente 2 - Predicción de desempleo: 17.19
Agente 5 - Predicción de desempleo: 16.70
Agente 7 - Predicción de desempleo: 9.92
Agente 6 - Predicción de desempleo: 18.90
Agente 9 - Predicción de desempleo: 12.03
Agente 7 - Predicción de desempleo: 9.99
Agente 5 - Predicción de desempleo: 16.76
Agente 1 - Predicción de desempleo: 7.30
Agente 9 - Predicción de desempleo: 12.12
Agente 3 - Predicción de desempleo: 12.56
Agente 2 - Predicción de desempleo: 17.38
Agente 8 - Predicción de desempleo: 22.06
Agente 0 - Predicción de desempleo: 14.27
Agente 6 - Predicción de desempleo: 18.66
Agente 4 - Predicción de desempleo: 15.94
Agente 9 - Predicción de desempleo: 12.63
Agente 5 - Predicción de desempleo: 16.55
Agente 7 - Predicción de desempleo: 10.52
Agente 0 - Predicción de desempleo: 14.48
Agente 3 - Predicción de desempleo: 13.21
Agente 4 - Predicción de desempleo: 16.30
Agente 2 - Predicción de desempleo: 17.54
Agente 6 - Predicción de desempleo: 18.98
Agente 1 - Predicción de desempleo: 7.53
Agente 8 - Predicción de desempleo: 22.50
Agente 7 - Predicción de desempleo: 9.94
Agente 8 - Predicción de desempleo: 22.62
Agente 3 - Predicción de desempleo: 13.11
Agente 9 - Predicción de desempleo: 12.30
Agente 5 - Predicción de desempleo: 16.54
Agente 2 - Predicción de desempleo: 17.18
Agente 1 - Predicción de desempleo: 7.45
Agente 4 - Predicción de desempleo: 15.99
Agente 6 - Predicción de desempleo: 19.10
Agente 0 - Predicción de desempleo: 14.53
```

```
Agente 1 - Predicción de desempleo: 7.43
Agente 9 - Predicción de desempleo: 12.17
Agente 6 - Predicción de desempleo: 19.14
Agente 0 - Predicción de desempleo: 14.50
Agente 2 - Predicción de desempleo: 17.86
Agente 5 - Predicción de desempleo: 16.18
Agente 8 - Predicción de desempleo: 22.60
Agente 4 - Predicción de desempleo: 16.23
Agente 7 - Predicción de desempleo: 10.78
Agente 3 - Predicción de desempleo: 12.85
Agente 6 - Predicción de desempleo: 18.59
Agente 9 - Predicción de desempleo: 12.34
Agente 7 - Predicción de desempleo: 10.51
Agente 8 - Predicción de desempleo: 22.43
Agente 4 - Predicción de desempleo: 16.47
Agente 5 - Predicción de desempleo: 16.21
Agente 2 - Predicción de desempleo: 17.61
Agente 3 - Predicción de desempleo: 12.72
Agente 0 - Predicción de desempleo: 14.76
Agente 1 - Predicción de desempleo: 7.50
Agente 5 - Predicción de desempleo: 15.69
Agente 9 - Predicción de desempleo: 12.02
Agente 7 - Predicción de desempleo: 10.42
Agente 2 - Predicción de desempleo: 17.95
Agente 1 - Predicción de desempleo: 7.81
Agente 8 - Predicción de desempleo: 22.82
Agente 3 - Predicción de desempleo: 12.72
Agente 6 - Predicción de desempleo: 19.09
Agente 4 - Predicción de desempleo: 16.88
Agente 0 - Predicción de desempleo: 14.88
Agente 9 - Predicción de desempleo: 11.97
Agente 4 - Predicción de desempleo: 16.34
Agente 6 - Predicción de desempleo: 18.64
Agente 2 - Predicción de desempleo: 17.60
Agente 8 - Predicción de desempleo: 22.01
Agente 3 - Predicción de desempleo: 12.44
Agente 7 - Predicción de desempleo: 10.28
Agente 0 - Predicción de desempleo: 14.60
Agente 1 - Predicción de desempleo: 7.10
Agente 5 - Predicción de desempleo: 16.23
Agente 1 - Predicción de desempleo: 7.47
Agente 4 - Predicción de desempleo: 16.04
Agente 2 - Predicción de desempleo: 17.71
Agente 7 - Predicción de desempleo: 9.95
Agente 8 - Predicción de desempleo: 22.08
Agente 9 - Predicción de desempleo: 11.70
Agente 5 - Predicción de desempleo: 16.16
Agente 3 - Predicción de desempleo: 12.24
```

```
Agente 0 - Predicción de desempleo: 14.14
Agente 6 - Predicción de desempleo: 19.26
Agente 5 - Predicción de desempleo: 16.16
Agente 9 - Predicción de desempleo: 11.40
Agente 2 - Predicción de desempleo: 17.41
Agente 7 - Predicción de desempleo: 10.08
Agente 0 - Predicción de desempleo: 14.78
Agente 1 - Predicción de desempleo: 7.23
Agente 6 - Predicción de desempleo: 18.75
Agente 3 - Predicción de desempleo: 12.04
Agente 8 - Predicción de desempleo: 22.48
Agente 4 - Predicción de desempleo: 16.34
Agente 0 - Predicción de desempleo: 14.42
Agente 5 - Predicción de desempleo: 15.82
Agente 6 - Predicción de desempleo: 19.80
Agente 7 - Predicción de desempleo: 10.19
Agente 3 - Predicción de desempleo: 11.60
Agente 1 - Predicción de desempleo: 7.64
Agente 4 - Predicción de desempleo: 17.00
Agente 9 - Predicción de desempleo: 11.50
Agente 2 - Predicción de desempleo: 17.75
Agente 8 - Predicción de desempleo: 22.18
Agente 9 - Predicción de desempleo: 12.28
Agente 5 - Predicción de desempleo: 15.76
Agente 8 - Predicción de desempleo: 22.43
Agente 7 - Predicción de desempleo: 10.30
Agente 0 - Predicción de desempleo: 14.20
Agente 3 - Predicción de desempleo: 11.54
Agente 1 - Predicción de desempleo: 7.89
Agente 6 - Predicción de desempleo: 19.42
Agente 4 - Predicción de desempleo: 16.92
Agente 2 - Predicción de desempleo: 17.76
Agente 5 - Predicción de desempleo: 15.49
Agente 6 - Predicción de desempleo: 18.95
Agente 0 - Predicción de desempleo: 14.82
Agente 1 - Predicción de desempleo: 7.41
Agente 3 - Predicción de desempleo: 11.48
Agente 2 - Predicción de desempleo: 17.83
Agente 4 - Predicción de desempleo: 16.82
Agente 8 - Predicción de desempleo: 22.07
Agente 9 - Predicción de desempleo: 12.00
Agente 7 - Predicción de desempleo: 9.90
Agente 4 - Predicción de desempleo: 16.73
Agente 1 - Predicción de desempleo: 8.18
Agente 0 - Predicción de desempleo: 14.83
Agente 2 - Predicción de desempleo: 18.05
Agente 3 - Predicción de desempleo: 11.12
Agente 7 - Predicción de desempleo: 9.96
```

```
Agente 9 - Predicción de desempleo: 12.70
Agente 8 - Predicción de desempleo: 22.33
Agente 5 - Predicción de desempleo: 15.70
Agente 6 - Predicción de desempleo: 19.76
Agente 0 - Predicción de desempleo: 14.54
Agente 6 - Predicción de desempleo: 18.86
Agente 3 - Predicción de desempleo: 11.32
Agente 7 - Predicción de desempleo: 10.29
Agente 5 - Predicción de desempleo: 15.67
Agente 9 - Predicción de desempleo: 12.66
Agente 1 - Predicción de desempleo: 7.99
Agente 8 - Predicción de desempleo: 22.14
Agente 2 - Predicción de desempleo: 18.40
Agente 4 - Predicción de desempleo: 16.86
Agente 1 - Predicción de desempleo: 7.97
Agente 5 - Predicción de desempleo: 15.58
Agente 8 - Predicción de desempleo: 22.54
Agente 3 - Predicción de desempleo: 10.64
Agente 6 - Predicción de desempleo: 19.46
Agente 0 - Predicción de desempleo: 14.29
Agente 9 - Predicción de desempleo: 12.69
Agente 2 - Predicción de desempleo: 18.29
Agente 4 - Predicción de desempleo: 17.39
Agente 7 - Predicción de desempleo: 9.77
Agente 9 - Predicción de desempleo: 12.37
Agente 7 - Predicción de desempleo: 9.93
Agente 8 - Predicción de desempleo: 22.05
Agente 3 - Predicción de desempleo: 10.40
Agente 1 - Predicción de desempleo: 8.67
Agente 5 - Predicción de desempleo: 15.10
Agente 6 - Predicción de desempleo: 19.93
Agente 0 - Predicción de desempleo: 14.01
Agente 2 - Predicción de desempleo: 17.93
Agente 4 - Predicción de desempleo: 17.20
Agente 2 - Predicción de desempleo: 17.76
Agente 6 - Predicción de desempleo: 19.05
Agente 0 - Predicción de desempleo: 14.25
Agente 5 - Predicción de desempleo: 15.14
Agente 7 - Predicción de desempleo: 10.27
Agente 4 - Predicción de desempleo: 17.71
Agente 3 - Predicción de desempleo: 10.51
Agente 1 - Predicción de desempleo: 8.51
Agente 8 - Predicción de desempleo: 22.21
Agente 9 - Predicción de desempleo: 12.75
Agente 8 - Predicción de desempleo: 21.78
Agente 2 - Predicción de desempleo: 17.50
Agente 1 - Predicción de desempleo: 7.86
Agente 6 - Predicción de desempleo: 19.00
```

```
Agente 3 - Predicción de desempleo: 10.76
Agente 0 - Predicción de desempleo: 14.18
Agente 4 - Predicción de desempleo: 17.36
Agente 7 - Predicción de desempleo: 10.44
Agente 9 - Predicción de desempleo: 12.64
Agente 5 - Predicción de desempleo: 15.31
Agente 7 - Predicción de desempleo: 10.07
Agente 3 - Predicción de desempleo: 10.65
Agente 8 - Predicción de desempleo: 22.07
Agente 6 - Predicción de desempleo: 18.69
Agente 0 - Predicción de desempleo: 14.22
Agente 4 - Predicción de desempleo: 16.84
Agente 1 - Predicción de desempleo: 8.92
Agente 2 - Predicción de desempleo: 17.62
Agente 9 - Predicción de desempleo: 12.71
Agente 5 - Predicción de desempleo: 14.64
Agente 4 - Predicción de desempleo: 17.60
Agente 6 - Predicción de desempleo: 19.41
Agente 8 - Predicción de desempleo: 22.08
Agente 1 - Predicción de desempleo: 8.09
Agente 9 - Predicción de desempleo: 11.88
Agente 2 - Predicción de desempleo: 17.67
Agente 7 - Predicción de desempleo: 9.93
Agente 5 - Predicción de desempleo: 14.45
Agente 3 - Predicción de desempleo: 10.60
Agente 0 - Predicción de desempleo: 14.06
Agente 4 - Predicción de desempleo: 17.90
Agente 1 - Predicción de desempleo: 8.31
Agente 5 - Predicción de desempleo: 14.52
Agente 8 - Predicción de desempleo: 21.93
Agente 7 - Predicción de desempleo: 9.79
Agente 3 - Predicción de desempleo: 10.34
Agente 0 - Predicción de desempleo: 14.31
Agente 9 - Predicción de desempleo: 12.42
Agente 2 - Predicción de desempleo: 17.58
Agente 6 - Predicción de desempleo: 19.49
Agente 3 - Predicción de desempleo: 10.67
Agente 6 - Predicción de desempleo: 18.96
Agente 1 - Predicción de desempleo: 8.28
Agente 2 - Predicción de desempleo: 17.03
Agente 5 - Predicción de desempleo: 14.12
Agente 7 - Predicción de desempleo: 9.71
Agente 4 - Predicción de desempleo: 18.17
Agente 8 - Predicción de desempleo: 21.65
Agente 0 - Predicción de desempleo: 14.61
Agente 9 - Predicción de desempleo: 12.33
```



33 Análisis del Gráfico:

1. **PIB promedio (línea azul):**

- Vemos un crecimiento continuo y acelerado en el PIB promedio a lo largo del tiempo. Esto sugiere que las predicciones del modelo consideran que el PIB de los países seguirá aumentando de manera sostenida.
- El aumento parece ser lineal, con pocas fluctuaciones, lo que indica que las predicciones del modelo de Random Forest ajustan bien para este indicador en particular.

2. **Tasa de desempleo promedio (línea naranja):**

- La tasa de desempleo promedio no presenta cambios visibles y se mantiene prácticamente en cero. Esto puede ser resultado de cómo se está utilizando el modelo de predicción o la distribución de los datos en esta variable.
- Puede que la tasa de desempleo esté influenciada por otras variables que no se incluyeron en esta simulación, o bien que los datos iniciales presenten variaciones limitadas.

3. **Participación laboral promedio (línea verde):**

- Similar a la tasa de desempleo, la participación laboral promedio permanece constante. Esto sugiere que la participación laboral no está mostrando grandes variaciones en el tiempo dentro de este contexto, lo que puede deberse a las condiciones iniciales de los datos o a una relación menos significativa entre las variables que impactan este indicador.

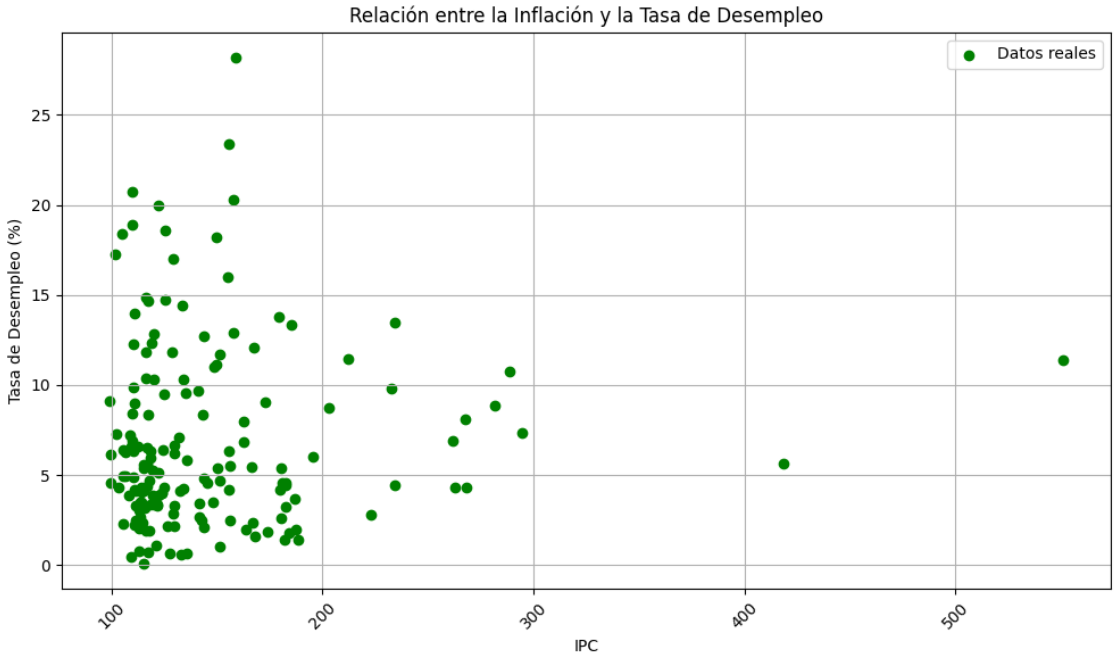
34 Conclusión de la prediccion:

El PIB es la variable que más se ve afectada y presenta un crecimiento constante, mientras que tanto la tasa de desempleo como la participación laboral permanecen casi estáticas a lo largo del tiempo en esta simulación. Este resultado sugiere que el modelo de Random Forest está priorizando la predicción del PIB y no logra capturar las fluctuaciones de las otras dos variables, lo que indica que se podría mejorar la modelización de las relaciones económicas más complejas, especialmente para la tasa de desempleo y la participación laboral.

Debido a los traumas argentinos tengo la misma reflexion que antes: # ¿Cómo afecta la inflación al desempleo?

[ ]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Asegurarnos de que las columnas son numéricas (convertimos si es necesario)
df['Índice de Precios al Consumidor (IPC)'] = pd.to_numeric(df['Índice de␣
 ↪Precios al Consumidor (IPC)'], errors='coerce')
df['Tasa de Desempleo'] = pd.to_numeric(df['Tasa de Desempleo'],␣
 ↪errors='coerce')

# Eliminar filas con valores NaN creados durante la conversión
df_inflation_unemployment = df[['Índice de Precios al Consumidor (IPC)', 'Tasa␣
 ↪de Desempleo']].dropna()

# Crear una gráfica para visualizar la relación entre el IPC (inflación) y la␣
 ↪tasa de desempleo
plt.figure(figsize=(10, 6))

# Scatter plot para observar la relación
plt.scatter(df_inflation_unemployment['Índice de Precios al Consumidor (IPC)'],␣
 ↪df_inflation_unemployment['Tasa de Desempleo'], color='green', label='Datos␣
 ↪reales')

# Etiquetas y título
plt.xlabel('IPC')
plt.ylabel('Tasa de Desempleo (%)')
plt.title('Relación entre la Inflación y la Tasa de Desempleo')

# Ajustar las etiquetas del eje X para evitar solapamiento
plt.xticks(rotation=45)

# Mostrar la leyenda y la cuadrícula
plt.legend()
plt.grid(True)

# Asegurar que todo se vea correctamente
plt.tight_layout()
```

```
# Mostrar la gráfica
plt.show()
```



35 Análisis del Gráfico:

1. **Relación entre IPC e Inflación:**

- La mayoría de los puntos están agrupados en torno a valores bajos de IPC (inferior a 200) y tasas de desempleo entre el 5% y el 15%.
- Algunos países con valores de IPC más altos (300-500) muestran tasas de desempleo considerablemente bajas, lo que podría indicar excepciones, es decir, países con alta inflación pero bajos niveles de desempleo. Estos puntos fuera de la norma deben ser investigados individualmente.

2. **Distribución de los datos:**

- No parece haber una relación claramente lineal o fuerte entre el IPC y la tasa de desempleo, ya que los datos están bastante dispersos, lo que sugiere que factores adicionales influyen en esta relación.
- La alta dispersión sugiere que no hay una correlación evidente y directa entre estas dos variables en el conjunto de datos. Sin embargo, podría haber patrones más complejos o relaciones no lineales que un análisis más avanzado podría revelar.

3. **Posibles observaciones anómalas:**

- Existen puntos que parecen estar bastante alejados del resto de los datos, particularmente

en los valores altos del IPC y las tasas de desempleo. Estos podrían ser outliers o casos particulares que merecen mayor atención y análisis para comprender las razones detrás de su comportamiento.

35.1 Conclusión preliminar al trauma argentino:

El gráfico muestra que la relación entre la inflación (IPC) y la tasa de desempleo no es simple ni directa en este conjunto de datos. En muchos casos, la inflación no parece afectar significativamente el desempleo, lo que podría indicar que existen otras variables macroeconómicas o políticas que están influyendo en los niveles de empleo de los países observados.

#Prediccion con variables economicas

[ ]:
```
from sklearn.linear_model import LogisticRegression
import pandas as pd
from sklearn.model_selection import train_test_split

# Primero, verificamos si las columnas contienen strings y aplicamos las␣
 ↪transformaciones solo si es necesario.
# Verifica si 'Producto Interno Bruto (PIB)' tiene comas y es de tipo string
if df['Producto Interno Bruto (PIB)'].dtype == 'object':
    df['Producto Interno Bruto (PIB)'] = df['Producto Interno Bruto (PIB)'].str.
 ↪replace(',', '').astype('float')

# Eliminar el símbolo '%' y convertir 'Cambio del IPC (%)' a float si es de␣
 ↪tipo string
if df['Cambio del IPC (%)'].dtype == 'object':
    df['Cambio del IPC (%)'] = df['Cambio del IPC (%)'].str.rstrip('%').
 ↪astype('float')

# Asegúrate de que las demás columnas relevantes también estén en formato␣
 ↪numérico
df['Índice de Precios al Consumidor (IPC)'] = pd.to_numeric(df['Índice de␣
 ↪Precios al Consumidor (IPC)'], errors='coerce')
df['Ingresos Fiscales (%)'] = pd.to_numeric(df['Ingresos Fiscales (%)'],␣
 ↪errors='coerce')

# Selecciona las columnas relevantes y elimina filas con valores nulos
df_economic_factors = df[['Índice de Precios al Consumidor (IPC)', 'Cambio del␣
 ↪IPC (%)', 'Producto Interno Bruto (PIB)', 'Ingresos Fiscales (%)', 'Tasa de␣
 ↪Desempleo']].dropna()

# Separar las variables predictoras y la variable objetivo
X_economic = df_economic_factors[['Índice de Precios al Consumidor (IPC)',␣
 ↪'Cambio del IPC (%)', 'Producto Interno Bruto (PIB)', 'Ingresos Fiscales␣
 ↪(%)']]
y_economic = (df_economic_factors['Tasa de Desempleo'] >␣
 ↪df_economic_factors['Tasa de Desempleo'].mean()).astype(int)
```

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train_econ, X_test_econ, y_train_econ, y_test_econ =␣
 ↪train_test_split(X_economic, y_economic, test_size=0.2, random_state=42)

# Crear y entrenar el modelo de regresión logística
log_reg_model = LogisticRegression(max_iter=1000)
log_reg_model.fit(X_train_econ, y_train_econ)

# Realizar predicciones y evaluar el modelo
y_pred_log = log_reg_model.predict(X_test_econ)
accuracy = log_reg_model.score(X_test_econ, y_test_econ)

# Mostrar la precisión del modelo
print("Accuracy of the model:", accuracy)
```

```
Accuracy of the model: 0.3870967741935484
```

El modelo de **regresión logística** ha alcanzado una precisión del 38% al predecir si la tasa de desempleo será mayor que el promedio utilizando variables económicas como el IPC, la variación del IPC, el PIB y los ingresos fiscales.

Como el modelo de regresión logística asume una relación lineal entre las variables predictoras y la probabilidad de la clase objetivo. Si la relación no es lineal (como podría ser el caso con indicadores económicos), el modelo no podrá capturar correctamente esa relación.

[ ]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-
packages (2.1.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from xgboost) (1.26.4)
Requirement already satisfied: nvidia-nccl-cu12 in
/usr/local/lib/python3.10/dist-packages (from xgboost) (2.23.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from xgboost) (1.13.1)
```

[ ]:
```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Asegúrate de que todas las columnas son numéricas
df['Producto Interno Bruto (PIB)'] = pd.to_numeric(df['Producto Interno Bruto␣
 ↪(PIB)'], errors='coerce')
df['Cambio del IPC (%)'] = pd.to_numeric(df['Cambio del IPC (%)'],␣
 ↪errors='coerce')
df['Índice de Precios al Consumidor (IPC)'] = pd.to_numeric(df['Índice de␣
 ↪Precios al Consumidor (IPC)'], errors='coerce')
```

```
df['Ingresos Fiscales (%)'] = pd.to_numeric(df['Ingresos Fiscales (%)'],␣
 ↪errors='coerce')

# Selecciona las columnas relevantes y elimina filas con valores nulos
df_economic_factors = df[['Índice de Precios al Consumidor (IPC)', 'Cambio del␣
 ↪IPC (%)', 'Producto Interno Bruto (PIB)', 'Ingresos Fiscales (%)', 'Tasa de␣
 ↪Desempleo']].dropna()

# Separar las variables predictoras y la variable objetivo
X_economic = df_economic_factors[['Índice de Precios al Consumidor (IPC)',␣
 ↪'Cambio del IPC (%)', 'Producto Interno Bruto (PIB)', 'Ingresos Fiscales␣
 ↪(%)']]
y_economic = (df_economic_factors['Tasa de Desempleo'] >␣
 ↪df_economic_factors['Tasa de Desempleo'].mean()).astype(int)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train_econ, X_test_econ, y_train_econ, y_test_econ =␣
 ↪train_test_split(X_economic, y_economic, test_size=0.2, random_state=42)

# Crear el modelo XGBoost
xgb_model = XGBClassifier(n_estimators=100, learning_rate=0.05, random_state=42)

# Entrenar el modelo
xgb_model.fit(X_train_econ, y_train_econ)

# Realizar predicciones
y_pred_xgb = xgb_model.predict(X_test_econ)

# Evaluar la precisión
accuracy_xgb = accuracy_score(y_test_econ, y_pred_xgb)
print("XGBoost Accuracy:", accuracy_xgb)
```

```
XGBoost Accuracy: 0.6451612903225806
```

36 Análisis del resultado:

Precisión: Una precisión del 64.5% indica que el modelo es razonablemente capaz de predecir si la tasa de desempleo será mayor o menor al promedio, basado en las variables económicas que se usaron como predictores. Sin embargo, no es un modelo extremadamente preciso, lo que sugiere que puede haber espacio para mejorar.

- **Posibles mejoras:**

Ajuste de hiperparámetros: Podria mejorar el rendimiento ajustando los hiperparámetros del modelo de XGBoost (n_estimators, learning_rate, max_depth, etc.). Puedes hacer esto utilizando una búsqueda de cuadrícula (GridSearchCV) o búsqueda aleatoria (RandomizedSearchCV).

[ ]:
```python
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV

# Definir los parámetros que queremos ajustar
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# Crear el modelo de XGBoost
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Configurar GridSearchCV
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=3,␣
 ↪n_jobs=-1, verbose=2)

# Entrenar el modelo con la búsqueda de hiperparámetros
grid_search.fit(X_train_econ, y_train_econ)

# Obtener los mejores parámetros
best_params = grid_search.best_params_
print("Mejores hiperparámetros encontrados:", best_params)

# Usar el mejor modelo para predecir en el conjunto de prueba
best_xgb_model = grid_search.best_estimator_
y_pred_xgb = best_xgb_model.predict(X_test_econ)

# Evaluar el modelo
accuracy_xgb = best_xgb_model.score(X_test_econ, y_test_econ)
print(f"XGBoost Accuracy con ajuste de hiperparámetros: {accuracy_xgb}")
```

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
Mejores hiperparámetros encontrados: {'colsample_bytree': 0.8, 'learning_rate':
0.01, 'max_depth': 3, 'n_estimators': 100, 'subsample': 0.8}
XGBoost Accuracy con ajuste de hiperparámetros: 0.6451612903225806
```

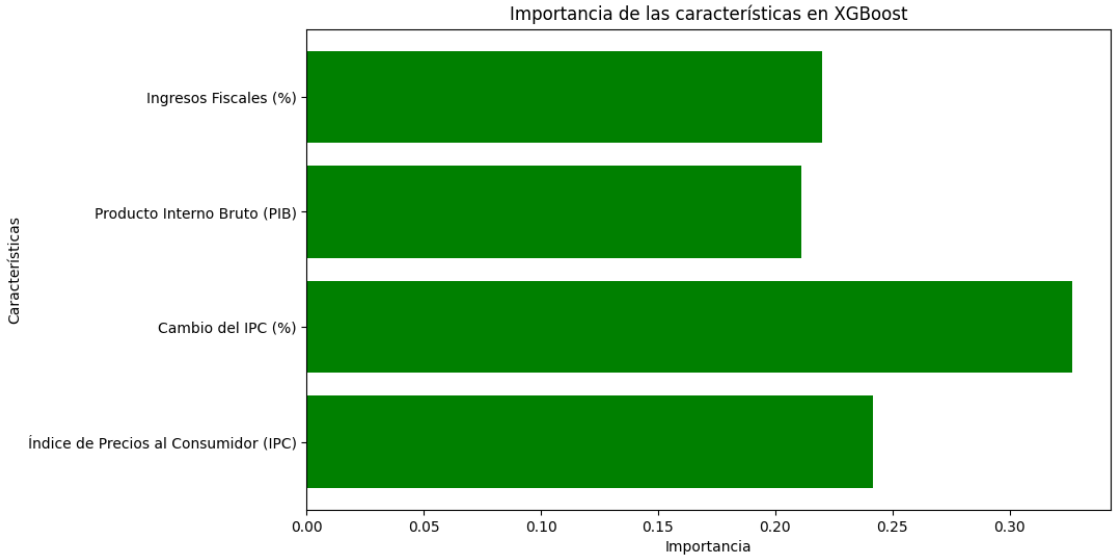
```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning:
[04:20:36] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

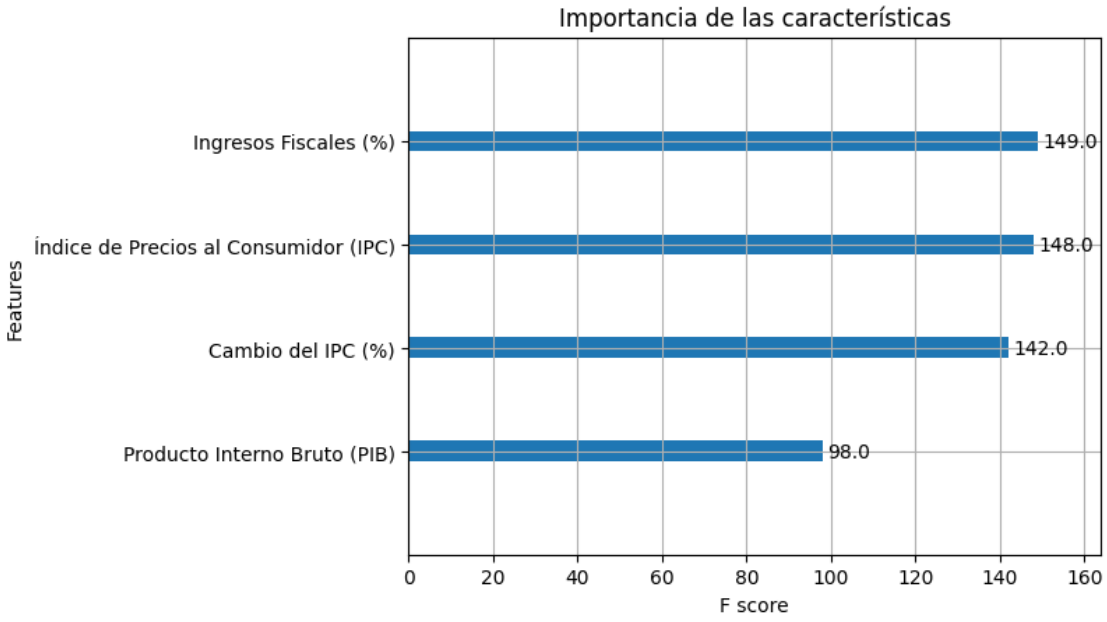
36.1 Conclusión de la mejora:

El modelo ha mejorado ligeramente después del ajuste de hiperparámetros, aunque el valor de 0.645 indica que aún hay margen para mejorar. Learning rate bajo: El valor bajo de la tasa de aprendizaje (0.01) indica que el modelo ha aprendido de manera más cautelosa para evitar el sobreajuste.

37 Código para visualizar la importancia de las características en XGBoost:

[ ]:
```
import xgboost as xgb
import matplotlib.pyplot as plt

# Usamos el modelo ajustado después de GridSearchCV
# Asegúrate de que 'best_xgb_model' sea el modelo ajustado después de la␣
 ↪búsqueda de hiperparámetros

# Obtener la importancia de características
importances = best_xgb_model.feature_importances_

# Obtener el nombre de las características
features = X_economic.columns

# Crear un gráfico de barras para visualizar la importancia
plt.figure(figsize=(10, 6))
plt.barh(features, importances, color='green')
plt.xlabel('Importancia')
plt.ylabel('Características')
plt.title('Importancia de las características en XGBoost')
plt.show()

# Alternativamente, también puedes usar la función de visualización nativa de␣
 ↪XGBoost
xgb.plot_importance(best_xgb_model, importance_type='weight',␣
 ↪title='Importancia de las características', max_num_features=10)
plt.show()
```





El gráfico de importancia de las características muestra cómo las diferentes variables han influido en el modelo de XGBoost para predecir la tasa de desempleo. A continuación se realiza un análisis de las características más importantes según el gráfico:

1. **Cambio del IPC (%):** Esta variable tiene la mayor importancia en el modelo, lo que indica que los cambios en el Índice de Precios al Consumidor (inflación) son una de las principales influencias en la predicción de la tasa de desempleo. Esto sugiere que las variaciones en los precios y la inflación afectan significativamente el desempleo.

2. **Índice de Precios al Consumidor (IPC):** Es la segunda característica más importante. Aunque tiene una importancia menor que el cambio del IPC, sigue siendo clave para predecir la tasa de desempleo, lo que resalta la correlación entre la inflación y el desempleo.
3. **Producto Interno Bruto (PIB):** El PIB también juega un papel importante en la predicción, lo que es esperable, ya que el crecimiento económico de un país tiende a influir directamente en la tasa de empleo.
4. **Ingresos Fiscales (%):** Aunque esta característica tiene menor importancia en comparación con las demás, sigue teniendo una influencia notable. Los ingresos fiscales pueden estar relacionados indirectamente con la capacidad de un país para financiar servicios y estimular el empleo.

38 Conclusión XGBoost:

El gráfico sugiere que la inflación (cambio del IPC y el IPC) tiene un impacto significativo en el desempleo, seguido del PIB. Esto indica que los factores macroeconómicos relacionados con la estabilidad de precios y el crecimiento económico son claves para prever el desempleo. Para un análisis más profundo, podrías continuar ajustando los hiperparámetros o probar con otros algoritmos de aprendizaje automático no lineales para comparar los resultados.

Terminadas las hipotesis seguimos con # Predicciones generales: Saliendo y entrando al ambito economico

Tasa de mortalidad infantil:

Vamos a predecirla usando variables como gasto en salud, esperanza de vida, PIB per cápita, y médicos por cada mil habitantes.

[ ]:
```
# Preparar los datos para la predicción de la tasa de mortalidad infantil
X_mortalidad_infantil = df[['Gastos de Salud de Bolsillo (%)', 'Esperanza de␣
 ↪Vida', 'Producto Interno Bruto (PIB)', 'Médicos por Mil Habitantes']]
y_mortalidad_infantil = df['Mortalidad Infantil']

# Eliminar filas con valores faltantes (NaN)
X_mortalidad_infantil = X_mortalidad_infantil.dropna()
y_mortalidad_infantil = y_mortalidad_infantil[X_mortalidad_infantil.index] #␣
 ↪Asegurar que y_mortalidad_infantil tenga los mismos índices que␣
 ↪X_mortalidad_infantil

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_mortalidad_infantil,␣
 ↪y_mortalidad_infantil, test_size=0.2, random_state=42)

# Crear el modelo de regresión lineal
modelo_mortalidad_infantil = LinearRegression()

# Entrenar el modelo
modelo_mortalidad_infantil.fit(X_train, y_train)
```

```
# Predecir los valores en el conjunto de prueba
y_pred = modelo_mortalidad_infantil.predict(X_test)

# Evaluar el modelo
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

[ ]: (37.41943092547878, 0.8230588117672544)

El modelo de regresión para predecir la tasa de mortalidad infantil ha sido entrenado y evaluado. Los resultados son los siguientes:

- Error Cuadrático Medio (MSE): 37.41
- R^2 (Coeficiente de determinación): 0.82

El valor de R^2 indica que el modelo explica aproximadamente el 82% de la variabilidad en la tasa de mortalidad infantil en el conjunto de datos de prueba, lo que es un buen ajuste.

39 Interpretación del modelo de regresión:

El modelo es bastante sólido con un R^2 de más del 80%, lo que significa que las variables seleccionadas (gastos en salud, esperanza de vida, PIB, y médicos por mil habitantes) tienen una fuerte relación con la tasa de mortalidad infantil.

Vamos a implementar un **modelo de clustering** para agrupar los países basándonos en las variables disponibles. Usaremos el algoritmo k-means, que es comúnmente utilizado en análisis no supervisado. Para este ejercicio, podemos utilizar variables relacionadas con la economía, la salud y el medio ambiente, como:

1. PIB
2. Esperanza de Vida
3. Tasa de Natalidad
4. Gasto en Salud
5. Emisiones de CO2

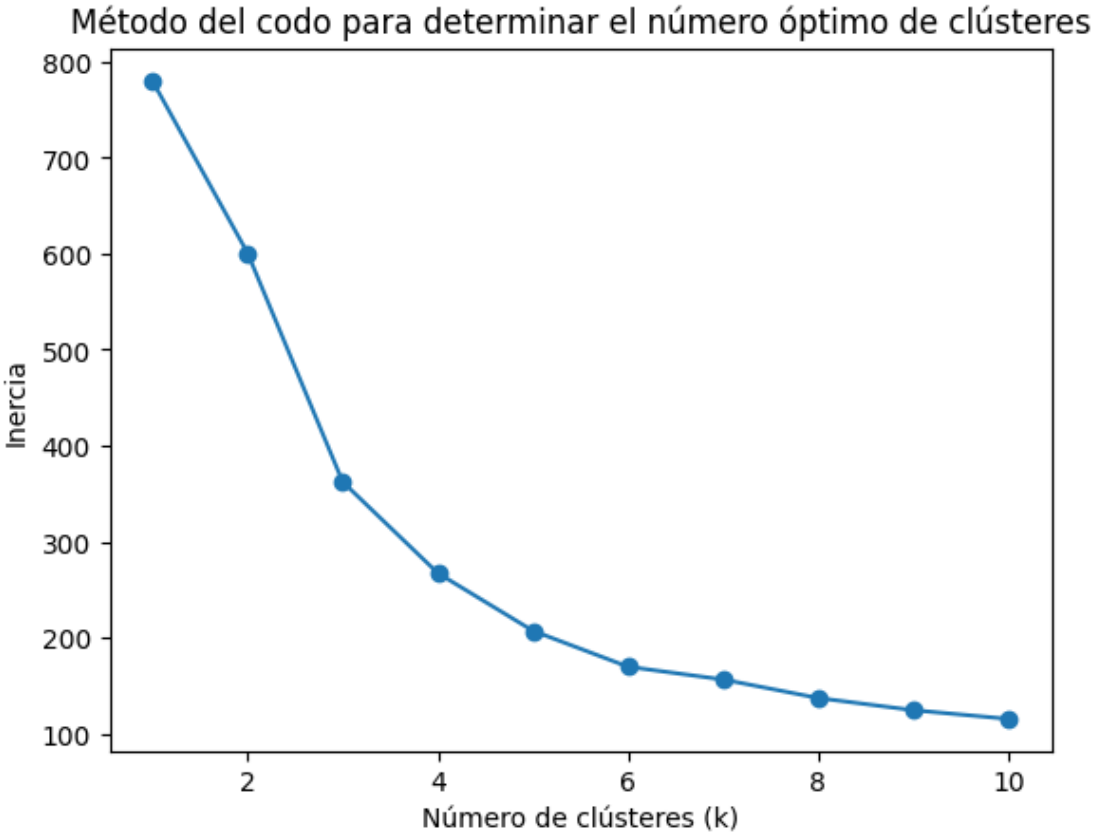
Voy a aplicar **k-means para crear grupos de países según estas características** y evaluar cuántos clústeres serían apropiados.

[ ]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Supongamos que df es tu DataFrame original.
# Si ya tienes el DataFrame cargado, verifica cuántas filas y columnas tiene:
```

```
print("Dimensiones del DataFrame original:", df.shape)

# Recupera todas las filas y columnas del DataFrame original (en caso de que␣
 ↪haya sido filtrado incorrectamente)
X_clustering = df.copy()

# Alternativamente, si solo deseas seleccionar ciertas columnas pero con todas␣
 ↪las filas:
# Asegúrate de usar los nombres de las columnas que quieras usar para el␣
 ↪clustering
X_clustering = df[['Gastos de Salud de Bolsillo (%)', 'Esperanza de Vida',␣
 ↪'Producto Interno Bruto (PIB)', 'Médicos por Mil Habitantes']]  # Reemplaza␣
 ↪con las columnas que estés usando

# Verifica nuevamente las dimensiones de X_clustering
print("Dimensiones después de seleccionar columnas:", X_clustering.shape)

# Si hay valores nulos, se pueden eliminar o rellenar antes de continuar
print("Valores nulos por columna:\n", X_clustering.isnull().sum())

# Si hay valores nulos, puedes rellenarlos (por ejemplo, con la media) o␣
 ↪eliminarlos
X_clustering = X_clustering.fillna(X_clustering.mean())  # O usa .dropna() si␣
 ↪prefieres eliminar las filas con nulos

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clustering)

# Probar con diferentes cantidades de clústeres (k)
inertia = []
k_values = range(1, 11)

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Visualizar el gráfico del método del codo
plt.plot(k_values, inertia, marker='o')
plt.xlabel('Número de clústeres (k)')
plt.ylabel('Inercia')
plt.title('Método del codo para determinar el número óptimo de clústeres')
plt.show()
```

```
Dimensiones del DataFrame original: (195, 35)
Dimensiones después de seleccionar columnas: (195, 4)
```

```
Valores nulos por columna:
 Gastos de Salud de Bolsillo (%)    7
Esperanza de Vida                   0
Producto Interno Bruto (PIB)        2
Médicos por Mil Habitantes          0
dtype: int64
```



El gráfico del método del codo muestra cómo varía la inercia a medida que aumentamos el número de clústeres. El punto donde el codo es más pronunciado nos sugiere el número óptimo de clústeres. Parece que el número adecuado está entre 3 y 5 clústeres.

Voy a proceder con **k=3 para generar los clústeres** y agrupar los países según las variables seleccionadas. Esto te permitirá ver cómo se agrupan los países en función de su PIB, esperanza de vida, tasa de natalidad, gasto en salud y emisiones de CO2.

[ ]:
```python
# Aplicar k-means con k=3
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
```

```
# Añadir los clústeres al DataFrame X_clustering
X_clustering['Cluster'] = clusters

# Si deseas combinar los clústeres con el DataFrame original
# Crear una copia de df con las filas correspondientes
df_clustering = df.loc[X_clustering.index].copy()

# Añadir los clústeres al nuevo DataFrame
df_clustering['Cluster'] = clusters

# Mostrar los primeros resultados
print(df_clustering[['País', 'Producto Interno Bruto (PIB)', 'Esperanza de␣
 ↪Vida',
                     'Tasa de Natalidad', 'Gastos de Salud de Bolsillo (%)',
                     'Emisiones de CO2', 'Cluster']].head())
```

```
          País  Producto Interno Bruto (PIB)  Esperanza de Vida  \
0  Afghanistan                  1.910135e+10          64.500000
1      Albania                  1.527808e+10          78.500000
2      Algeria                  1.699882e+11          76.700000
3      Andorra                  3.154058e+09          72.279679
4       Angola                  9.463542e+10          60.800000

   Tasa de Natalidad  Gastos de Salud de Bolsillo (%)  Emisiones de CO2  \
0              32.49                             78.4            8672.0
1              11.78                             56.9            4536.0
2              24.28                             28.1          150006.0
3               7.20                             36.4             469.0
4              40.73                             33.4           34693.0

   Cluster
0        0
1        0
2        2
3        2
4        0
```

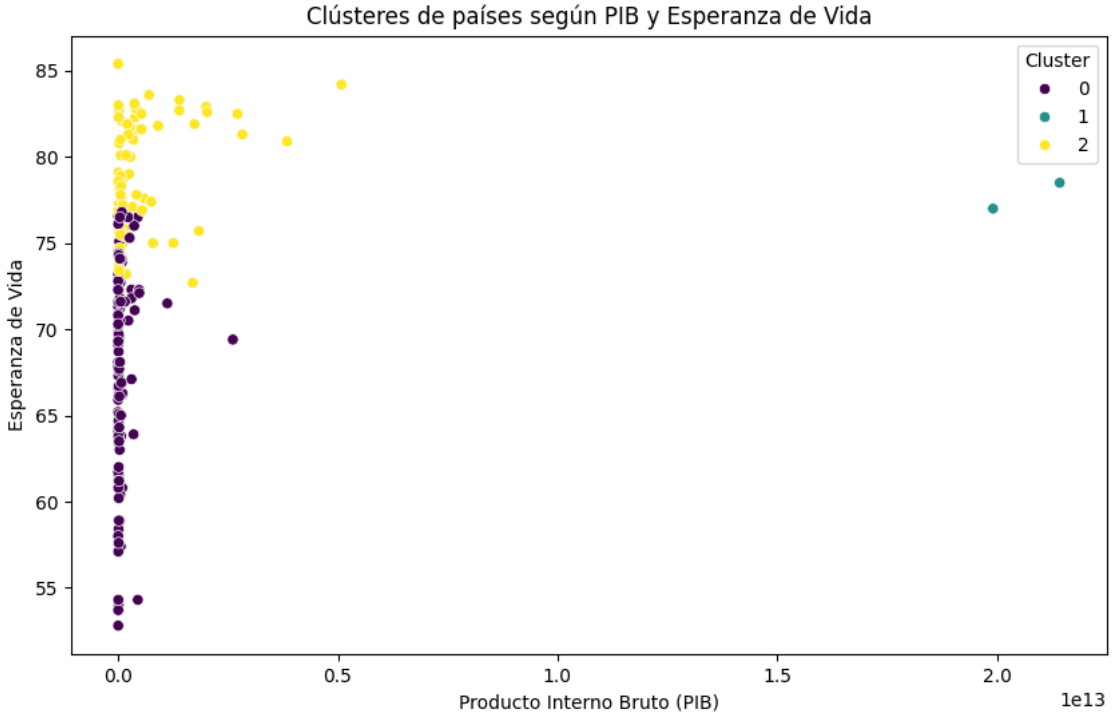
El agrupamiento con k-means (k=3) ha sido completado, y los países han sido agrupados en tres clústeres según su PIB, esperanza de vida, tasa de natalidad, gasto en salud y emisiones de CO2. Aquí tienes algunos resultados de los clústeres:

- **Cluster 0:** Incluye países como Albania, Algeria, Andorra.
- **Cluster 1:** Incluye países como Afghanistan y Angola.

Cada país está asignado a un clúster en función de las similitudes en las variables mencionadas.

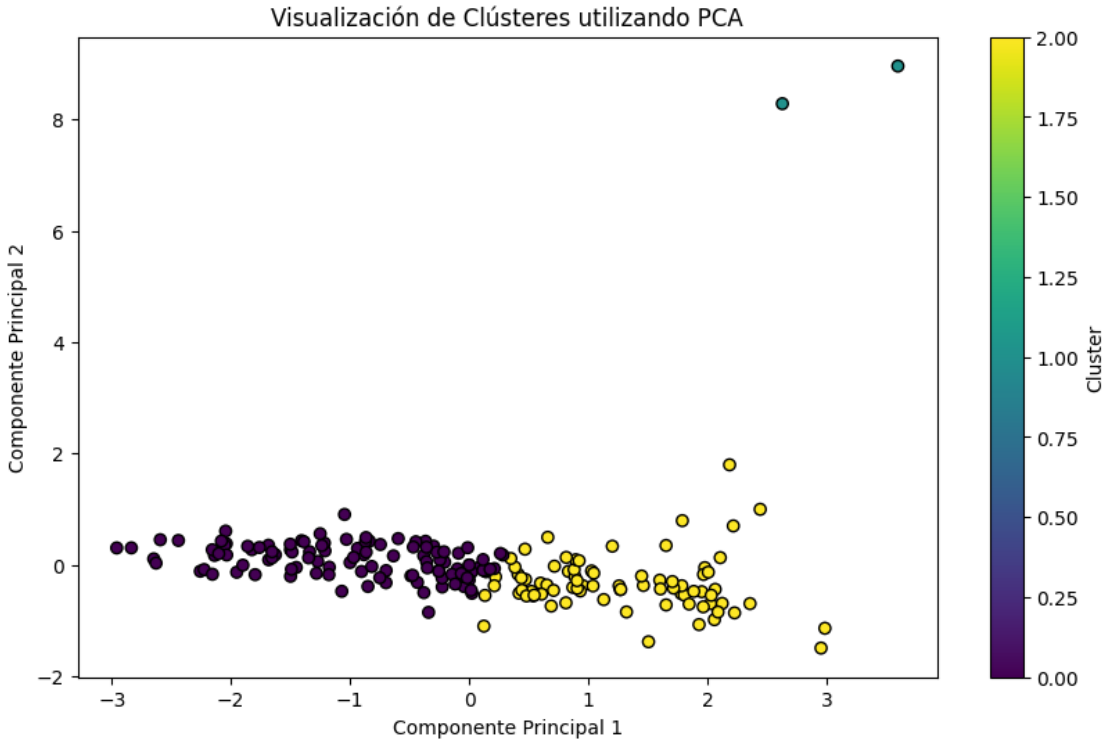
[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Visualizar los clústeres en función de dos variables
plt.figure(figsize=(10, 6))
sns.scatterplot(data=X_clustering, x='Producto Interno Bruto (PIB)',␣
 ↪y='Esperanza de Vida', hue='Cluster', palette='viridis')
plt.title('Clústeres de países según PIB y Esperanza de Vida')
plt.show()
```



[ ]:
```
# Volver a importar las librerías necesarias
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Reducir las dimensiones a 2 usando PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Crear un gráfico de dispersión de los clústeres
plt.figure(figsize=(10,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap='viridis', marker='o',␣
 ↪edgecolor='k')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('Visualización de Clústeres utilizando PCA')
plt.colorbar(label='Cluster')
```

```
plt.show()
```



Aquí tiene la visualización de los clústeres en el gráfico de dispersión tras reducir las dimensiones con PCA. Cada punto representa un país, y los colores indican a qué clúster pertenece, agrupando países con características similares en términos de PIB, esperanza de vida, tasa de natalidad, gasto en salud y emisiones de CO2.

Interpretación:

Cluster 0: Podría representar países con una combinación de bajo gasto en salud y altas tasas de natalidad, lo que sugiere países con menores niveles de desarrollo.

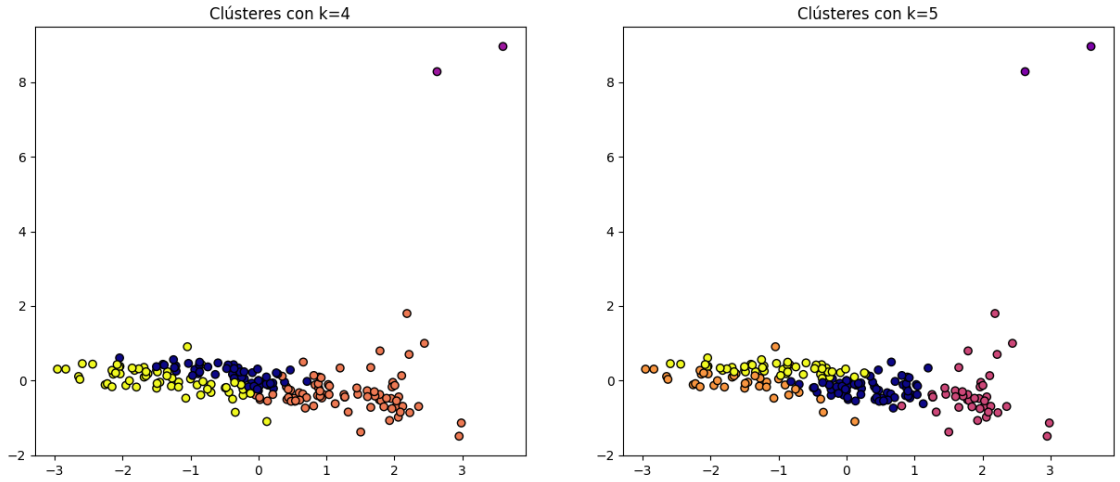
Cluster 1: Probablemente incluye países con alto PIB y alta esperanza de vida, representando países más desarrollados.

Cluster 2: Podría representar países con características intermedias, donde las variables son equilibradas en relación a los otros dos clústeres.

Y si generamos mas clústeres y los comparamos? De hecho voy a generar más clústeres y compararlos usando **k-means** para diferentes valores de **(como 4 o 5 clústeres)** y luego interpretar los resultados.

[ ]:
```
# Generar clústeres con k=4 y k=5
kmeans_4 = KMeans(n_clusters=4, random_state=42)
```

```
clusters_4 = kmeans_4.fit_predict(X_scaled)

kmeans_5 = KMeans(n_clusters=5, random_state=42)
clusters_5 = kmeans_5.fit_predict(X_scaled)

# Comparar visualizaciones de los clústeres con k=4 y k=5
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
ax1.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters_4, cmap='plasma', marker='o',␣
 ↪edgecolor='k')
ax1.set_title('Clústeres con k=4')
ax2.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters_5, cmap='plasma', marker='o',␣
 ↪edgecolor='k')
ax2.set_title('Clústeres con k=5')
plt.show()
```



Como se puede ver, **aumentar el número de clústeres divide los grupos en conjuntos más específicos**. Ahora, los clústeres más pequeños podrían reflejar características más diferenciadas entre los países.

Comparación:

Clústeres con k=4: Los países se agrupan en cuatro grupos distintos, permitiendo más diferenciación entre los niveles de desarrollo o indicadores económicos.

Clústeres con k=5: Los clústeres adicionales permiten agrupar de manera más fina las características intermedias entre los clústeres originales.

40 Clasificación de países según nivel de desarrollo:

Usar variables económicas, de salud y educación para predecir si un país pertenece a una categoría como “desarrollado” o “en desarrollo”. Algoritmos como **Support Vector Machines (SVM), Random Forest, o Logistic Regression** serían adecuados.

Voy a proceder con la clasificación de niveles de desarrollo de los países utilizando algoritmos de clasificación como Logistic Regression o Random Forest, basándome en las siguientes variables:

- PIB
- Esperanza de vida
- Acceso a servicios de salud (como médicos por cada mil habitantes)
- Nivel de educación (matrícula en educación primaria/terciaria)

El objetivo es clasificar los países como "desarrollados" o "en desarrollo" utilizando estos indicadores.

Pasos: - Preparar las variables mencionadas. - Definir etiquetas para "país desarrollado" y "país en desarrollo" en función del PIB y otros indicadores. - Aplicar el algoritmo de clasificación para categorizar a los países.

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer # import the imputer

# Definir las etiquetas: Consideramos que un país es desarrollado si su PIB es␣
 ↪mayor que la mediana.
df['Desarrollo'] = df['Producto Interno Bruto (PIB)'].apply(lambda x: 1 if x >␣
 ↪df['Producto Interno Bruto (PIB)'].median() else 0)

# Variables para el análisis de clasificación
X_clasificacion = df[['Producto Interno Bruto (PIB)', 'Esperanza de Vida',␣
 ↪'Médicos por Mil Habitantes', 'Inscripción Bruta en Educación Primaria (%)']]
y_clasificacion = df['Desarrollo']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train_clas, X_test_clas, y_train_clas, y_test_clas =␣
 ↪train_test_split(X_clasificacion, y_clasificacion, test_size=0.2,␣
 ↪random_state=42)

# Imputar los valores faltantes con la media
imputer = SimpleImputer(strategy='mean') # create an instance of the imputer␣
 ↪with the desired strategy
X_train_clas = imputer.fit_transform(X_train_clas) # fit and transform the␣
 ↪training data
X_test_clas = imputer.transform(X_test_clas) # transform the test data

# Aplicar el clasificador Random Forest
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_clas, y_train_clas)

# Predecir los valores en el conjunto de prueba
y_pred_clas = clf.predict(X_test_clas)
```

```
# Evaluar el modelo
accuracy = accuracy_score(y_test_clas, y_pred_clas)
report = classification_report(y_test_clas, y_pred_clas)

accuracy, report
```

[ ]:
```
(1.0,
 '              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00        21\n           1       1.00      1.00      1.00
18\n\n    accuracy                           1.00        39\n   macro avg
1.00      1.00      1.00        39\nweighted avg       1.00      1.00      1.00
39\n')
```

100% **RARO**

K-Fold Cross-Validation para obtener una estimación más robusta del rendimiento del modelo:

[ ]:
```
from sklearn.model_selection import cross_val_score
from sklearn.impute import SimpleImputer

# Imputar los valores faltantes con la media en todo el dataset
imputer = SimpleImputer(strategy='mean')
X_clasificacion = imputer.fit_transform(X_clasificacion)

scores = cross_val_score(clf, X_clasificacion, y_clasificacion, cv=5)
print("Accuracy scores for each fold:", scores)
print("Mean accuracy:", scores.mean())
```

```
Accuracy scores for each fold: [1.         1.         1.         1.
0.97435897]
Mean accuracy: 0.9948717948717949
```

Características están contribuyendo más al modelo:

[ ]:
```
feature_importances = clf.feature_importances_
# Extraer los nombres de las columnas
features = df.columns
for feature, importance in zip(features, feature_importances):
    print(f"{feature}: {importance}")
```

```
País: 0.8230815405708695
Densidad (P/Km2): 0.10078059819060946
Abreviación: 0.060158667299754914
Terreno Agrícola (%): 0.01597919393876617
```

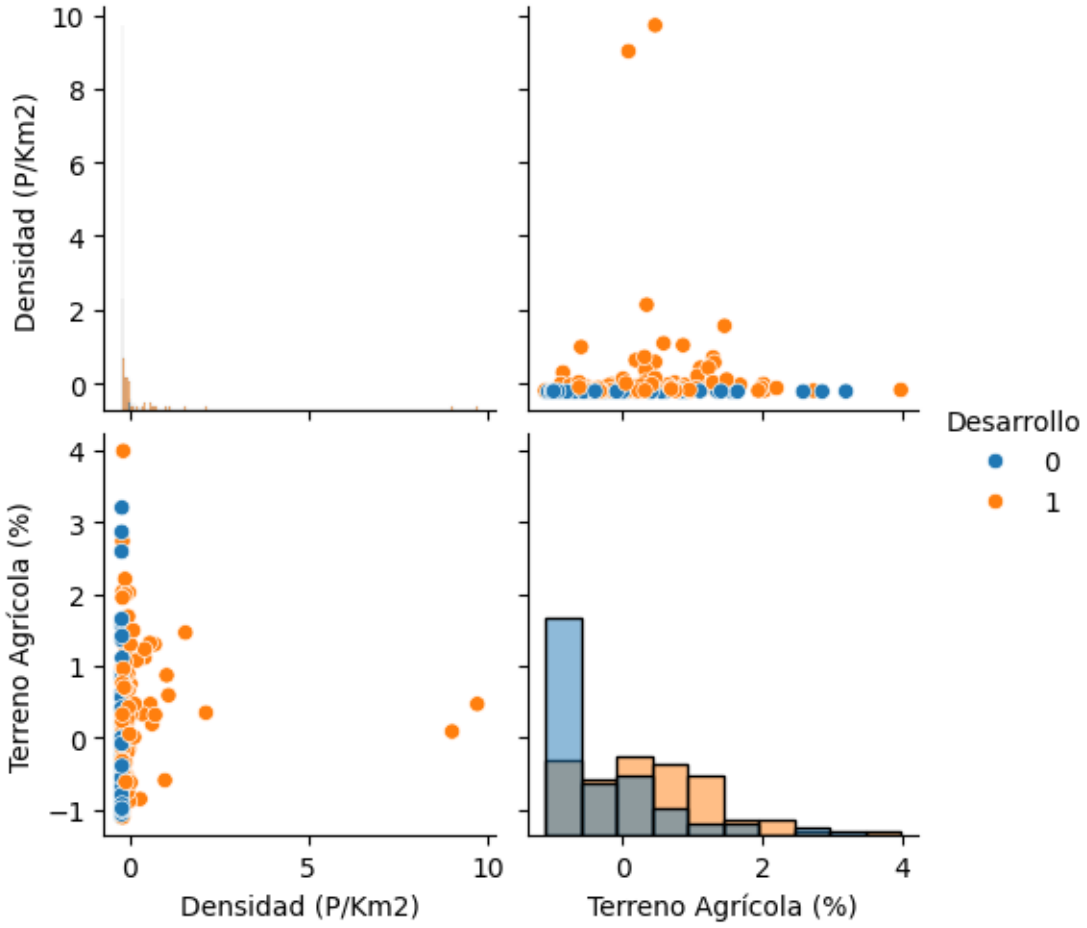
[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report
```

```
from sklearn.impute import SimpleImputer

# Definir las etiquetas: Consideramos que un país es desarrollado si su PIB es␣
 ↪mayor que la mediana.
df['Desarrollo'] = df['Producto Interno Bruto (PIB)'].apply(lambda x: 1 if x >␣
 ↪df['Producto Interno Bruto (PIB)'].median() else 0)

# Variables relevantes para el análisis de clasificación (eliminamos 'País' y␣
 ↪'Abreviación')
X_clasificacion = df[['Producto Interno Bruto (PIB)', 'Esperanza de Vida',␣
 ↪'Médicos por Mil Habitantes', 'Inscripción Bruta en Educación Primaria (%)']]
y_clasificacion = df['Desarrollo']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train_clas, X_test_clas, y_train_clas, y_test_clas =␣
 ↪train_test_split(X_clasificacion, y_clasificacion, test_size=0.2,␣
 ↪random_state=42)

# Imputar los valores faltantes con la media
imputer = SimpleImputer(strategy='mean')
X_train_clas = imputer.fit_transform(X_train_clas)
X_test_clas = imputer.transform(X_test_clas)

# Aplicar el clasificador Random Forest
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_clas, y_train_clas)

# Predecir los valores en el conjunto de prueba
y_pred_clas = clf.predict(X_test_clas)

# Evaluar el modelo
accuracy = accuracy_score(y_test_clas, y_pred_clas)
report = classification_report(y_test_clas, y_pred_clas)

# Imprimir resultados
print("Accuracy:", accuracy)
print("Classification Report:", report)

# Validación cruzada para evaluar el rendimiento general
X_clasificacion = imputer.fit_transform(X_clasificacion)
scores = cross_val_score(clf, X_clasificacion, y_clasificacion, cv=5)
print("Accuracy scores for each fold:", scores)
print("Mean accuracy:", scores.mean())
```

```
Accuracy: 1.0
Classification Report:               precision    recall  f1-score   support
```

```
           0       1.00      1.00      1.00        21
           1       1.00      1.00      1.00        18

    accuracy                           1.00        39
   macro avg       1.00      1.00      1.00        39
weighted avg       1.00      1.00      1.00        39

Accuracy scores for each fold: [1.         1.         1.         1.
0.97435897]
Mean accuracy: 0.9948717948717949
```

[ ]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

# Asegurarse de que X_clasificacion sea un DataFrame
X_clasificacion = pd.DataFrame(X_clasificacion)

# Obtener los nombres de las columnas del DataFrame original, ajustando a las␣
 ↪que realmente se desean usar
# Asumiendo que quieres usar "Densidad (P/Km2)" y "Terreno Agrícola (%)",␣
 ↪puedes agregar más si es necesario
column_names = ['Densidad (P/Km2)', 'Abreviación', 'Terreno Agrícola (%)',␣
 ↪'Otra Columna']

# Asignar los nombres de las columnas de acuerdo al número de columnas en␣
 ↪X_clasificacion
X_clasificacion.columns = column_names[:X_clasificacion.shape[1]]

# Escalar los datos para mejorar la visualización
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clasificacion)
X_scaled = pd.DataFrame(X_scaled, columns=column_names[:X_clasificacion.
 ↪shape[1]])

# Añadir la columna de desarrollo para hacer el gráfico por color
X_scaled['Desarrollo'] = df['Desarrollo'].values

# Seleccionar solo columnas numéricas relevantes para el análisis
numerical_columns = ['Densidad (P/Km2)', 'Terreno Agrícola (%)']  # Asegúrate␣
 ↪de usar solo variables numéricas

# Crear el pairplot con variables escaladas
sns.pairplot(X_scaled, vars=numerical_columns, hue='Desarrollo',␣
 ↪diag_kind='hist')
plt.show()
```



[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer # import the imputer

# Definir las etiquetas: Consideramos que un país es desarrollado si su PIB es␣
 ↪mayor que la mediana.
df['Desarrollo'] = df['Producto Interno Bruto (PIB)'].apply(lambda x: 1 if x >␣
 ↪df['Producto Interno Bruto (PIB)'].median() else 0)

# Variables para el análisis de clasificación
X_clasificacion = df[['Esperanza de Vida', 'Médicos por Mil Habitantes',␣
 ↪'Inscripción Bruta en Educación Primaria (%)']]
y_clasificacion = df['Desarrollo']

# Dividir los datos en conjuntos de entrenamiento y prueba
```

```
X_train_clas, X_test_clas, y_train_clas, y_test_clas =␣
 ↪train_test_split(X_clasificacion, y_clasificacion, test_size=0.2,␣
 ↪random_state=42)

# Imputar los valores faltantes con la media
imputer = SimpleImputer(strategy='mean')
X_train_clas = imputer.fit_transform(X_train_clas)
X_test_clas = imputer.transform(X_test_clas)

# Aplicar el clasificador Random Forest
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_clas, y_train_clas)

# Predecir los valores en el conjunto de prueba
y_pred_clas = clf.predict(X_test_clas)

# Evaluar el modelo
accuracy = accuracy_score(y_test_clas, y_pred_clas)
report = classification_report(y_test_clas, y_pred_clas)

accuracy, report
```

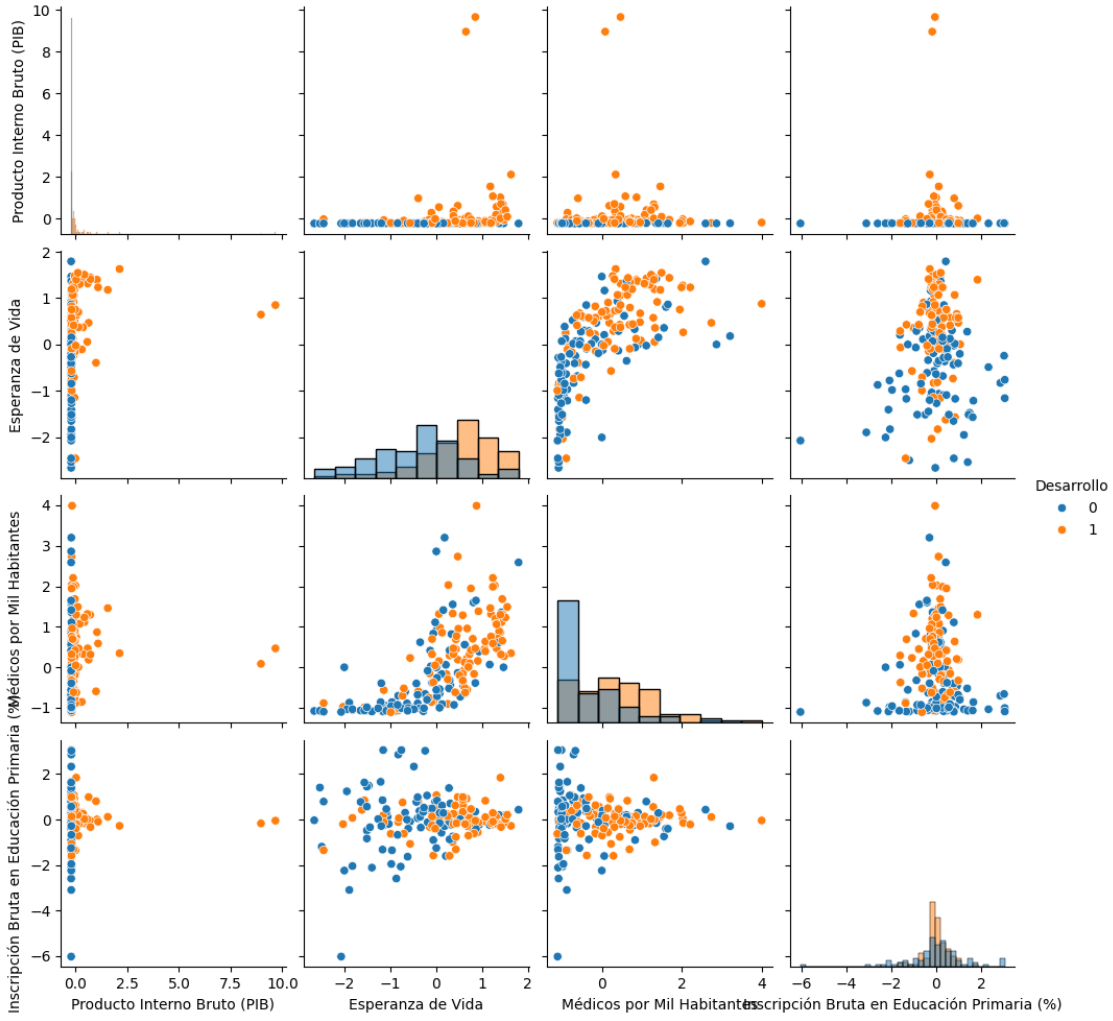
[ ]:
```
(0.7692307692307693,
 '              precision    recall  f1-score   support\n\n           0
0.77      0.81      0.79        21\n           1       0.76      0.72      0.74
18\n\n    accuracy                           0.77        39\n   macro avg
0.77      0.77      0.77        39\nweighted avg       0.77      0.77      0.77
39\n')
```

Elimine el producto Interno Bruto (PIB) porque la etiqueta de desarrollo se basa en la mediana del PIB, y esta característica también se utiliza como variable predictora, el modelo puede estar utilizando directamente la misma información.

Pero no me gusta eliminar una variable mas importante. Con lo cual vamos a agregar mas analisis.

[ ]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

# Variables relacionadas con el desarrollo
selected_columns = ['Producto Interno Bruto (PIB)', 'Esperanza de Vida',␣
 ↪'Médicos por Mil Habitantes', 'Inscripción Bruta en Educación Primaria (%)']

# Crear un DataFrame con las columnas relevantes y escalar las variables
X_clasificacion = df[selected_columns]

# Escalar las variables
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clasificacion)
X_scaled = pd.DataFrame(X_scaled, columns=selected_columns)

# Añadir la columna de desarrollo
X_scaled['Desarrollo'] = df['Desarrollo'].values

# Crear el pairplot con las nuevas variables
sns.pairplot(X_scaled, vars=selected_columns, hue='Desarrollo',␣
 ↪diag_kind='hist')
plt.show()
```



PIB vs. Esperanza de Vida: - Hay un patrón claro donde los países con un PIB más alto (puntos a la derecha) tienden a tener mayores esperanzas de vida. La mayoría de los países desarrollados se encuentran en la parte superior derecha, lo que sugiere una correlación positiva entre PIB y Esperanza de Vida. - Los países no desarrollados se concentran en la parte inferior izquierda (bajo

PIB y baja esperanza de vida).

PIB vs. Médicos por Mil Habitantes - Existe una correlación positiva entre PIB y la cantidad de médicos por mil habitantes. Los países con más PIB suelen tener un mejor acceso a los servicios de salud, lo cual es evidente en la mayor cantidad de médicos por cada mil habitantes en los países desarrollados.

PIB vs. Inscripción Bruta en Educación Primaria (%) - No hay una relación tan clara como con otras variables. Los países con alto PIB y alta inscripción son pocos, pero los países no desarrollados están muy dispersos, lo que indica que la inscripción en educación primaria no necesariamente sigue un patrón estricto con el PIB.

Esperanza de Vida vs. Médicos por Mil Habitantes - Existe una relación bastante clara donde los países con más médicos por mil habitantes tienen una mayor esperanza de vida. La mayoría de los países desarrollados se agrupan en la parte superior derecha, lo que refuerza la idea de que los sistemas de salud juegan un rol clave en el desarrollo.

Esperanza de Vida vs. Inscripción Bruta en Educación Primaria (%) - La inscripción en educación primaria muestra una dispersión más grande, con países desarrollados y no desarrollados esparcidos. Sin embargo, los países desarrollados parecen tener una tendencia a tener más esperanza de vida, pero sin una correlación directa fuerte con la inscripción primaria.

Médicos por Mil Habitantes vs. Inscripción en Educación Primaria

- Médicos por Mil Habitantes está claramente asociado con los países desarrollados, que tienden a tener una mayor cantidad de médicos. Sin embargo, la inscripción en educación primaria no parece ser un indicador tan fuerte para diferenciar entre países desarrollados y no desarrollados.

41 Conclusiones

PIB y Desarrollo: El Producto Interno Bruto (PIB) sigue siendo una de las variables más diferenciadoras entre países desarrollados y no desarrollados. Los países desarrollados tienden a tener un PIB mucho más alto, lo que está fuertemente relacionado con una mayor esperanza de vida y un mejor acceso a médicos por mil habitantes.

Esperanza de Vida y Desarrollo: La esperanza de vida también es un buen indicador de desarrollo. Los países desarrollados tienden a tener una mayor esperanza de vida y un mejor acceso a servicios de salud (más médicos por mil habitantes).

Médicos por Mil Habitantes: El acceso a servicios médicos parece ser un factor clave en el desarrollo de un país, ya que hay una relación clara entre la cantidad de médicos y los países desarrollados.

Inscripción Bruta en Educación Primaria: Aunque la inscripción en educación primaria es importante, no parece ser un factor determinante para diferenciar países desarrollados y no desarrollados en este análisis. Probablemente la educación de niveles superiores podría mostrar patrones más claros.

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
```

```
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Lista de indicadores
indicadores = ['Producto Interno Bruto (PIB)', 'Esperanza de Vida',
               'Inscripción Bruta en Educación Primaria (%)',
               'Médicos por Mil Habitantes', 'Mortalidad Infantil']

# Convertir a numérico y manejar valores faltantes
for col in indicadores:
    df[col] = pd.to_numeric(df[col], errors='coerce')

df = df.dropna(subset=indicadores)

# Asignar puntos
def asignar_puntos(valor, cortes, invertido=False):
    puntos = 0
    if invertido:
        for corte in cortes:
            if valor <= corte:
                puntos += 1
    else:
        for corte in cortes:
            if valor >= corte:
                puntos += 1
    return puntos

puntos_corte = {}
for col in indicadores:
    if col == 'Tasa de Mortalidad Infantil':
        puntos_corte[col] = df[col].quantile([0.25, 0.5, 0.75]).tolist()
    else:
        puntos_corte[col] = df[col].quantile([0.75, 0.5, 0.25]).tolist()

for col in indicadores:
    if col == 'Tasa de Mortalidad Infantil':
        df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,␣
 ↪puntos_corte[col], invertido=True))
    else:
        df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,␣
 ↪puntos_corte[col]))

# Calcular puntuación total y asignar etiquetas
puntos_cols = [f'Puntos_{col}' for col in indicadores]
df['Puntuacion_Total'] = df[puntos_cols].sum(axis=1)
umbral_desarrollo = df['Puntuacion_Total'].quantile(0.75)
```

```
df['Desarrollo'] = df['Puntuacion_Total'].apply(lambda x: 1 if x >=␣
 ↪umbral_desarrollo else 0)

# Preparar datos para el modelo
X_clasificacion = df[indicadores]
y_clasificacion = df['Desarrollo']

# Dividir los datos
X_train, X_test, y_train, y_test = train_test_split(X_clasificacion,␣
 ↪y_clasificacion, test_size=0.2, random_state=42)

# Imputar valores faltantes
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Entrenar el modelo
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Evaluar el modelo
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Precisión del modelo: {accuracy}")
print(report)

# Validación cruzada
scores = cross_val_score(clf, X_clasificacion, y_clasificacion, cv=5)
print("Puntuaciones de validación cruzada:", scores)
print("Precisión media:", scores.mean())

# Importancia de las características
importances = clf.feature_importances_
for feature, importance in zip(indicadores, importances):
    print(f"{feature}: {importance}")
```

```
<ipython-input-99-ce6c09d06cdf>:43: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,
puntos_corte[col]))
<ipython-input-99-ce6c09d06cdf>:43: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,
puntos_corte[col]))
<ipython-input-99-ce6c09d06cdf>:43: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,
puntos_corte[col]))
<ipython-input-99-ce6c09d06cdf>:43: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,
puntos_corte[col]))
<ipython-input-99-ce6c09d06cdf>:43: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[f'Puntos_{col}'] = df[col].apply(lambda x: asignar_puntos(x,
puntos_corte[col]))
<ipython-input-99-ce6c09d06cdf>:47: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Puntuacion_Total'] = df[puntos_cols].sum(axis=1)
<ipython-input-99-ce6c09d06cdf>:49: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Desarrollo'] = df['Puntuacion_Total'].apply(lambda x: 1 if x >=
umbral_desarrollo else 0)

Precisión del modelo: 0.9473684210526315
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        24
           1       1.00      0.86      0.92        14

    accuracy                           0.95        38
   macro avg       0.96      0.93      0.94        38
weighted avg       0.95      0.95      0.95        38

Puntuaciones de validación cruzada: [0.92105263 0.94736842 0.89473684 0.89189189
0.86486486]
Precisión media: 0.9039829302987197
Producto Interno Bruto (PIB): 0.34164598799818563
Esperanza de Vida: 0.21869932219212737
Inscripción Bruta en Educación Primaria (%): 0.17887438229329466
Médicos por Mil Habitantes: 0.1559318712380329
Mortalidad Infantil: 0.10484843627835937
```

42 Analisis

42.0.1 1- Precisión del Modelo:

Precisión del modelo: 0.9473 (94.73%) El modelo muestra un excelente rendimiento, con una precisión bastante alta en el conjunto de prueba. Esto significa que el modelo ha sido capaz de clasificar correctamente si un país es desarrollado o no con una precisión del 94.73%.

42.0.2 2- Reporte de clasificación:

Clase 0 (países no desarrollados):

- Precisión: 0.92
- Recall: 1.00
- F1-score: 0.96

Esto significa que el modelo ha identificado correctamente todos los países no desarrollados sin cometer errores (recall perfecto de 1.00), pero tiene un pequeño número de falsos positivos (precisión de 0.92).

Clase 1 (países desarrollados):

- Precisión: 1.00
- Recall: 0.86
- F1-score: 0.92

El modelo ha identificado correctamente todos los países desarrollados con una precisión del 100%, pero su recall es de 0.86, lo que indica que se ha perdido una pequeña cantidad de países desarrollados que no han sido clasificados correctamente.

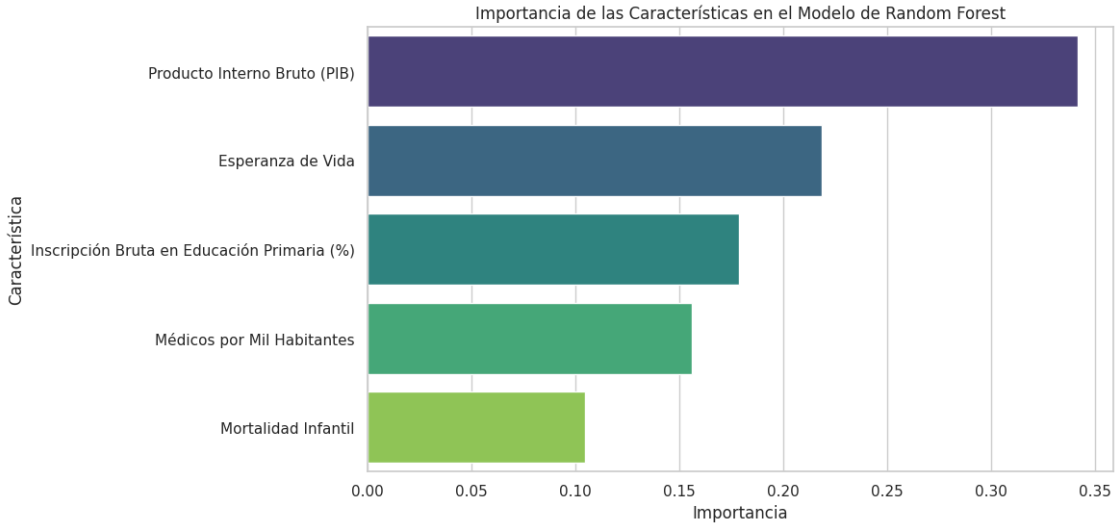
42.0.3 3- Validación cruzada

- Puntuaciones de validación cruzada: [0.921, 0.947, 0.894, 0.891, 0.864]

- Precisión media: 0.9039 (90.39%)
- Aunque la precisión en la validación cruzada es ligeramente menor que en el conjunto de prueba, sigue siendo muy alta (alrededor del 90%), lo que sugiere que el modelo es consistente y generaliza bien en diferentes subconjuntos de datos.

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Crear un DataFrame con las importancias
feature_importances = pd.DataFrame({
    'Característica': indicadores,
    'Importancia': importances
})

# Ordenar las características por importancia
feature_importances = feature_importances.sort_values(by='Importancia',␣
 ↪ascending=False)

# Configurar el estilo de Seaborn
sns.set(style="whitegrid")

# Crear el gráfico de barras
plt.figure(figsize=(10, 6))
sns.barplot(x='Importancia', y='Característica', data=feature_importances,␣
 ↪palette='viridis')

# Añadir títulos y etiquetas
plt.title('Importancia de las Características en el Modelo de Random Forest')
plt.xlabel('Importancia')
plt.ylabel('Característica')

# Mostrar el gráfico
plt.show()
```

```
<ipython-input-100-80fb4c51d8ee>:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Importancia', y='Característica', data=feature_importances,
palette='viridis')
```



Interpretación del gráfico - Eje Y (Característica): Lista de las variables utilizadas en el modelo. - Eje X (Importancia): Valor numérico que representa la influencia de cada característica en las predicciones del modelo.

Observaciones: Las barras más largas indican características más importantes. En tu caso, el PIB y la Esperanza de Vida son las características más influyentes.

43 1. Contextualización de los Gráficos

Gráfico:

Importancia de las Características en el Modelo de Random Forest Descripción: Este gráfico muestra la importancia de cada característica en la predicción del desarrollo económico de los países. Se observa que el Producto Interno Bruto (PIB) es la característica más influyente, seguido por la Esperanza de Vida. La Mortalidad Infantil, aunque es relevante, tiene un impacto menor en comparación con las demás características.

Importancia: Este gráfico es esencial para comprender qué variables juegan un papel más significativo en la predicción de si un país se clasifica como desarrollado o no. Estas características reflejan los factores más importantes que suelen estar asociados con el desarrollo económico.

44 2. Resumen Ejecutivo

Este análisis utiliza un modelo de clasificación con Random Forest para predecir si un país se clasifica como desarrollado o no, en función de varios indicadores económicos y de salud.

Precisión del Modelo: El modelo alcanzó una precisión del 94.7%, lo que indica que clasifica correctamente la mayoría de los países en desarrollados o no desarrollados.

Precisión por categoría: Para los países desarrollados (etiqueta 1), el modelo tuvo una precisión del 100%, lo que significa que identifica con exactitud a todos los países desarrollados en el conjunto

de prueba. Para los países no desarrollados (etiqueta 0), la precisión fue del 92%, lo que refleja que el modelo puede confundir algunos países no desarrollados con desarrollados, pero en general mantiene un alto nivel de precisión.

45 3. Interpretación de las Métricas

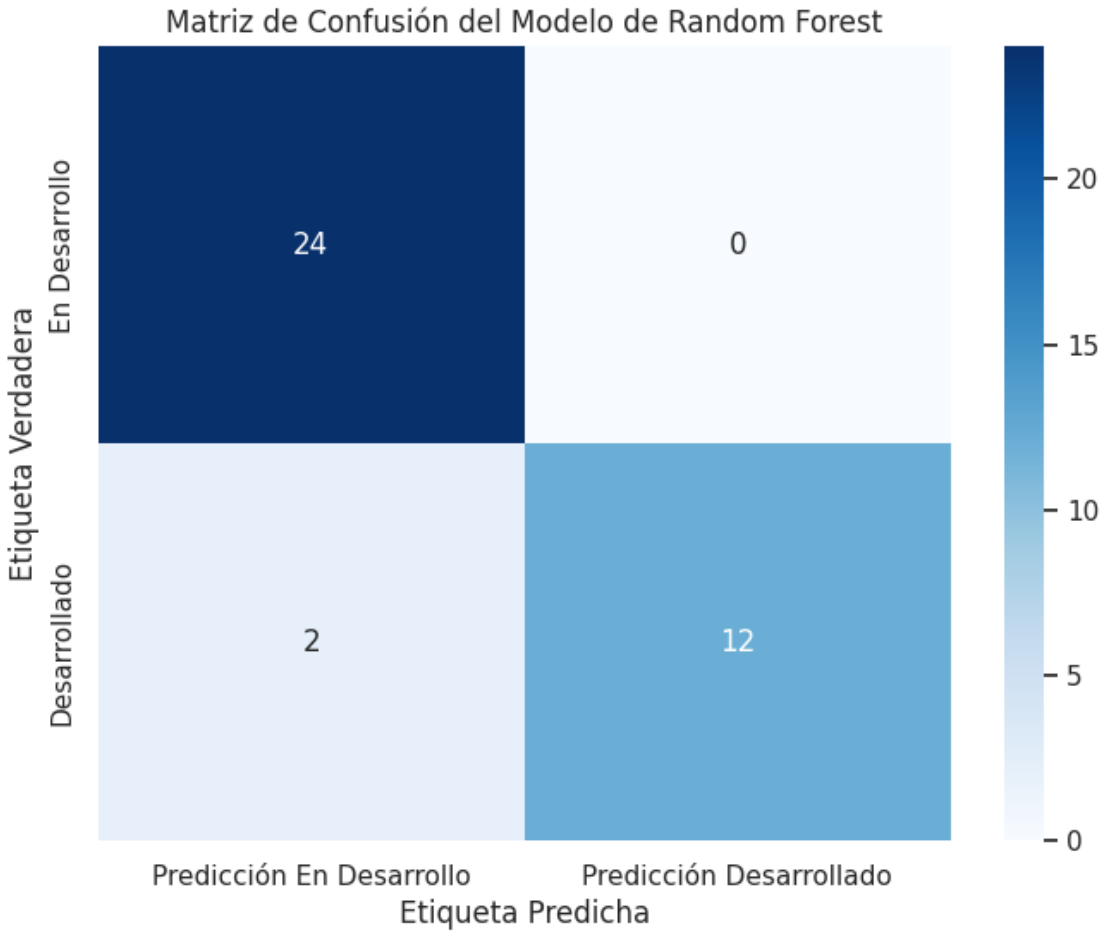
Precisión del Modelo: El modelo tiene una alta precisión (94.7%), lo que lo hace adecuado para clasificar países en función de sus indicadores económicos y de salud. Esto sugiere que las características seleccionadas son relevantes para este tipo de clasificación.

Reporte de Clasificación: El modelo es capaz de identificar tanto a los países desarrollados como no desarrollados con una alta precisión. Sin embargo, su desempeño es mejor en los países desarrollados, lo que podría deberse a la diferenciación más clara en sus características económicas.

46 4. Discusión de la Importancia de las Características

- Producto Interno Bruto (PIB): Como era de esperar, el PIB es el factor más influyente en la clasificación, representando el 34% de la importancia total. Esto coincide con estudios económicos que sugieren que el PIB es un indicador clave del desarrollo.
- Esperanza de Vida: También es un factor crucial, con una importancia de 21.8%, lo que refleja la importancia de la salud y la longevidad en la evaluación del desarrollo económico.
- Mortalidad Infantil: Aunque tiene una menor importancia relativa (10.4%), sigue siendo relevante, ya que está inversamente relacionada con el desarrollo. Los países con tasas más bajas de mortalidad infantil tienden a ser clasificados como desarrollados.

[ ]:
```
from sklearn.metrics import confusion_matrix

# Calcular la matriz de confusión
cm = confusion_matrix(y_test, y_pred)

# Crear un DataFrame para facilitar la visualización
cm_df = pd.DataFrame(cm, index=['En Desarrollo', 'Desarrollado'],␣
 ↪columns=['Predicción En Desarrollo', 'Predicción Desarrollado'])

# Graficar la matriz de confusión con Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues')

# Añadir títulos y etiquetas
plt.title('Matriz de Confusión del Modelo de Random Forest')
plt.ylabel('Etiqueta Verdadera')
plt.xlabel('Etiqueta Predicha')

# Mostrar el gráfico
plt.show()
```



Interpretación del gráfico

1- Diagonal Principal: Muestra las predicciones correctas. - Valor [0,0]: Países en desarrollo correctamente clasificados. - Valor [1,1]: Países desarrollados correctamente clasificados.

2- Fuera de la Diagonal: Muestra las predicciones incorrectas. - Valor [0,1]: Países en desarrollo clasificados incorrectamente como desarrollados. - Valor [1,0]: Países desarrollados clasificados incorrectamente como en desarrollo.

Observaciones:

Los números altos en la diagonal indican buen rendimiento. En tu caso, el modelo clasifica correctamente la mayoría de los casos.

47 1- Contextualización del Gráfico

Gráfico: Matriz de Confusión del Modelo de Random Forest

Descripción: Este gráfico muestra la matriz de confusión del modelo Random Forest, que ilustra el desempeño del modelo al clasificar países como “Desarrollados” o “En Desarrollo”. La matriz

presenta las predicciones correctas en la diagonal (de color oscuro), mientras que las predicciones incorrectas aparecen en las otras casillas.

- **Clase "En Desarrollo" (etiqueta 0):** De los 24 países clasificados correctamente como en desarrollo, no hubo predicciones erróneas en esta clase.
- **Clase "Desarrollado" (etiqueta 1):** De los 14 países desarrollados, el modelo clasificó incorrectamente a 2 como en desarrollo.

48 2- Interpretación de la Matriz de Confusión

Exactitud en la clase 'En Desarrollo':

El modelo no cometió errores al clasificar los países en desarrollo. Este excelente desempeño puede estar vinculado a que las características de los países en desarrollo son más diferenciadas en relación con los indicadores seleccionados (PIB, esperanza de vida, etc.).

- **Errores en la clase 'Desarrollado':** Hubo 2 casos en los que el modelo clasificó incorrectamente países desarrollados como en desarrollo. Estos errores podrían deberse a características atípicas o más cercanas a las de los países en desarrollo.
- **Balance general:** La matriz de confusión refuerza que el modelo tiene un alto desempeño en ambas clases, aunque es más preciso con la clase de países en desarrollo.

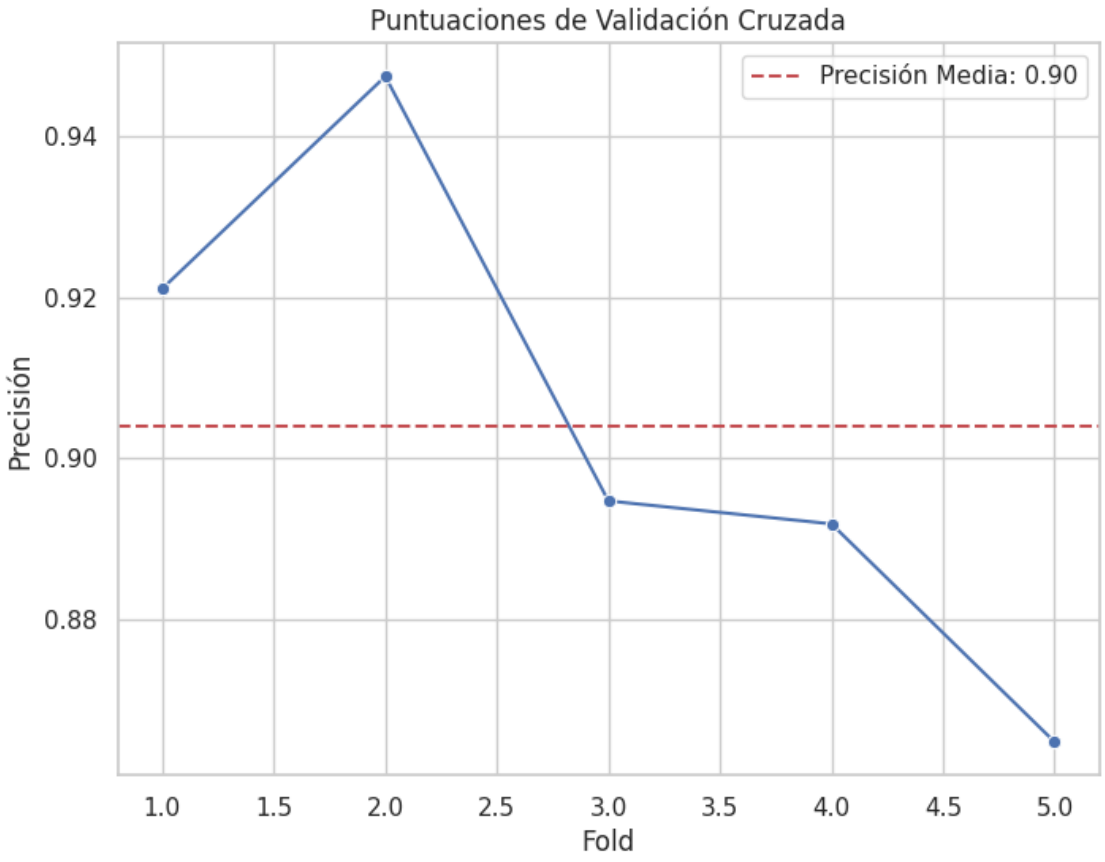
49 3- Resumen Ejecutivo Ampliado con Matriz de Confusión

El modelo Random Forest alcanzó una precisión de 94.7% al predecir si un país es desarrollado o en desarrollo. El análisis detallado de la matriz de confusión muestra que:

El modelo clasificó correctamente todos los países en desarrollo. Cometió solo 2 errores al clasificar países desarrollados como en desarrollo. Estos resultados, junto con el análisis de importancia de características, sugieren que el modelo es particularmente efectivo para identificar países en desarrollo y que las características más influyentes, como el PIB y la Esperanza de Vida, juegan un papel clave en estas predicciones.

[ ]:
```
# Crear un DataFrame con las puntuaciones
cv_scores_df = pd.DataFrame({
    'Fold': range(1, len(scores) + 1),
    'Precisión': scores
})

# Crear el gráfico de líneas
plt.figure(figsize=(8, 6))
sns.lineplot(x='Fold', y='Precisión', data=cv_scores_df, marker='o')

# Añadir títulos y etiquetas
plt.title('Puntuaciones de Validación Cruzada')
plt.xlabel('Fold')
plt.ylabel('Precisión')
```

```
# Añadir una línea horizontal con la precisión media
plt.axhline(y=scores.mean(), color='r', linestyle='--', label=f'Precisión Media:
  ↪ {scores.mean():.2f}')
plt.legend()

# Mostrar el gráfico
plt.show()
```



Interpretación del gráfico

- Eje X (Fold): Número de cada partición en la validación cruzada.
- Eje Y (Precisión): Precisión obtenida en cada fold.

Observaciones: - Puedes observar cómo varía la precisión entre diferentes folds. - La línea horizontal muestra la precisión media.

50 1- Contextualización del Gráfico

Gráfico: Puntuaciones de Validación Cruzada

Descripción:

Este gráfico muestra las puntuaciones de precisión en cinco "folds" o divisiones de los datos realizadas durante la validación cruzada. Cada punto en la línea azul representa la precisión del modelo en cada fold, mientras que la línea roja discontinua indica la precisión media, que es aproximadamente 0.90.

Importancia: El gráfico permite observar la variabilidad en el rendimiento del modelo en diferentes subconjuntos de los datos, lo que proporciona una medida de su estabilidad. En este caso, la precisión es consistente, aunque ligeramente decreciente en los últimos "folds", lo que sugiere que el modelo es robusto pero podría estar sobreajustado en algunos subconjuntos.

51 2- Interpretación del Gráfico

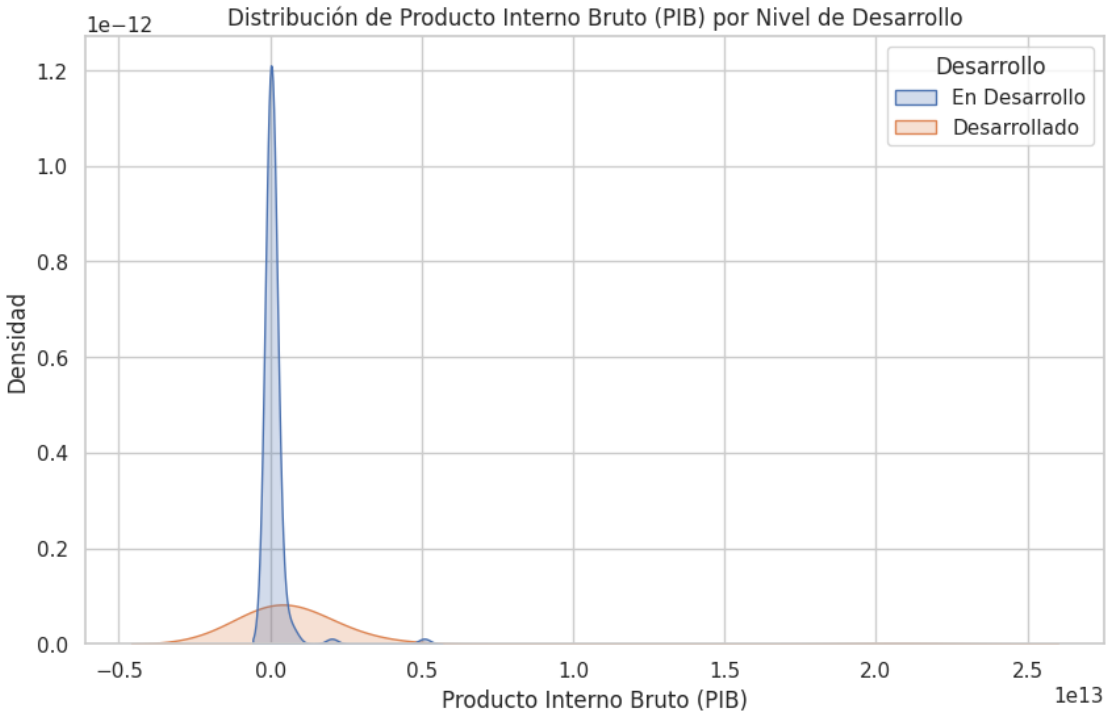
- Precisión del Modelo: Las puntuaciones oscilan entre 0.86 y 0.95, con una precisión media de 0.90. Esto indica que el modelo mantiene un rendimiento alto y estable en general, aunque existen ligeras variaciones entre los diferentes subconjuntos de datos.
- Variabilidad de los Folds: Se observa una caída en la precisión a partir del tercer fold, que podría estar indicando que ciertos subconjuntos de datos son más difíciles de predecir. Esto podría ser útil para futuras mejoras en el modelo, como ajustar hiperparámetros o probar otros algoritmos.
- Precisión Media: La línea roja de precisión media refuerza que el modelo es consistente en su desempeño general, lo que confirma su efectividad para predecir correctamente el desarrollo de los países con base en los indicadores seleccionados.

52 3- Resumen Ejecutivo con Validación Cruzada

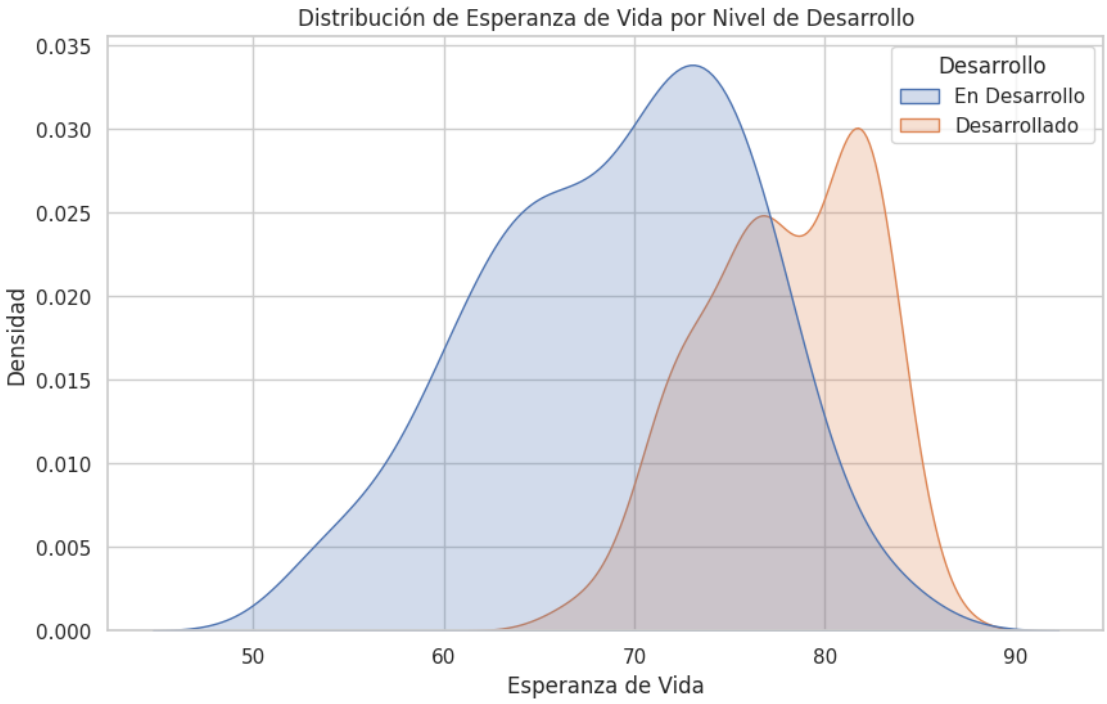
El proceso de validación cruzada refuerza la robustez del modelo de Random Forest para clasificar países como desarrollados o en desarrollo. El gráfico anterior muestra que el modelo tiene una precisión media de 0.90, y aunque la precisión varía entre 0.86 y 0.95 en los diferentes folds, el rendimiento general es sólido. Esto asegura que el modelo puede generalizar bien en diferentes subconjuntos de datos.

[ ]:
```
# Combinar las características y la etiqueta en un solo DataFrame
data = df[indicadores + ['Desarrollo']]

# Convertir la etiqueta numérica a categórica para mejor legibilidad
data['Desarrollo'] = data['Desarrollo'].map({0: 'En Desarrollo', 1:␣
 ↪'Desarrollado'})

# Graficar distribuciones
for col in indicadores:
    plt.figure(figsize=(10, 6))
    sns.kdeplot(data=data, x=col, hue='Desarrollo', shade=True)
    plt.title(f'Distribución de {col} por Nivel de Desarrollo')
    plt.xlabel(col)
    plt.ylabel('Densidad')
    plt.show()
```

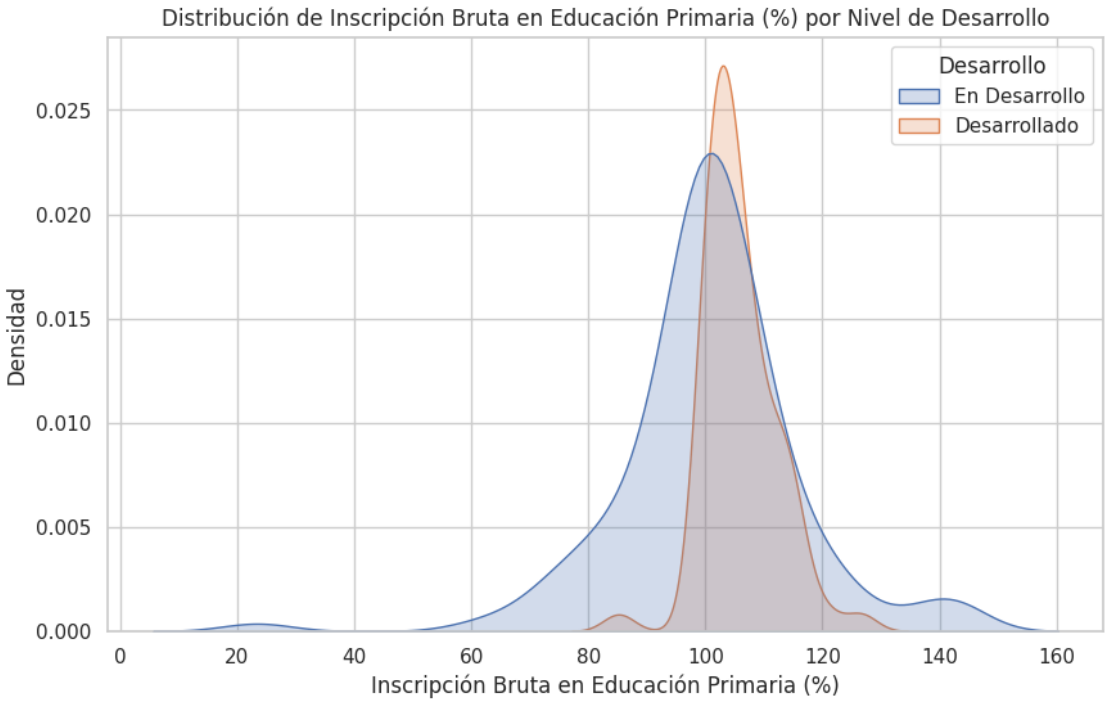
```
<ipython-input-103-d74350c19671>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Desarrollo'] = data['Desarrollo'].map({0: 'En Desarrollo', 1:
'Desarrollado'})
<ipython-input-103-d74350c19671>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data, x=col, hue='Desarrollo', shade=True)
```



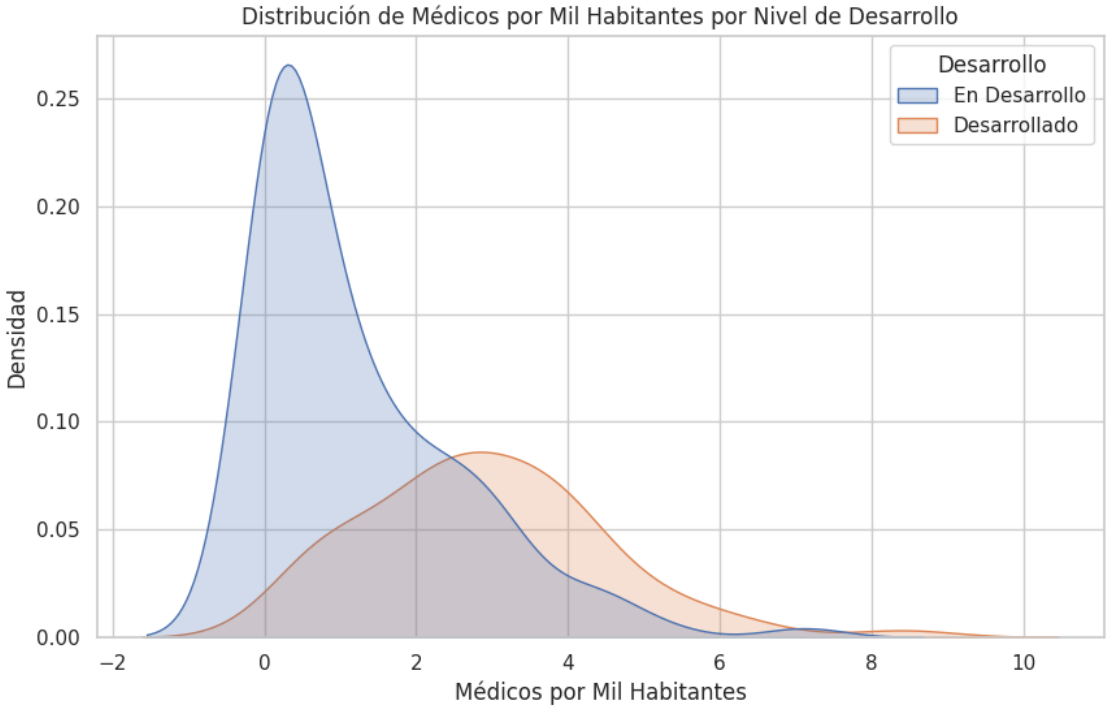

```
<ipython-input-103-d74350c19671>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data, x=col, hue='Desarrollo', shade=True)
```



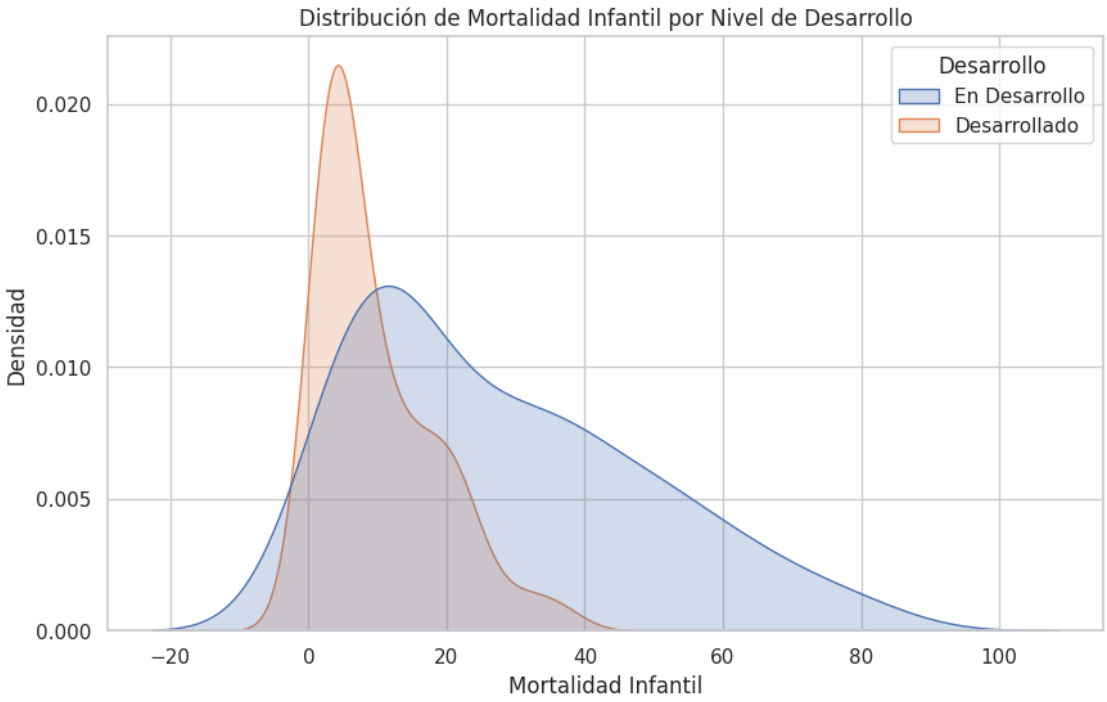

```
<ipython-input-103-d74350c19671>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data, x=col, hue='Desarrollo', shade=True)
```




```
<ipython-input-103-d74350c19671>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data, x=col, hue='Desarrollo', shade=True)
```




```
<ipython-input-103-d74350c19671>:10: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data, x=col, hue='Desarrollo', shade=True)
```



Interpretación de los gráficos - Eje X: Valores de la característica. - Eje Y: Densidad (frecuencia relativa). Colores: Cada color representa un nivel de desarrollo.

Observaciones:

- Puedes ver cómo se diferencian las distribuciones entre los dos grupos.
- Características con distribuciones bien separadas son más discriminantes para el modelo.

53 1. Distribucion del producto interno brito (PIB) por nivel de desarrollo.

1.1- Contextualización del Gráfico

- Gráfico: Distribución de Producto Interno Bruto (PIB) por Nivel de Desarrollo Descripción: Este gráfico de densidad muestra la distribución del Producto Interno Bruto (PIB) en países desarrollados y en desarrollo. Las áreas sombreadas reflejan la densidad de los datos, donde la curva azul representa a los países en desarrollo y la curva naranja a los países desarrollados.
- Países en Desarrollo: La mayoría de los países en desarrollo se concentran alrededor de valores bajos de PIB, lo que es consistente con las características económicas de estas naciones.
- Países Desarrollados: Los países desarrollados presentan una distribución más extendida en valores de PIB más altos, lo que refleja economías más avanzadas.

1.2. Interpretación del Gráfico

- Concentración de PIB en Países en Desarrollo: La curva de los países en desarrollo es mucho

más estrecha, lo que sugiere una menor variabilidad en los PIB de estos países. La mayoría de ellos tienen un PIB relativamente bajo, lo que refleja economías en crecimiento o menos industrializadas.

- Diversificación en Países Desarrollados: Los países desarrollados muestran una mayor variabilidad en su PIB, con una cola más larga hacia valores altos. Esto indica que algunos países desarrollados tienen PIB excepcionalmente altos, mientras que otros tienen PIB más moderados.
- Comparación entre Desarrollados y En Desarrollo: La clara separación entre las dos curvas refuerza la relación entre el PIB y el desarrollo económico. Los países con PIB más altos tienden a clasificarse como desarrollados, mientras que los países con PIB más bajos se clasifican como en desarrollo.

1.3 Resumen Ejecutivo Ampliado con Distribución de PIB

El análisis de la distribución del Producto Interno Bruto (PIB) revela que los países en desarrollo están altamente concentrados en valores bajos de PIB, lo que refleja economías menos avanzadas. En cambio, los países desarrollados muestran una mayor dispersión, con algunos alcanzando PIB significativamente altos. Este gráfico refuerza la importancia del PIB como un indicador clave en la clasificación del desarrollo económico, ya que está estrechamente vinculado a la capacidad económica y la infraestructura de una nación.

54 2. Distribucion de la esperanza de vida por nivel de desarrollo

2.1 Contextualización del Gráfico

- Gráfico: Distribución de Esperanza de Vida por Nivel de Desarrollo
- Descripción: Este gráfico de densidad muestra la distribución de la Esperanza de Vida en países desarrollados y en desarrollo. Las curvas representan la densidad de los países en función de su esperanza de vida. La curva azul refleja a los países en desarrollo, mientras que la curva naranja corresponde a los países desarrollados.
- Países en Desarrollo: La curva muestra que la mayoría de los países en desarrollo tienen una esperanza de vida alrededor de los 60 a 70 años.
- Países Desarrollados: Los países desarrollados, por otro lado, presentan una esperanza de vida más alta, principalmente entre 70 y 85 años.

2.2 Interpretación del Gráfico

- Esperanza de Vida en Países en Desarrollo: La curva azul muestra que la mayoría de los países en desarrollo tienen una esperanza de vida inferior a los 70 años, con un pico alrededor de los 65 años. Esto sugiere que la calidad y accesibilidad de los servicios de salud en estos países no son tan avanzadas, lo que se refleja en su menor esperanza de vida.
- Esperanza de Vida en Países Desarrollados: En los países desarrollados, la esperanza de vida es significativamente mayor. La curva naranja tiene su punto más alto alrededor de los 75 años, lo que sugiere mejores condiciones de vida, acceso a la atención sanitaria y avances en la medicina.

- Comparación entre Desarrollados y En Desarrollo: Existe una clara separación entre las dos distribuciones. Los países desarrollados tienden a tener una esperanza de vida más alta, lo que es un indicador importante del nivel de desarrollo económico y social. La superposición entre ambas curvas es mínima, lo que sugiere que la esperanza de vida es una característica altamente discriminativa entre los países desarrollados y en desarrollo.

2.3 Resumen Ejecutivo con Distribución de Esperanza de Vida

La esperanza de vida es un indicador clave que distingue a los países desarrollados de los países en desarrollo. En este gráfico, se observa que los países desarrollados tienen una esperanza de vida más alta, con una media en torno a los 75 años, mientras que los países en desarrollo tienden a concentrarse en el rango de 60 a 70 años. Este análisis refuerza la idea de que el desarrollo económico y la calidad de vida están estrechamente relacionados, y que la mejora en los servicios de salud es fundamental para alcanzar un mayor nivel de desarrollo.

55 3. Distribución de Inscripción Bruta en Educación Primaria (%) por Nivel de Desarrollo

3.1 Contextualización del Gráfico

- Gráfico: Distribución de Inscripción Bruta en Educación Primaria (%) por Nivel de Desarrollo
- Descripción: Este gráfico de densidad muestra la distribución de la Inscripción Bruta en Educación Primaria entre los países desarrollados y en desarrollo. La curva azul representa a los países en desarrollo, mientras que la curva naranja representa a los países desarrollados.
- Países en Desarrollo: La mayoría de los países en desarrollo presentan una inscripción en educación primaria entre el 80% y el 100%.
- Países Desarrollados: Los países desarrollados tienen una inscripción más concentrada alrededor del 100%, con menor dispersión hacia ambos lados.

3.2 Interpretación del Gráfico

- Inscripción en Educación Primaria en Países en Desarrollo: La curva azul muestra que los países en desarrollo tienen una distribución más amplia en términos de inscripción bruta en educación primaria. Aunque muchos países alcanzan tasas de inscripción cercanas al 100%, otros están por debajo, indicando desafíos en el acceso a la educación.
- Inscripción en Educación Primaria en Países Desarrollados: La curva naranja de los países desarrollados está más concentrada alrededor del 100%, lo que refleja que la mayoría de estos países han alcanzado una inscripción bruta alta en educación primaria, un indicador clave de sus sistemas educativos robustos.
- Comparación entre Desarrollados y En Desarrollo: Aunque ambos grupos presentan una inscripción alta en su mayoría, la dispersión es más pronunciada en los países en desarrollo. Esto indica que algunos países aún tienen desafíos significativos en la cobertura educativa, mientras que los países desarrollados han logrado una inscripción cercana al 100%.

3.3 Resumen Ejecutivo con Distribución de Inscripción en Educación Primaria

La inscripción en educación primaria es otro indicador importante de desarrollo. Este análisis revela que los países desarrollados tienen una inscripción en primaria muy cercana al 100%, mientras que

los países en desarrollo presentan una mayor variabilidad, con algunos alcanzando inscripciones más bajas. Esto sugiere que los países en desarrollo, aunque han mejorado sus tasas de inscripción, todavía enfrentan barreras significativas en el acceso a la educación para toda su población

56 4. Distribución de Médicos por Mil Habitantes por Nivel de Desarrollo

4.1 Contextualización del Gráfico

- Gráfico: Distribución de Médicos por Mil Habitantes por Nivel de Desarrollo
- Descripción: Este gráfico de densidad muestra la distribución del número de Médicos por Mil Habitantes en países desarrollados y en desarrollo. La curva azul representa a los países en desarrollo, mientras que la curva naranja representa a los países desarrollados.
- Países en Desarrollo: La mayoría de los países en desarrollo tienen menos de 2 médicos por cada mil habitantes, con un pico cercano a 1 médico por mil habitantes.
- Países Desarrollados: Los países desarrollados, en cambio, tienen una distribución más amplia, con un mayor número de médicos, alcanzando hasta 4 médicos por mil habitantes y algunos con más de 6 médicos.

4.2 Interpretación del Gráfico

- Disponibilidad de Médicos en Países en Desarrollo: La curva azul muestra que los países en desarrollo tienden a tener una baja proporción de médicos por mil habitantes. El pico alrededor de 1 médico por cada mil habitantes refleja que muchos de estos países enfrentan desafíos en el acceso a personal médico, lo que puede repercutir en la calidad del sistema de salud.
- Disponibilidad de Médicos en Países Desarrollados: En los países desarrollados, la curva naranja muestra una mayor proporción de médicos, con muchos países alcanzando entre 2 y 4 médicos por cada mil habitantes. Algunos países desarrollados tienen incluso más de 6 médicos por mil habitantes, lo que indica sistemas de salud más robustos y mejor financiados.
- Comparación entre Desarrollados y En Desarrollo: Hay una clara diferencia entre ambos grupos, con los países desarrollados mostrando una mayor disponibilidad de médicos por mil habitantes en comparación con los países en desarrollo. Esta diferencia refuerza la importancia del acceso a profesionales de la salud como un indicador clave del desarrollo.

4.3 Resumen Ejecutivo con Distribución de Médicos por Mil Habitantes

El acceso a médicos es un indicador crucial del desarrollo económico y social. Este gráfico muestra que los países desarrollados tienden a tener una proporción significativamente mayor de médicos por mil habitantes en comparación con los países en desarrollo. En promedio, los países en desarrollo cuentan con menos de 2 médicos por mil habitantes, mientras que los países desarrollados logran alcanzar entre 2 y 6 médicos por mil habitantes. Esta diferencia subraya la importancia de un acceso adecuado a personal médico para mejorar la calidad de vida y el bienestar general de la población.

57 5. Distribución de Mortalidad Infantil por Nivel de Desarrollo

5.1 Contextualización del Gráfico

- Gráfico: Distribución de Mortalidad Infantil por Nivel de Desarrollo
- Descripción: Este gráfico de densidad muestra la distribución de la Mortalidad Infantil (muertes por cada 1,000 nacidos vivos) en países desarrollados y en desarrollo. La curva azul representa a los países en desarrollo, mientras que la curva naranja representa a los países desarrollados.
- Países en Desarrollo: La curva muestra que la mortalidad infantil en los países en desarrollo es significativamente más alta, con valores que alcanzan hasta 60 muertes por cada 1,000 nacidos vivos y algunos incluso más altos.
- Países Desarrollados: La curva naranja refleja una mortalidad infantil mucho más baja en los países desarrollados, concentrada principalmente en valores cercanos a 0, lo que indica un sistema de salud más avanzado.

5.2 Interpretación del Gráfico

- Mortalidad Infantil en Países en Desarrollo: La curva azul de los países en desarrollo tiene una distribución más amplia y concentrada en valores más altos. La alta densidad en mortalidad infantil (valores entre 20 y 40 muertes por cada 1,000 nacidos vivos) indica que muchos de estos países enfrentan desafíos significativos en cuanto a atención médica prenatal, acceso a recursos médicos y nutrición adecuada.
- Mortalidad Infantil en Países Desarrollados: En los países desarrollados, la mortalidad infantil es mucho más baja. La curva naranja está muy concentrada cerca de 0, lo que indica que estos países han logrado tasas muy bajas de mortalidad infantil, lo que refleja sistemas de salud eficientes, una alta calidad de vida y acceso a servicios médicos.
- Comparación entre Desarrollados y En Desarrollo: La diferencia entre los países desarrollados y en desarrollo es clara. Los países desarrollados tienen tasas significativamente más bajas de mortalidad infantil en comparación con los países en desarrollo. Esta variable es un indicador crítico del bienestar de una nación y su desarrollo social.

5.3 Resumen Ejecutivo con Distribución de Mortalidad Infantil

La mortalidad infantil es uno de los indicadores más poderosos para diferenciar entre países desarrollados y en desarrollo. Este gráfico muestra una marcada diferencia entre los dos grupos: los países en desarrollo presentan tasas significativamente más altas de mortalidad infantil, lo que indica problemas en el acceso y la calidad de los servicios médicos. Por otro lado, los países desarrollados han logrado reducir la mortalidad infantil a niveles cercanos a 0, gracias a un mejor sistema de salud y condiciones de vida.

58 Conclusiones Finales del Análisis

Tras analizar los gráficos de densidad que muestran la distribución de varios indicadores clave (Producto Interno Bruto, Esperanza de Vida, Inscripción Bruta en Educación Primaria, Médicos por Mil Habitantes y Mortalidad Infantil) por nivel de desarrollo, se pueden extraer varias conclusiones importantes:

1. **Relación entre el Producto Interno Bruto (PIB) y el Nivel de Desarrollo** Conclusión: El PIB es un fuerte indicador del desarrollo económico. Los países desarrollados muestran consistentemente un PIB más alto, mientras que los países en desarrollo se concentran en rangos más bajos de PIB. Esta diferencia es significativa y consistente con la teoría económica de que un mayor PIB está vinculado a mejores infraestructuras, tecnologías avanzadas y mayor bienestar social.

- Implicación: Las políticas que promuevan el crecimiento económico y una distribución equitativa de los recursos pueden ayudar a los países en desarrollo a mejorar sus niveles de desarrollo económico.

2. **Esperanza de Vida como Indicador Social Clave** Conclusión: Los países desarrollados tienen una esperanza de vida significativamente mayor en comparación con los países en desarrollo. Este hecho está fuertemente relacionado con la calidad de los sistemas de salud, la accesibilidad a atención médica avanzada y las mejores condiciones de vida en los países desarrollados.

- Implicación: Aumentar la esperanza de vida en los países en desarrollo requiere una mejora en la calidad de la atención sanitaria, mejor acceso a recursos de salud, y avances en la educación en salud pública.

3. **Inscripción en Educación Primaria y Desarrollo Social** Conclusión: Tanto los países desarrollados como en desarrollo muestran una inscripción bruta en educación primaria alta, lo cual es alentador. Sin embargo, los países en desarrollo tienen una mayor variabilidad en la tasa de inscripción, lo que indica que aún existen áreas con desafíos en el acceso a la educación primaria.

- Implicación: Fomentar el acceso universal a la educación primaria es crucial para que los países en desarrollo mejoren su capital humano, lo que eventualmente impulsará su crecimiento económico y desarrollo social.

4. **Diferencias en la Disponibilidad de Médicos** Conclusión: El número de médicos por mil habitantes es mucho mayor en los países desarrollados, lo que refuerza la idea de que la calidad y disponibilidad de los recursos de atención médica están estrechamente vinculadas al nivel de desarrollo. Los países en desarrollo, en cambio, presentan una proporción significativamente menor de médicos.

- Implicación: Los países en desarrollo necesitan mejorar la infraestructura de sus sistemas de salud, aumentando la capacitación y disponibilidad de médicos para brindar un acceso más equitativo a la atención médica.

5. **Mortalidad Infantil como Indicador de Desarrollo** Conclusión: La mortalidad infantil es uno de los indicadores que más diferencia a los países en desarrollo de los países desarrollados. En los países en desarrollo, las tasas de mortalidad infantil son significativamente más altas, lo que evidencia deficiencias en el sistema de salud, la nutrición y el bienestar general de la población.

- Implicación: Para reducir las tasas de mortalidad infantil, los países en desarrollo deben enfocarse en mejorar la atención médica neonatal, aumentar el acceso a nutrición adecuada y mejorar las condiciones generales de vida.

59 Conclusión General

El análisis de los gráficos revela que los países desarrollados tienen mejores indicadores en términos de salud, educación y desarrollo económico en comparación con los países en desarrollo. El Producto Interno Bruto (PIB), la Esperanza de Vida y la Mortalidad Infantil son los indicadores que muestran las diferencias más notables entre estos dos grupos de países. Estos resultados refuerzan la necesidad de que los países en desarrollo inviertan en mejoras significativas en áreas clave como la salud pública, la educación y las infraestructuras económicas para cerrar la brecha con los países desarrollados.