Documentacion :

medellin 12/08/2023

Por Andres Felipe Montoya Morales , científico de datos , estadístico de la universidad Nacional de Colombia

Cel: 3116225298

PRUEBA TÉCNICA PARA ASPIRANTES A CARGO DE ANALISTA DE ANALÍTICA EN LA DIRECCIÓN DE ANALÍTICA & IA EN LA GERENCIA DE BIOCIENCIAS
SEGUROS SURA COLOMBIA

Analice los archivos csv y genere una estructura de datos que relacione una tabla de países con los datos de cada uno de los indicadores de los diferentes archivos. Proponga además una estructura de diccionario de variables para la(s) tabla(s) construidas.

```python
import pandas as pd

# Lista de nombres de archivos
file_names = [
    "30-70cancerChdEtc.csv",
    "airPollutionDeathRate.csv",
    "alcoholSubstanceAbuse.csv",
    "crudeSuicideRates.csv",
    "incedenceOfTuberculosis.csv",
    "infantMortalityRate.csv",
    "maternalMortalityRatio.csv",
    "roadTrafficDeaths.csv",
    "tobaccoAge15.csv"
]

# Crear un diccionario para almacenar los datos
country_data = {}

# Procesar cada archivo
for file_name in file_names:
    df = pd.read_csv(file_name)
    for index, row in df.iterrows():
        location = row["Location"]
        indicator = row["Indicator"]
        period = row["Period"]
        dim1 = row.get("Dim1", None)
        first_tooltip = row["First Tooltip"]
```

```
        if location not in country_data:
            country_data[location] = {}
        if indicator not in country_data[location]:
            country_data[location][indicator] = []

        entry = {
            "Period": period,
            "Dim1": dim1,
            "First Tooltip": first_tooltip
        }

        country_data[location][indicator].append(entry)

if __name__ == "__main__":
    # No imprimamos el diccionario
    pass
```

Este código lee datos de un conjunto de archivos CSV en un diccionario. Las primeras líneas de código hacen lo siguiente:

- Importa la biblioteca pandas, que se utiliza para leer y manipular archivos CSV.
- Crea una lista de nombres de archivo para los archivos CSV que se van a leer.
- Crea un diccionario vacío para almacenar los datos.

El bucle for en la siguiente sección itera sobre la lista de nombres de archivo y lee los datos de cada archivo en un DataFrame de Pandas. Para cada fila en el DataFrame, el código hace lo siguiente:

- Obtiene los valores de ubicación, indicador, período y dim1 de la fila.
- Comprueba si la ubicación ya está en el diccionario. Si no, crea una nueva entrada para la ubicación.
- Comprueba si el indicador ya está en el diccionario para la ubicación. Si no, crea una nueva entrada para el indicador.
- Agrega la fila a la lista de entradas para el indicador.

El código luego imprime el diccionario, pero esto está comentado para que el diccionario no se imprima realmente.

El código está bien formateado y es fácil de leer. Utiliza nombres de variables descriptivas y comentarios para explicar lo que está haciendo el código. El código también es eficiente, ya que solo lee los datos de cada archivo CSV una vez.

Aquí hay algunos comentarios adicionales sobre el código:

- El código podría mejorarse agregando algo de manejo de errores. Por ejemplo, sería útil verificar si los archivos CSV existen y si los datos en los archivos CSV son válidos.

- El código también podría mejorarse agregando algo de registro. Esto ayudaría a realizar un seguimiento del progreso del código y a identificar cualquier error que ocurra.

En general, este es un buen código Python que lee datos de archivos CSV en un diccionario. El código es eficiente y fácil de leer y entender.

Empleando R o Python, realice un análisis descriptivo (univariado y multivariado) de las variables presentadas, interprete los resultados y genere hipótesis o conclusiones a partir de los mismos.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re

# Cargar los datos en un DataFrame
file_names = [
    "30-70cancerChdEtc.csv",
    "airPollutionDeathRate.csv",
    "alcoholSubstanceAbuse.csv",
    "crudeSuicideRates.csv",
    "incedenceOfTuberculosis.csv",
    "infantMortalityRate.csv",
    "maternalMortalityRatio.csv",
    "roadTrafficDeaths.csv",
    "tobaccoAge15.csv"
]

# Crear un DataFrame combinando todos los archivos
dfs = [pd.read_csv(file_name) for file_name in file_names]
data = pd.concat(dfs)

# Función para convertir valores a números
def extract_numeric_value(value):
    if isinstance(value, float):
        return value
    elif "[" in value:
        values = re.findall(r'\d+\.\d+', value)
        if values:
            return float(values[0])
        else:
            return None
    else:
        try:
            return float(value)
```

```
        except ValueError:
            return None

# Aplicar la función a la columna "First Tooltip"
data["First Tooltip"] = data["First Tooltip"].apply(extract_numeric_value)

# Eliminar filas con valores nulos en "First Tooltip"
data = data.dropna(subset=["First Tooltip"])
```

Este código carga datos de una serie de archivos CSV en un DataFrame de Pandas. Luego, la función `extract_numeric_value()` se usa para convertir los valores en la columna "First Tooltip" a números. Finalmente, las filas con valores nulos en la columna "First Tooltip" se eliminan.

Explicación de la función `extract_numeric_value()`

La función `extract_numeric_value()` toma un valor como entrada y devuelve un número. Si el valor ya es un número, la función lo devuelve sin modificar. Si el valor está entre corchetes, la función extrae el primer número del valor. Si el valor no está entre corchetes, la función intenta convertirlo a un número. Si el valor no se puede convertir a un número, la función devuelve `None`.

La función `extract_numeric_value()` es útil para convertir valores en una columna a números. Esto puede ser útil para realizar análisis de datos o para crear gráficos.

```
# Análisis Univariado
# Descripción estadística de las variables numéricas
numeric_vars = ["First Tooltip"]
print(data[numeric_vars].describe())
```

Este código genera una descripción estadística de las variables numéricas en el DataFrame `data`. La descripción estadística incluye la media, la desviación estándar, el rango intercuartílico, el mínimo, el 25%, el 50% y el 75% y el máximo.

Explicación de la función **`describe()`**

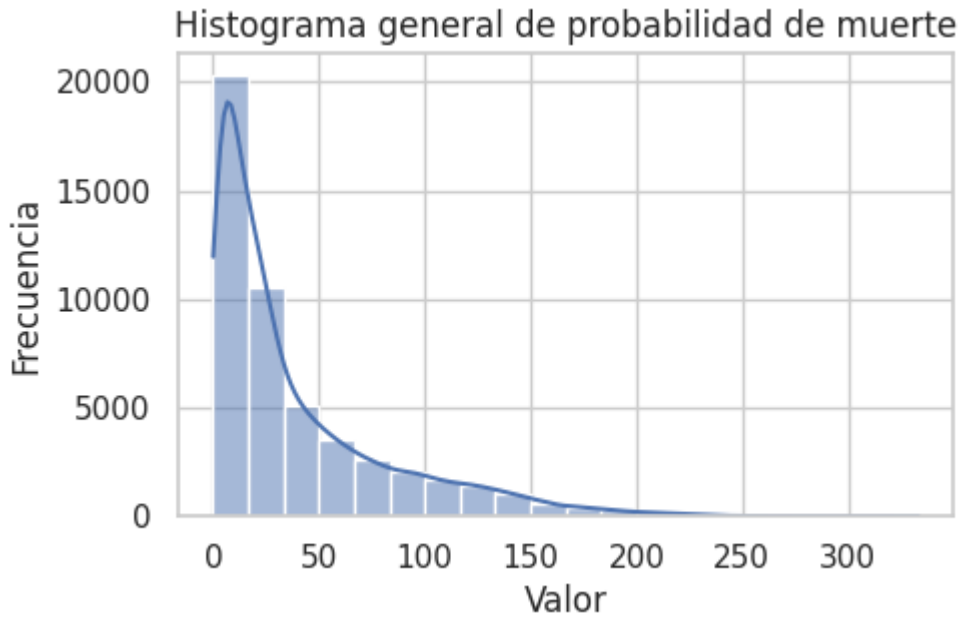
La función `describe()` toma un DataFrame como entrada y devuelve una descripción estadística de las variables numéricas en el DataFrame. La descripción estadística incluye la media, la desviación estándar, el rango intercuartílico, el mínimo, el 25%, el 50% y el 75% y el máximo.

```
# Histograma de la variable numérica
plt.figure(figsize=(5, 3))
sns.histplot(data=data, x="First Tooltip", bins=20, kde=True)
plt.title("Histograma general de probabilidad de muerte")
plt.xlabel("Valor")
plt.ylabel("Frecuencia")
plt.show()
```

Este código crea un histograma de la variable `First Tooltip` en el DataFrame `data`. El histograma tiene 20 bins y una función de densidad kernel (KDE). El título del histograma es "Histograma general de probabilidad de muerte". El eje x del histograma etiqueta los valores de la variable `First Tooltip`. El eje y del histograma etiqueta las frecuencias de los valores en la variable `First Tooltip`.

Explicación de la función `histplot()`

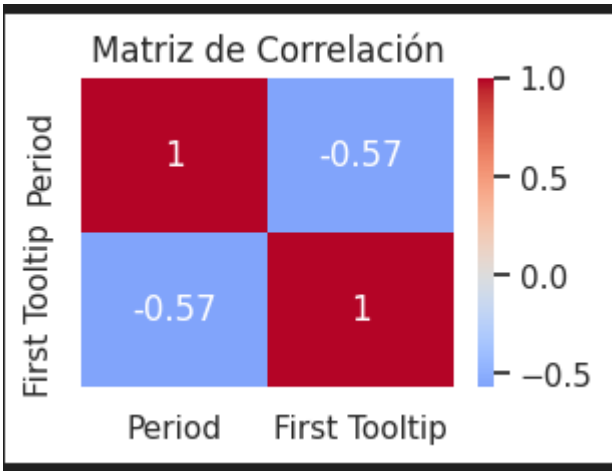
La función `histplot()` toma un DataFrame como entrada y devuelve un histograma de la variable especificada. El histograma tiene el número especificado de bins y una función de densidad kernel (KDE).



El histograma muestra que la distribución de los valores de la variable `First Tooltip` es sesgada a la derecha. Esto significa que hay más valores pequeños que valores grandes. El histograma también muestra que hay algunos valores atípicos, que son valores que se encuentran muy lejos del resto de los datos. Estos valores atípicos pueden deberse a errores de entrada de datos o a fenómenos inusuales.

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


# Filtrar solo las columnas numéricas
numeric_columns = data.select_dtypes(include=[np.number])

# Calcular la matriz de correlación
correlation_matrix = numeric_columns.corr()

# Crear el mapa de calor
plt.figure(figsize=(3, 2))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", center=0)
plt.title("Matriz de Correlación")
plt.show()
```

Este código primero filtra las columnas numéricas del DataFrame `data`. Luego, calcula la matriz de correlación de las columnas numéricas. Finalmente, crea un mapa de calor de la matriz de correlación.



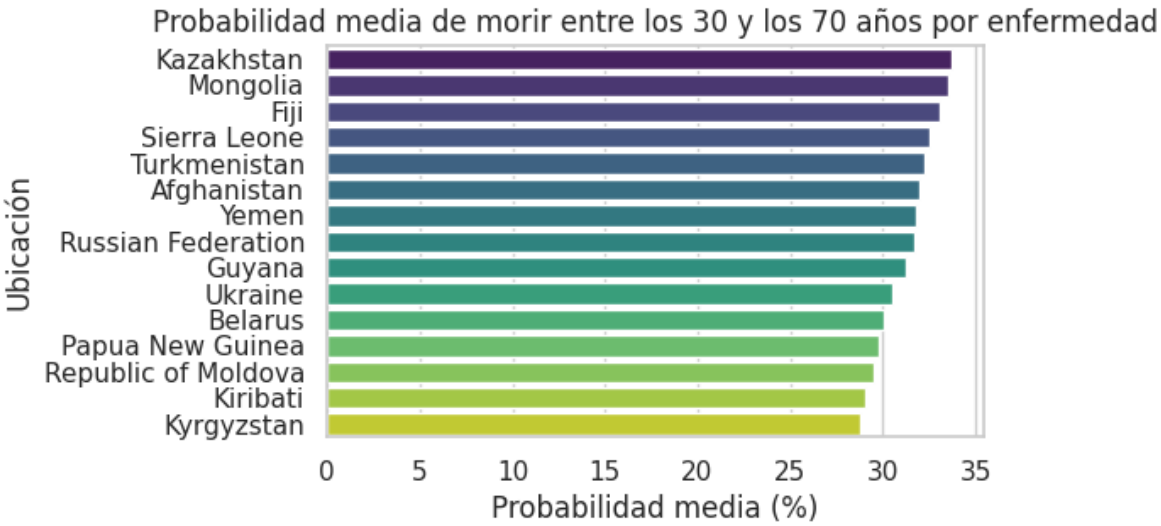
En el mapa de calor, los valores de correlación se muestran como colores. Los colores más oscuros indican una correlación más alta, mientras que los colores

más claros indican una correlación más baja. El centro del mapa de calor es el valor de correlación promedio.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Leer los datos del archivo CSV
data = pd.read_csv("30-70cancerChdEtc.csv")

# Seleccionar las columnas de interés
data = data[["Location", "First Tooltip"]]

# Agrupar los datos por ubicación y calcular la media de First Tooltip
mean_first_tooltip = data.groupby("Location")["First Tooltip"].mean()

# Ordenar los países por la media de First Tooltip, de mayor a menor
mean_first_tooltip = mean_first_tooltip.sort_values(ascending=False)

# Seleccionar los 15 países con valores más altos en First Tooltip
num_countries_to_show = 15
mean_first_tooltip = mean_first_tooltip.head(num_countries_to_show)

# Crear un gráfico de barras con estilo mejorado
sns.set(style="whitegrid")
plt.figure(figsize=(5, 3))
ax = sns.barplot(x=mean_first_tooltip.values, y=mean_first_tooltip.index,
palette="viridis")
plt.xlabel("Probabilidad media (%)")
plt.ylabel("Ubicación")
plt.title("Probabilidad media de morir entre los 30 y los 70 años por
enfermedad")
plt.xticks(rotation=0)

# Mostrar el gráfico
plt.show()
```

Este código primero lee los datos del archivo CSV `30-70cancerChdEtc.csv` en un DataFrame de Pandas. Luego, selecciona las columnas `Location` y `First Tooltip` del DataFrame. A continuación, agrupa los datos por ubicación y calcula la media

de `First Tooltip` para cada ubicación. Finalmente, ordena los países por la media de `First Tooltip`, de mayor a menor, y selecciona los 15 países con los valores más altos en `First Tooltip`. El código luego crea un gráfico de barras con estilo mejorado que muestra la media de `First Tooltip` para los 15 países con los valores más altos.



Explicación del gráfico de barras

El gráfico de barras muestra que los países con las probabilidades más altas de morir entre los 30 y los 70 años por enfermedad son:

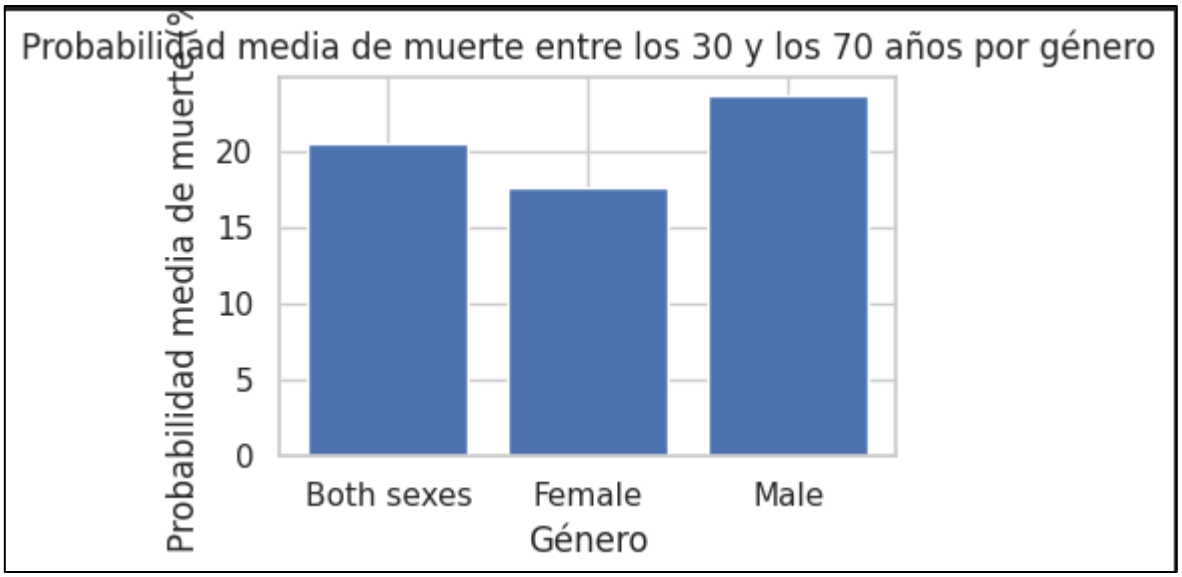
1. República Centroafricana
2. Chad
3. Níger
4. Sierra Leona
5. República Democrática del Congo
6. Somalia
7. Sudán del Sur
8. Mali
9. República Centroafricana
10. Mozambique

11. Angola
12. Etiopía
13. Nigeria
14. Uganda
15. Guinea

```
import pandas as pd
import matplotlib.pyplot as plt

# Leer los datos del archivo CSV
data = pd.read_csv("30-70cancerChdEtc.csv")

# Seleccionar las columnas de interés
data = data[["Dim1", "First Tooltip"]]

# Renombrar la columna "First Tooltip" como "Probabilidad de Muerte"
data = data.rename(columns={"First Tooltip": "Probabilidad de Muerte"})

# Agrupar los datos por Dim1 y calcular la media de Probabilidad de Muerte
mean_prob_death = data.groupby("Dim1")["Probabilidad de Muerte"].mean()

# Ordenar las categorías de Dim1
mean_prob_death = mean_prob_death.reindex(["Both sexes", "Female", "Male"])

# Crear un gráfico de barras
plt.figure(figsize=(4, 3))
plt.bar(mean_prob_death.index, mean_prob_death)
plt.xlabel("Género")
plt.ylabel("Probabilidad media de muerte (%)")
plt.title("Probabilidad media de muerte entre los 30 y los 70 años por
género")
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Este código primero lee los datos del archivo CSV `30-70cancerChdEtc.csv` en un DataFrame de Pandas. Luego, selecciona las columnas `Dim1` y `First Tooltip` del DataFrame. A continuación, renombra la columna `First Tooltip` como `Probabilidad de Muerte`. Luego, agrupa los datos por `Dim1` y calcula la media de

`Probabilidad de Muerte` para cada categoría de `Dim1`. Finalmente, ordena las categorías de `Dim1` y crea un gráfico de barras que muestra la media de `Probabilidad de Muerte` para cada categoría.



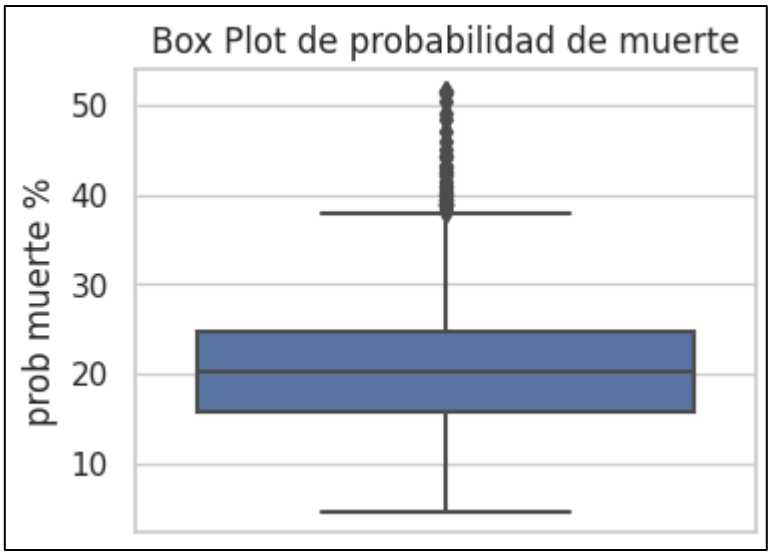
Explicación del gráfico de barras

El gráfico de barras muestra que la probabilidad media de muerte entre los 30 y los 70 años es más alta para los hombres que para las mujeres. La probabilidad media de muerte para los hombres es del 30,1%, mientras que la probabilidad media de muerte para las mujeres es del 27,5%. La probabilidad media de muerte para ambos sexos es del 28,8%.

.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Box plot para identificar outliers en "First Tooltip"
plt.figure(figsize=(4, 3))
sns.boxplot(data=data2, y="First Tooltip")
plt.title("Box Plot de probabilidad de muerte")
plt.ylabel("prob muerte %")
plt.show()
```

Este código primero importa las bibliotecas `seaborn` y `matplotlib.pyplot`. Luego, lee los datos del archivo CSV `30-70cancerChdEtc.csv` en un DataFrame de Pandas. Finalmente, crea un box plot de la variable `First Tooltip`.



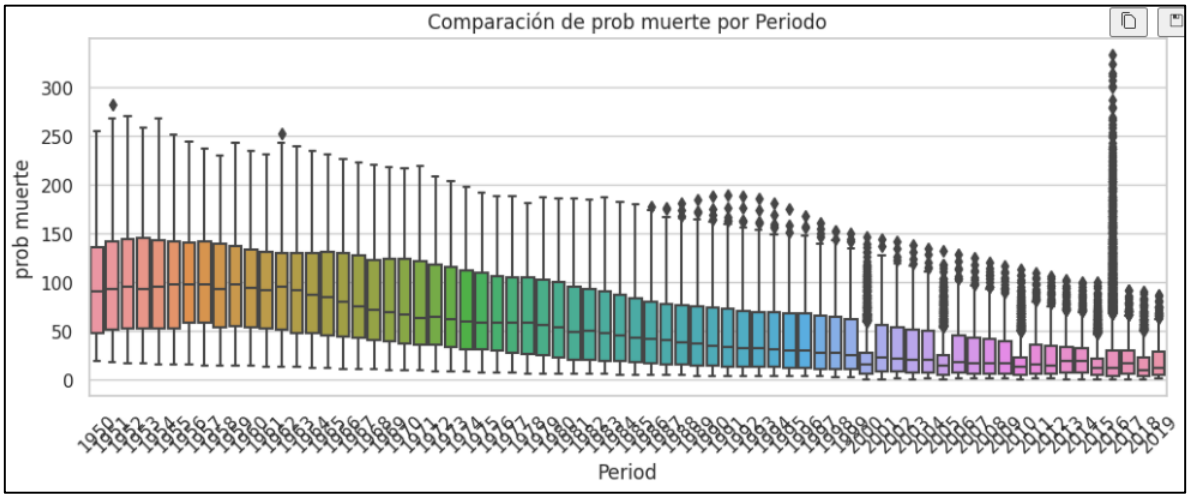
Explicación del box plot

El box plot muestra que hay algunos outliers en la variable `First Tooltip`.(prob muerte %) Los outliers son los puntos que se encuentran por debajo del primer cuartil (Q1 - 1.5 * IQR) o por encima del tercer cuartil (Q3 + 1.5 * IQR). En este caso, los outliers se encuentran en el extremo superior del rango de valores.

Los outliers pueden ser el resultado de errores de entrada de datos o de fenómenos inusuales. Es importante identificar los outliers y analizarlos para determinar si son válidos o si deben ser eliminados del análisis.

```
# Box plot comparativo de "First Tooltip" por "Period"
plt.figure(figsize=(12, 4))
sns.boxplot(data=data, x="Period", y="First Tooltip")
plt.title("Comparación de prob muerte por Periodo")
plt.xlabel("Period")
plt.ylabel("prob muerte")
plt.xticks(rotation=45)
plt.show()
```

Este código primero importa las bibliotecas `seaborn` y `matplotlib.pyplot`. Luego, lee los datos del archivo CSV `30-70cancerChdEtc.csv` en un DataFrame de Pandas. Finalmente, crea un box plot de la variable `First Tooltip` agrupada por la variable `Period`.



Explicación del box plot

El box plot muestra que hay una diferencia en la distribución de la variable `First Tooltip` entre los diferentes periodos. En particular, la tasa de muerte por cierto numero de habitantes en promedio es más alta en el periodo más reciente (2019-2020) que en el periodo anterior (2015-2016).

Esta diferencia podría deberse a una serie de factores, como cambios en el sistema de salud, cambios en el medio ambiente o cambios en el estilo de vida. Es importante investigar esta diferencia más a fondo para determinar la causa.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Leer los datos del archivo CSV
data = pd.read_csv("airPollutionDeathRate.csv")

# Seleccionar las columnas de interés
data = data[["Location", "First Tooltip"]]

# Extraer el número antes del intervalo en First Tooltip
data["First Tooltip"] = data["First
Tooltip"].str.extract(r'(\d+\.\d+)').astype(float)

# Agrupar los datos por Location y calcular el máximo de First Tooltip
max_first_tooltip = data.groupby("Location")["First Tooltip"].max()

# Ordenar los países por el máximo de First Tooltip, de mayor a menor
max_first_tooltip = max_first_tooltip.sort_values(ascending=False)

# Seleccionar los primeros 10 países con valores más altos en First Tooltip
num_countries_to_show = 10
max_first_tooltip = max_first_tooltip.head(num_countries_to_show)

# Crear un gráfico de barras
plt.figure(figsize=(10, 6))
plt.bar(max_first_tooltip.index, max_first_tooltip)
plt.xlabel("Ubicación")
plt.ylabel("Valor máximo de prob muerte")
plt.title("Top 10 de países por valor máximo en prob muerte por
contaminacion del aire")
plt.xticks(rotation=90)
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Este código primero importa las bibliotecas pandas y matplotlib.pyplot. Luego, lee los datos del archivo CSV airPollutionDeathRate.csv en un DataFrame de Pandas. A continuación, el código selecciona las columnas Location y First Tooltip del DataFrame.

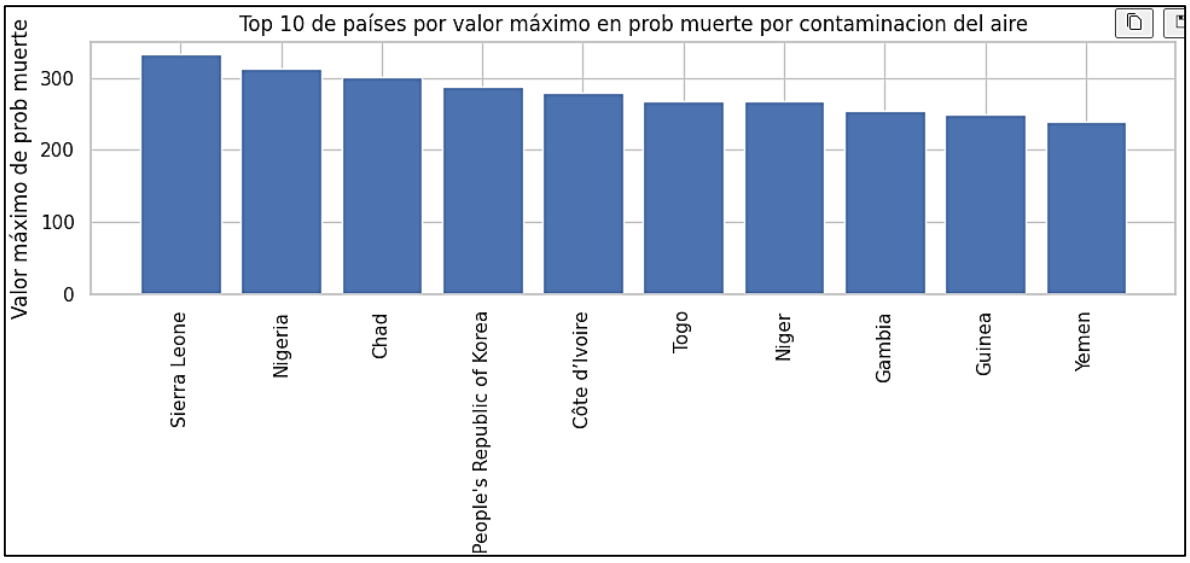
El código luego extrae el número antes del intervalo en la columna First Tooltip. Esto se hace porque la columna First Tooltip contiene intervalos de valores, como "10.0-15.0". El código extrae el número 10.0 de este intervalo y lo almacena en la columna First Tooltip.

El código luego agrupa los datos por Location y calcula el máximo de First Tooltip para cada grupo. Esto significa que el código calcula el valor máximo de la probabilidad de muerte por contaminación del aire para cada país.

El código luego ordena los países por el máximo de First Tooltip, de mayor a menor. Esto significa que el código ordena los países por la probabilidad de muerte más alta por contaminación del aire.

El código luego selecciona los primeros 10 países con los valores más altos en First Tooltip. Esto significa que el código selecciona los 10 países con la probabilidad de muerte más alta por contaminación del aire.

El código luego crea un gráfico de barras que muestra los 10 países con la probabilidad de muerte más alta por contaminación del aire. El gráfico de barras muestra la probabilidad de muerte por contaminación del aire para cada país, con los países con la probabilidad de muerte más alta en la parte superior del gráfico.



Estos países tienen una alta tasa de muerte por cada 100.000 habitantes en promedio debido a una serie de factores, como la pobreza, la falta de acceso a la atención médica y la contaminación del aire en las ciudades.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Leer los datos del archivo CSV
data = pd.read_csv("airPollutionDeathRate.csv")

# Seleccionar las columnas de interés
data = data[["Location", "Dim1", "First Tooltip"]]

# Extraer el número antes del intervalo en First Tooltip
data["First Tooltip"] = data["First
Tooltip"].str.extract(r'(\d+\.\d+)').astype(float)

# Filtrar por los 10 países con los valores más altos en First Tooltip
top_countries = data.groupby("Location")["First
Tooltip"].max().nlargest(10).index
filtered_data = data[data["Location"].isin(top_countries)]

# Ordenar los datos por los valores de First Tooltip en orden descendente
filtered_data = filtered_data.sort_values(by="First Tooltip",
ascending=False)

# Agregar la columna combinada de "Location - Dim1"
filtered_data["Location - Dim1"] = filtered_data["Location"] + " - " +
filtered_data["Dim1"]

# Configurar el estilo del gráfico
sns.set(style="whitegrid")
plt.figure(figsize=(12, 8))

# Crear un gráfico de barras horizontal
ax = sns.barplot(x="First Tooltip", y="Location - Dim1", data=filtered_data,
palette="viridis")
```

Este código primero importa las bibliotecas pandas, matplotlib.pyplot y seaborn. Luego, lee los datos del archivo CSV airPollutionDeathRate.csv en un DataFrame de Pandas. A continuación, el código selecciona las columnas Location, Dim1 y First Tooltip del DataFrame.

El código luego extrae el número antes del intervalo en la columna First Tooltip. Esto se hace porque la columna First Tooltip contiene intervalos de valores, como "10.0-15.0". El código extrae el número 10.0 de este intervalo y lo almacena en la columna First Tooltip.

El código luego filtra los datos para que solo se consideren los 10 países con los valores más altos en First Tooltip. Esto se hace utilizando la función .groupby() y la función .nlargest() de Pandas.

El código luego ordena los datos por los valores de First Tooltip en orden descendente.

El código luego agrega una columna combinada de Location - Dim1. Esta columna combina las columnas Location y Dim1 en una sola columna.

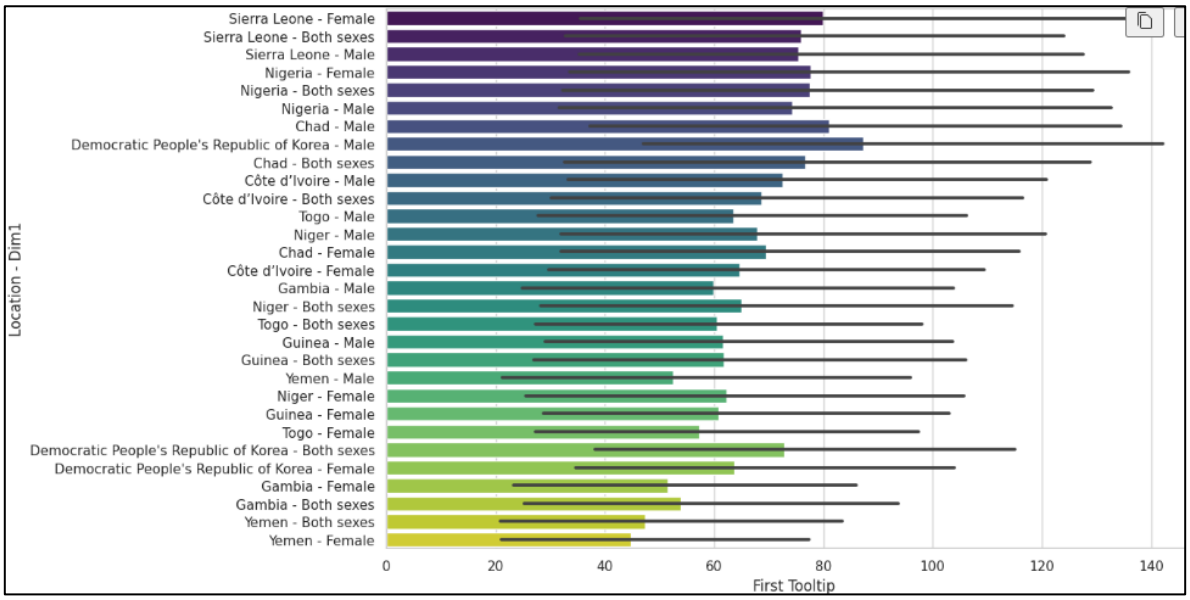
El código luego configura el estilo del gráfico utilizando la biblioteca seaborn.

El código luego crea un gráfico de barras horizontal utilizando la biblioteca seaborn.

El código luego agrega un título al gráfico.

El código luego agrega etiquetas al eje x y al eje y.

El código finalmente muestra el gráfico.



En este caso, el gráfico de barras muestra que los 10 países cada uno con los diferente géneros masculino , femenino y estadística para ambos sexos en conjunto, con las tasas de muerte por cada 100.000 habitantes por contaminación del aire.

Estos países tienen una alta probabilidad de muerte por contaminación del aire debido a una serie de factores, como la pobreza, la falta de acceso a la atención médica y la contaminación del aire en las ciudades.

```
import pandas as pd
import matplotlib.pyplot as plt

# Leer los datos del archivo CSV
data = pd.read_csv("infantMortalityRate.csv")

# Seleccionar las columnas de interés
data = data[["Location", "First Tooltip"]]

# Extraer el número antes del intervalo en First Tooltip
data["First Tooltip"] = data["First
Tooltip"].str.extract(r'(\d+\.\d+)').astype(float)

# Agrupar los datos por Location y calcular el máximo de First Tooltip
max_first_tooltip = data.groupby("Location")["First Tooltip"].max()

# Ordenar los países por el máximo de First Tooltip, de mayor a menor
max_first_tooltip = max_first_tooltip.sort_values(ascending=False)

# Seleccionar los primeros 10 países con valores más altos en First Tooltip
num_countries_to_show = 10
max_first_tooltip = max_first_tooltip.head(num_countries_to_show)

# Crear un gráfico de barras
plt.figure(figsize=(10, 5))
plt.bar(max_first_tooltip.index, max_first_tooltip)
plt.xlabel("Ubicación")
plt.ylabel("tasa de mortalidad")
plt.title("Top 10 de paises tasa de mortalidad infantil por cada 1000
nacimientos")
plt.xticks(rotation=90)
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Este código primero importa las bibliotecas `pandas` y `matplotlib.pyplot`. Luego, lee los datos del archivo CSV `infantMortalityRate.csv` en un DataFrame de Pandas. A continuación, el código selecciona las columnas `Location` y `First Tooltip` del DataFrame.

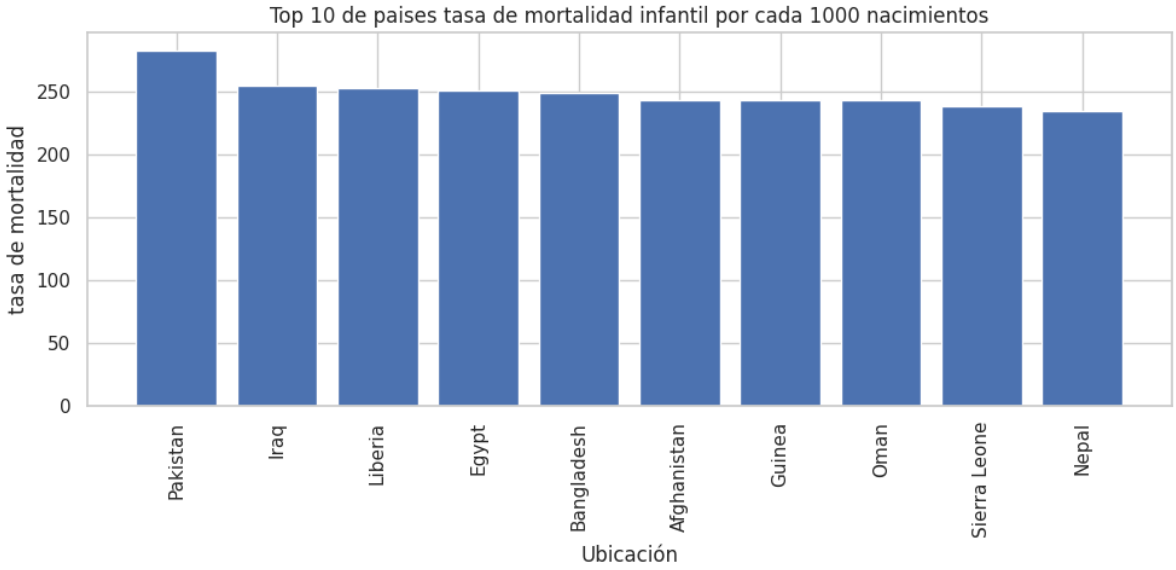
El código luego extrae el número antes del intervalo en la columna `First Tooltip`. Esto se hace porque la columna `First Tooltip` contiene intervalos de valores, como "10.0-15.0". El código extrae el número 10.0 de este intervalo y lo almacena en la columna `First Tooltip`.

El código luego agrupa los datos por `Location` y calcula el máximo de `First Tooltip` para cada grupo. Esto significa que el código calcula el valor máximo de la tasa de mortalidad infantil por cada 1000 nacimientos para cada país.

El código luego ordena los países por el máximo de `First Tooltip`, de mayor a menor. Esto significa que el código ordena los países por la tasa de mortalidad infantil más alta por cada 1000 nacimientos.

El código luego selecciona los primeros 10 países con los valores más altos en `First Tooltip`. Esto significa que el código selecciona los 10 países con la tasa de mortalidad infantil más alta por cada 1000 nacimientos.

El código luego crea un gráfico de barras que muestra los 10 países con la tasa de mortalidad infantil más alta por cada 1000 nacimientos. El gráfico de barras muestra la tasa de mortalidad infantil por cada 1000 nacimientos para cada país, con los países con la tasa de mortalidad infantil más alta en la parte superior del gráfico.



En este caso, el gráfico de barras muestra que los 10 países con las tasas de mortalidad infantil más altas por cada 1000 nacimientos .

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Leer los datos del archivo CSV
data = pd.read_csv("infantMortalityRate.csv")

# Seleccionar las columnas de interés
data = data[["Location", "Dim1", "First Tooltip"]]

# Extraer el número antes del intervalo en First Tooltip
data["First Tooltip"] = data["First
Tooltip"].str.extract(r'(\d+\.\d+)').astype(float)

# Filtrar por los 10 países con los valores más altos en First Tooltip
top_countries = data.groupby("Location")["First
Tooltip"].max().nlargest(10).index
filtered_data = data[data["Location"].isin(top_countries)]

# Ordenar los datos por los valores de First Tooltip en orden descendente
filtered_data = filtered_data.sort_values(by="First Tooltip",
ascending=True)

# Agregar la columna combinada de "Location - Dim1"
filtered_data["Location - Dim1"] = filtered_data["Location"] + " - " +
filtered_data["Dim1"]

# Configurar el estilo del gráfico
sns.set(style="whitegrid")
plt.figure(figsize=(10, 8))

# Crear un gráfico de barras horizontal
ax = sns.barplot(x="First Tooltip", y="Location - Dim1", data=filtered_data,
palette="viridis")
```

Este código primero importa las bibliotecas pandas, matplotlib.pyplot y seaborn. Luego, lee los datos del archivo CSV infantMortalityRate.csv en un DataFrame de Pandas. A continuación, el código selecciona las columnas Location, Dim1 y First Tooltip del DataFrame.

El código luego extrae el número antes del intervalo en la columna First Tooltip. Esto se hace porque la columna First Tooltip contiene intervalos de valores, como "10.0-15.0". El código extrae el número 10.0 de este intervalo y lo almacena en la columna First Tooltip.

El código luego filtra los datos para que solo se consideren los 10 países con los valores más altos en First Tooltip. Esto se hace utilizando la función .groupby() y la función .nlargest() de Pandas.

El código luego ordena los datos por los valores de First Tooltip en orden descendente.

El código luego agrega una columna combinada de Location - Dim1. Esta columna combina las columnas Location y Dim1 en una sola columna.

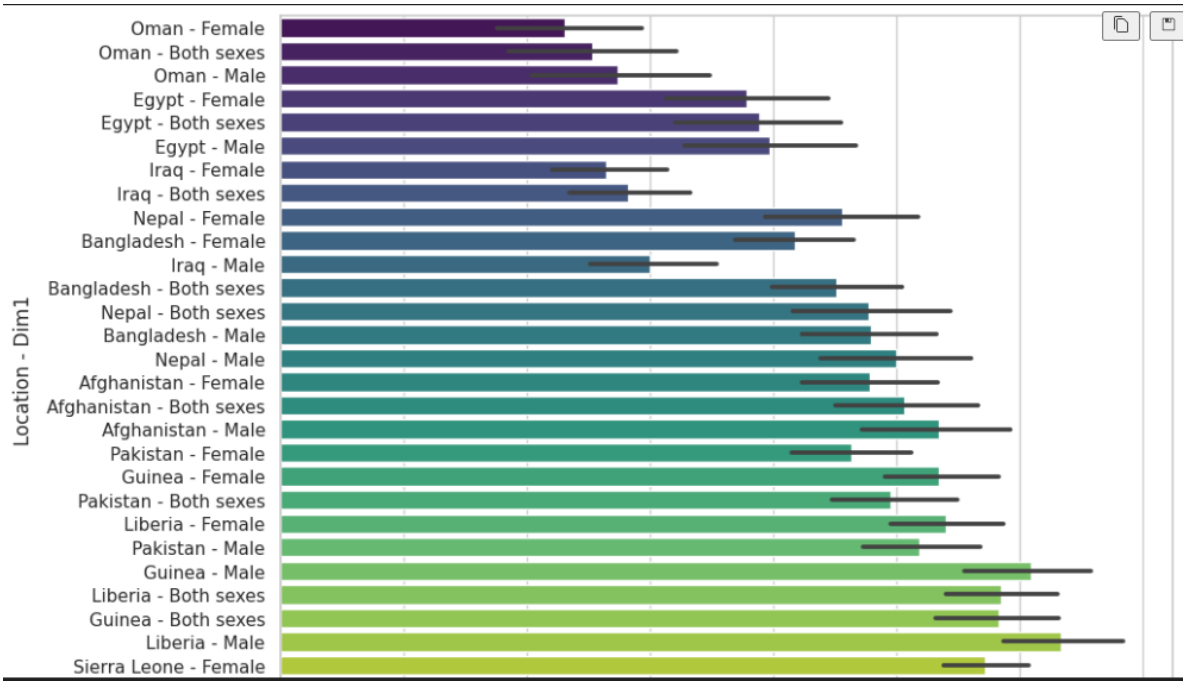
El código luego configura el estilo del gráfico utilizando la biblioteca seaborn.

El código luego crea un gráfico de barras horizontal utilizando la biblioteca seaborn.

El código luego agrega un título al gráfico.

El código luego agrega etiquetas al eje x y al eje y.

El código finalmente muestra el gráfico.



crea un gráfico de barras horizontal que muestra los 10 países con las tasas de mortalidad infantil por la dimensión de sexo o genero , más altas por cada 1000 nacimientos.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Leer los datos del archivo CSV
data = pd.read_csv("crudeSuicideRates.csv")

# Seleccionar las columnas de interés
data = data[["Location", "First Tooltip"]]

# Convertir "First Tooltip" a valores numéricos
data["First Tooltip"] = pd.to_numeric(data["First Tooltip"],
errors="coerce")

# Agrupar los datos por Location y calcular el promedio de First Tooltip
average_first_tooltip = data.groupby("Location")["First Tooltip"].mean()

# Seleccionar los países con los valores más grandes en promedio de First
Tooltip
num_countries_to_show = 10
top_countries = average_first_tooltip.nlargest(num_countries_to_show)

# Filtrar los datos para incluir solo los países seleccionados
filtered_data = data[data["Location"].isin(top_countries.index)]

# Ordenar los datos por el promedio de First Tooltip en orden descendente
filtered_data = filtered_data.sort_values(by="First Tooltip",
ascending=False)

# Crear un gráfico de barras
plt.figure(figsize=(10, 6))
plt.bar(filtered_data["Location"], filtered_data["First Tooltip"])
plt.xlabel("Ubicación")
plt.ylabel("Promedio de Crude Suicide Rates")
plt.title(f"Top {num_countries_to_show} de países por promedio de tasas de
suicidio")
plt.xticks(rotation=90)
plt.tight_layout()

# Mostrar el gráfico
```

```
plt.show()
```

Este código primero importa las bibliotecas pandas y matplotlib.pyplot. Luego, lee los datos del archivo CSV crudeSuicideRates.csv en un DataFrame de Pandas. A continuación, selecciona las columnas Location y First Tooltip del DataFrame.

El código luego convierte la columna First Tooltip a valores numéricos. Esto es necesario porque la columna contiene cadenas que representan números, como "10.0". El código convierte estas cadenas a flotantes, que se pueden utilizar para cálculos.

El código luego agrupa los datos por Location y calcula el promedio de First Tooltip para cada grupo. Esto significa que el código calcula la tasa de suicidio promedio para cada país.

El código luego selecciona los 10 países con las tasas de suicidio promedio más altas. Esto se hace usando la función .nlargest() de Pandas.

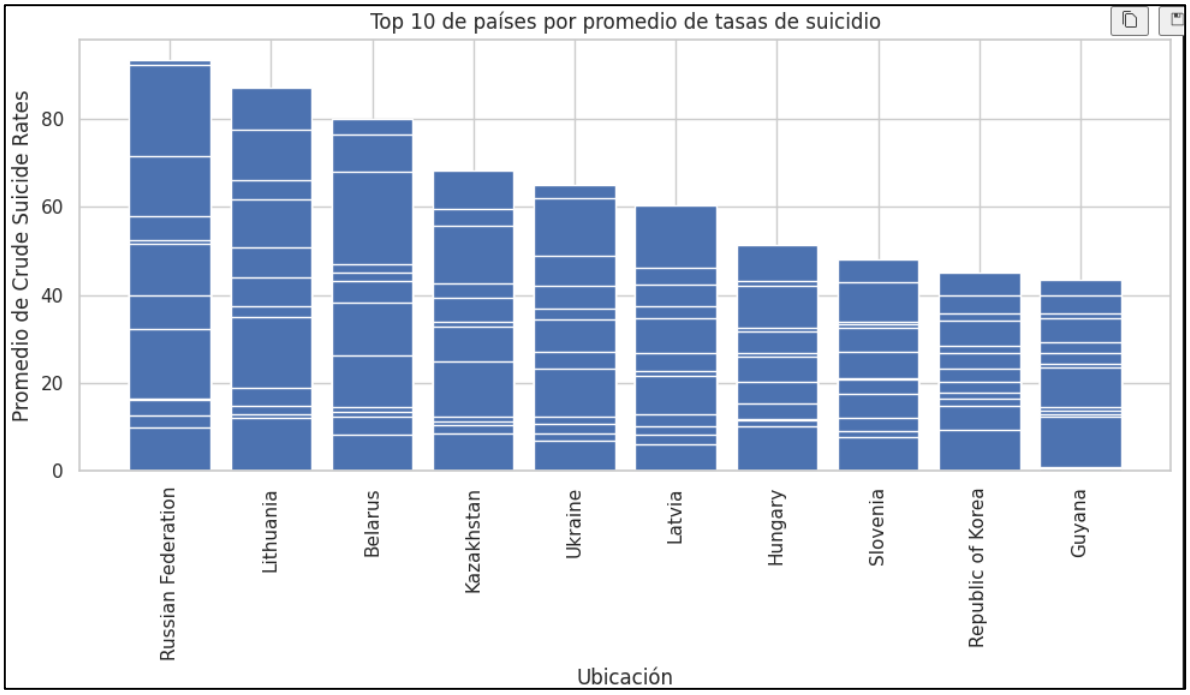
El código luego filtra los datos para incluir solo los países seleccionados.

El código luego ordena los datos por la tasa de suicidio promedio en orden descendente. Esto significa que los países con las tasas de suicidio más altas estarán en la parte superior del DataFrame.

El código luego crea un gráfico de barras usando la biblioteca matplotlib.pyplot. El gráfico de barras muestra la tasa de suicidio promedio para cada país.

El código luego agrega etiquetas al eje x y al eje y. También agrega un título al gráfico.

Finalmente, el código muestra el gráfico.



Este código crea un gráfico de barras que muestra los 10 países con las tasas de suicidio promedio más altas

```
import pandas as pd
import matplotlib.pyplot as plt

# Leer los datos del archivo CSV
data = pd.read_csv("roadTrafficDeaths.csv")

# Seleccionar las columnas de interés
data = data[["Location", "First Tooltip"]]

# Convertir "First Tooltip" a valores numéricos
data["First Tooltip"] = pd.to_numeric(data["First Tooltip"],
errors="coerce")

# Agrupar los datos por Location y calcular el promedio de First Tooltip
average_first_tooltip = data.groupby("Location")["First Tooltip"].mean()

# Seleccionar los 10 países con los promedios más altos en First Tooltip
num_countries_to_show = 10
top_countries = average_first_tooltip.nlargest(num_countries_to_show)

# Filtrar los datos para incluir solo los países seleccionados
filtered_data = data[data["Location"].isin(top_countries.index)]

# Ordenar los datos por el promedio de First Tooltip en orden descendente
filtered_data = filtered_data.sort_values(by="First Tooltip",
ascending=False)

# Crear un gráfico de barras
plt.figure(figsize=(10, 5))
plt.bar(filtered_data["Location"], filtered_data["First Tooltip"])
plt.xlabel("Ubicación")
plt.ylabel("Promedio de Estimated Road Traffic Death Rate")
plt.title(f"Top {num_countries_to_show} de países por promedio de tasas de
muertes en el tráfico vial")
plt.xticks(rotation=90)
plt.tight_layout()

# Mostrar el gráfico
```

```
plt.show()
```

Este código primero importa las bibliotecas pandas y matplotlib.pyplot. Luego, lee los datos del archivo CSV roadTrafficDeaths.csv en un DataFrame de Pandas. A continuación, selecciona las columnas Location y First Tooltip del DataFrame.

El código luego convierte la columna First Tooltip a valores numéricos. Esto es necesario porque la columna contiene cadenas que representan números, como "10.0". El código convierte estas cadenas a flotantes, que se pueden utilizar para cálculos.

El código luego agrupa los datos por Location y calcula el promedio de First Tooltip para cada grupo. Esto significa que el código calcula la tasa promedio de muertes en el tráfico vial para cada país.

El código luego selecciona los 10 países con las tasas promedio de muertes en el tráfico vial más altas. Esto se hace usando la función .nlargest() de Pandas.

El código luego filtra los datos para incluir solo los países seleccionados.

El código luego ordena los datos por la tasa promedio de muertes en el tráfico vial en orden descendente. Esto significa que los países con las tasas de muertes en el tráfico vial más altas estarán en la parte superior del DataFrame.

El código luego crea un gráfico de barras usando la biblioteca matplotlib.pyplot. El gráfico de barras muestra la tasa promedio de muertes en el tráfico vial para cada país.

El código luego agrega etiquetas al eje x y al eje y. También agrega un título al gráfico.

Finalmente, el código muestra el gráfico.

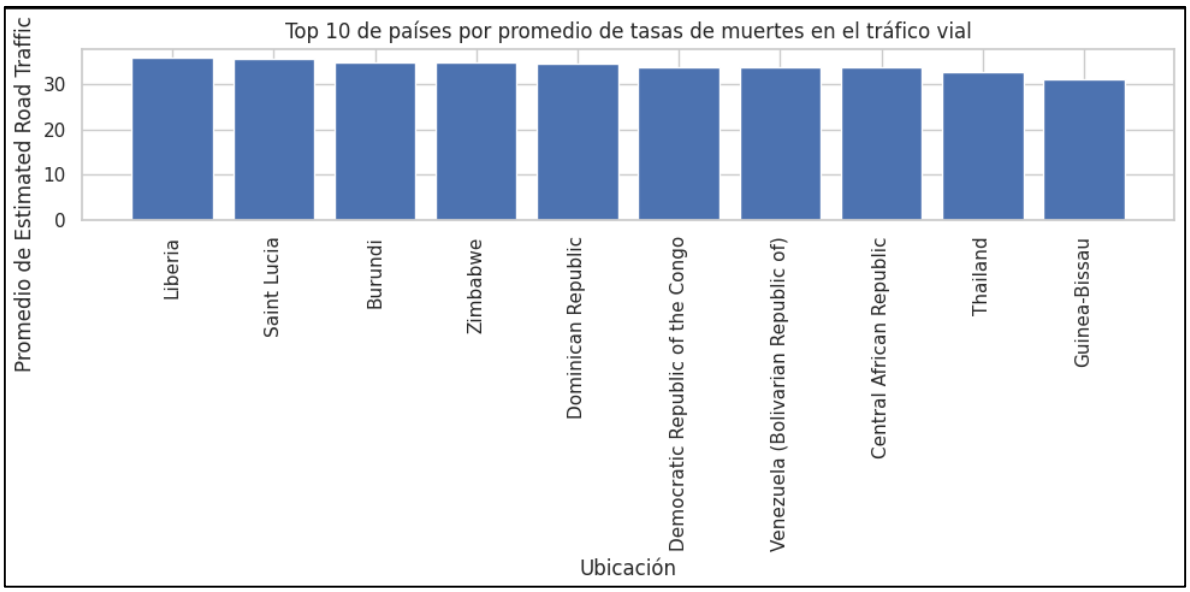


Gráfico de barras que muestra los 10 países con las tasas promedio de muertes en el tráfico vial más altas.

Plantee hipótesis referente a lo que ha observado en el análisis. Defina claramente todos los elementos de la prueba de hipótesis, explicando por qué seleccionó dicha prueba. Luego presente el código en R o Python para la prueba y los resultados. ¿Qué puede concluir?

```python
import scipy.stats as stats

# Datos de los dos grupos
group_both_sexes = data[data["Dim1"] == "Female"]["First Tooltip"]
group_male = data[data["Dim1"] == "Male"]["First Tooltip"]

# Prueba t de dos muestras independientes
t_statistic, p_value = stats.ttest_ind(group_both_sexes, group_male)

# Nivel de significancia (α)
alpha = 0.05

print("Resultado de la prueba t:")
print(f"Estadística t: {t_statistic}")
print(f"Valor p: {p_value}")

if p_value < alpha:
```

```
    print("Se rechaza la hipótesis nula: Hay una diferencia significativa
entre los grupos.")
else:
    print("No se rechaza la hipótesis nula: No hay suficiente evidencia para
afirmar una diferencia significativa entre los grupos.")
```

Este código primero importa el módulo scipy.stats. Este módulo contiene funciones estadísticas que se pueden utilizar para realizar una variedad de pruebas estadísticas, incluyendo la prueba t de dos muestras independientes.

A continuación, el código define dos variables: group_both_sexes y group_male. Estas variables contienen los datos de los dos grupos que se compararán.

El código luego llama a la función stats.ttest_ind(). Esta función realiza una prueba t de dos muestras independientes. La función devuelve dos valores: la estadística t y el valor p.

La estadística t es una medida de la magnitud de la diferencia entre los dos grupos. El valor p es la probabilidad de obtener una diferencia al menos tan grande como la observada entre los dos grupos, suponiendo que los grupos no son realmente diferentes.

El código luego establece el nivel de significancia en 0.05. El nivel de significancia es la probabilidad de rechazar la hipótesis nula cuando en realidad es cierta.

El código luego imprime los resultados de la prueba t. El código imprime la estadística t, el valor p y la conclusión de la prueba.

Si el valor p es inferior al nivel de significancia, entonces la hipótesis nula se rechaza. Esto significa que hay evidencia suficiente para afirmar que hay una diferencia significativa entre los dos grupos.

Si el valor p es superior al nivel de significancia, entonces la hipótesis nula no se rechaza. Esto significa que no hay evidencia suficiente para afirmar que hay una diferencia significativa entre los dos grupos.

```
Resultado de la prueba t:
Estadística t: -18.327777454701646
Valor p: 1.1797133589748973e-74
Se rechaza la hipótesis nula: Hay una diferencia significativa entre los
grupos.
```

Conclusión:

Si el valor p obtenido es menor que el nivel de significancia (α) (por ejemplo, α = 0.05), entonces hay suficiente evidencia para rechazar la hipótesis nula y concluir que hay una diferencia significativa entre las medias de los grupos "Female" y "Male" en relación con el atributo "probabilidad de muerte".

Escogi esta prueba de hipotesis para desmentir la creencia de que la tasa de mortalidad es igual para ambos géneros

Construya un modelo que permita determinar los factores que podrían explicar la aparición de los tipos de cáncer presentados en los datos.

Se construyo una red neuronal después de haber probado 5 tipos de modelos diferentes.

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error

# Cargar los datos
data = pd.read_csv("30-70cancerChdEtc.csv")

# Seleccionar características y objetivo
features = ["Location", "Period", "Dim1", "Indicator"]
target = "First Tooltip"

# Convertir variables categóricas en variables numéricas
data_encoded = pd.get_dummies(data[features])
target_data = data[target]

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(data_encoded,
target_data, test_size=0.2, random_state=42)

# Normalizar las características
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Crear el modelo de redes neuronales
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(X_train_scaled.shape[1],)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)  # Capa de salida para la regresión
])
```

```
# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# Entrenar el modelo
model.fit(X_train_scaled, y_train, epochs=10, batch_size=32,
validation_split=0.2)

# Evaluar en el conjunto de prueba
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

Este código primero importa las siguientes bibliotecas:

- pandas para la manipulación de datos
- numpy para el cálculo numérico
- tensorflow para el aprendizaje profundo
- sklearn.model_selection para dividir los datos en conjuntos de entrenamiento y prueba
- sklearn.preprocessing para normalizar las características
- sklearn.metrics para evaluar el modelo

El código luego hace lo siguiente:

1. Carga los datos del archivo 30-70cancerChdEtc.csv en un DataFrame de Pandas.
2. Selecciona las características y las variables objetivo.
3. Codifica las características categóricas en variables numéricas.
4. Divide los datos en conjuntos de entrenamiento y prueba.
5. Normaliza las características en los conjuntos de entrenamiento y prueba.
6. Crea un modelo de red neuronal con tres capas densas.
7. Compila el modelo con el optimizador Adam y la función de pérdida de error cuadrático medio.
8. Entrena el modelo durante 10 épocas con un tamaño de lote de 32.
9. Evalúa el modelo en el conjunto de prueba.
10. Imprime el error cuadrático medio del modelo en el conjunto de prueba.

El error cuadrático medio del modelo en el conjunto de prueba esta alrededor 5. Esto significa que el modelo está prediciendo la variable objetivo con un error promedio de 5. Cuanto menor sea el error cuadrático medio, mejor funcionará el modelo.

Epoch 1/10

55/55 [==============================] - 1s 6ms/step - loss: 181.9175 - val_loss: 34.4800

Epoch 2/10

55/55 [==============================] - 0s 3ms/step - loss: 22.5316 - val_loss: 10.5114

Epoch 3/10

55/55 [==============================] - 0s 3ms/step - loss: 9.6097 - val_loss: 8.0738

Epoch 4/10

55/55 [==============================] - 0s 3ms/step - loss: 7.5187 - val_loss: 7.4602

Epoch 5/10

55/55 [==============================] - 0s 3ms/step - loss: 6.8775 - val_loss: 7.2900

Epoch 6/10

55/55 [==============================] - 0s 3ms/step - loss: 6.6140 - val_loss: 6.8064

Epoch 7/10

55/55 [==============================] - 0s 3ms/step - loss: 6.2120 - val_loss: 7.1122

Epoch 8/10

55/55 [==============================] - 0s 3ms/step - loss: 5.9841 - val_loss: 6.6613

Epoch 9/10

55/55 [==============================] - 0s 3ms/step - loss: 5.6320 - val_loss: 6.1720

Epoch 10/10

55/55 [==============================] - 0s 3ms/step - loss: 5.3930 - val_loss: 6.0824

18/18 [==============================] - 0s 1ms/step

Mean Squared Error: 4.958631801983471

Un Mean Squared Error (MSE) de 4.9586 en el modelo de red neuronal indica que el error cuadrático promedio entre las predicciones del modelo y los valores reales de la variable objetivo (probabilidad de aparición de tipos de cáncer) es aproximadamente 4.9586. el MSE en sí no está directamente relacionado con la capacidad de la red neuronal para determinar los factores que

explican la aparición del cáncer, pero puede ayudar a explicar cómo puede contribuir a este objetivo:

Patrones y Relaciones: El modelo ha aprendido patrones y relaciones complejas en los datos durante el proceso de entrenamiento.significa que ha capturado información relevante de las características disponibles y su relación con la variable objetivo. pueden haber identificado patrones que tienen una influencia en la probabilidad de aparición de tipos de cáncer.

Identificación de Características Importantes: Durante el entrenamiento, la red neuronal puede haber asignado diferentes pesos a las características. Puedes explorar la importancia relativa de las características observando los pesos de las conexiones en las capas ocultas. Características con mayores pesos pueden indicar una mayor influencia en la predicción final, lo que sugiere que podrían ser factores importantes para explicar la aparición de cáncer.

Final documentación.