Seaborn

Introducción a la visualización de datos con Seaborn

CertiDevs

Índice de contenidos

1. Introducción

Seaborn es una biblioteca de **visualización de datos** en Python basada en **Matplotlib** que proporciona una interfaz de alto nivel para crear gráficos estadísticos informativos y avanzados.

Sitio web de Seaborn

Seaborn es especialmente útil para explorar y comprender los datos de manera más rápida y sencilla. Está diseñado para trabajar con **DataFrames** de **Pandas**, lo que facilita la manipulación de datos y la generación de visualizaciones más avanzadas.

Algunas de las características clave de Seaborn incluyen:

- **Estilos** y **paletas** de colores predefinidos para mejorar la apariencia de sus gráficos.
- Funciones para crear **gráficos estadísticos avanzados**, como gráficos de distribución, de regresión, de matriz de dispersión y más.
- **Integración con Pandas** para trabajar directamente con DataFrames.
- Funciones para visualizar **datos categóricos**, como gráficos de barras, de violín y de enjambre (swarm).

2. Tipos de gráficos

API de Seaborn

Métodos para crear **visualizaciones**:

1. `sns.lineplot()`: Crea un gráfico de línea que muestra la relación entre dos variables.
2. `sns.barplot()`: Crea un gráfico de barras que muestra la media y la incertidumbre de una variable en función de otra.
3. `sns.countplot()`: Crea un gráfico de barras que muestra la frecuencia de una variable categórica.
4. `sns.boxplot()`: Crea un gráfico de caja que muestra la distribución de una variable.
5. `sns.violinplot()`: Crea un gráfico de violín que muestra la distribución de una variable y refleja tanto la frecuencia como la forma de la distribución.
6. `sns.swarmplot()`: Crea un gráfico de abejas que muestra la relación entre dos variables y permite ver la concentración de los datos.
7. `sns.jointplot()`: Crea un gráfico de dos variables que incluye un gráfico de dispersión, un histograma de cada variable, y una curva de densidad ajustada.
8. `sns.pairplot()`: Crea una matriz de gráficos de dos variables que muestra la relación entre cada pareja de variables en un conjunto de datos.
9. `sns.lmplot()`: Crea un gráfico de regresión lineal que muestra la relación entre dos variables y ajusta una recta a través de los datos.
10. `sns.regplot()`: Crea un gráfico de regresión que muestra la relación entre dos variables y ajusta

una curva a través de los datos.

11. `sns.kdeplot()`: Crea un gráfico de curva de densidad que muestra la distribución de una variable.
12. `sns.displot()`: Crea un gráfico de distribución que muestra la distribución de una variable y permite personalizar la apariencia de la curva de densidad ajustada.
13. `sns.histplot()`: Crea un gráfico de histograma que muestra la distribución de una variable.
14. `sns.scatterplot()`: Crea un gráfico de dispersión que muestra la relación entre dos variables.

3. Limitaciones

Algunos problemas de rendimiento: Seaborn puede tener problemas de rendimiento con grandes cantidades de datos, lo que puede ser un problema para ciertas aplicaciones, especialmente si se intenta realizar gráficos de tipo `pairplot`.

4. Alternativas

- **Plotly**: https://plotly.com/python/
- **Bokeh**: https://docs.bokeh.org/en/latest/index.html
- **Vega-Altair**: https://altair-viz.github.io/
- Ver librerías en **GitHub**: https://github.com/topics/data-visualization?l=python

5. Usar seaborn

Para comenzar a usar Seaborn, primero debe ser instalarlo con `pip`:

```
pip install seaborn
```

Una vez instalado, puede **importar** Seaborn y comenzar a crear visualizaciones de datos:

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

Cargar un conjunto de datos de ejemplo

```
data = sns.load_dataset('tips')
```

Crear un diagrama de dispersión

```
sns.scatterplot(x='total_bill', y='tip', data=data)
```

Mostrar el gráfico

```
plt.show()
```

6. Mostrar gráficos automáticamente

Es cierto que en algunos entornos, no es necesario utilizar `plt.show()` para mostrar un gráfico.

`plt.show()` se utiliza generalmente en scripts de Python y en la línea de comandos para asegurarse de que se muestre el gráfico.

Sin embargo, en entornos interactivos, como **Jupyter Notebook** y **Google Colab**, a menudo no es necesario utilizar `plt.show()` para visualizar gráficos.

En Jupyter Notebook y Google Colab, puedes utilizar la siguiente línea de código al comienzo de tu cuaderno para que los gráficos se muestren automáticamente sin la necesidad de llamar a `plt.show()`:

```
%matplotlib inline
```

Este comando, conocido como "comando mágico", le indica al entorno de Jupyter que muestre automáticamente los gráficos en línea, es decir, directamente en el cuaderno, en lugar de abrir una ventana separada.

Después de ejecutar este comando, cualquier gráfico creado con Matplotlib o Seaborn **se mostrará automáticamente** al final de la celda de código sin tener que llamar a `plt.show()`.

Aunque en estos entornos no es necesario utilizar `plt.show()`, no afectará negativamente tu código si lo incluyes. Por lo tanto, si estás trabajando en un entorno interactivo y deseas garantizar la compatibilidad con otros entornos, aún puedes incluir `plt.show()` en tu código.