Análisis Exploratorio

de Datos

Sesión 5

Visualización de Datos con Seaborn

La visualización de datos es una parte esencial del análisis exploratorio de datos (EDA). Permite comprender patrones, tendencias y relaciones en los datos de manera intuitiva. Seaborn es una librería de Python basada en Matplotlib que facilita la creación de gráficos estadísticos atractivos y fáciles de interpretar.

Características Principales de Seaborn

1 **Integración con Pandas**

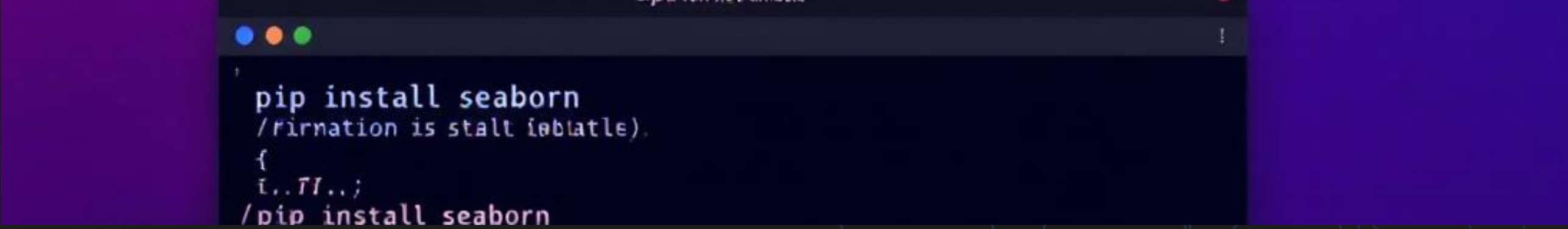
Seaborn trabaja directamente con DataFrames de Pandas, lo que facilita la visualización de datos sin necesidad de transformaciones adicionales. Esta integración permite un flujo de trabajo más eficiente durante el análisis de datos.

2 **Estética Mejorada**

Proporciona temas de diseño y paletas de colores atractivos por defecto, lo que mejora la presentación de los gráficos sin necesidad de configuraciones avanzadas. Esto permite crear visualizaciones profesionales con poco esfuerzo.

3 **Gráficos Estadísticos**

Incluye funciones específicas para visualizar distribuciones, relaciones y datos categóricos, como histogramas, diagramas de dispersión y gráficos de caja (boxplots), facilitando el análisis estadístico visual.



Importación e Instalación

Instalación

Si aún no tienes Seaborn instalado, puedes hacerlo con el siguiente comando en la terminal

```
pip install seaborn
```

Importación Básica

Para utilizar Seaborn, primero debemos importarla junto con otras librerías complementarias como Matplotlib y Pandas

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

Configuración

Una vez importadas las librerías, podemos comenzar a crear gráficos de manera sencilla utilizando los datos de Pandas y aplicando los estilos predeterminados de Seaborn o personalizándolos según nuestras necesidades.

Gráficos de Distribución

Histplot

La función histplot() es la recomendada en Seaborn para visualizar histogramas, reemplazando al antiguo distplot(). Permite examinar la distribución de los datos, identificar asimetrías, sesgos o valores atípicos, y evaluar la forma y dispersión de la variable.

Ejemplo en Python

```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Datos simulados (Distribución Normal)
data = np.random.randn(1000)

# Gráfico de histograma con KDE activado
sns.histplot(data, kde=True, color="blue", bins=30)
plt.title("Distribución de Datos con KDE")
plt.xlabel("Valores")
plt.ylabel("Frecuencia")
plt.show()
```

Salida:

- Un histograma con 30 bins (barras) que representa la frecuencia de los datos.
- Una curva KDE (Kernel Density Estimation) opcional que suaviza la distribución.

Gráficos de Distribución

Kdeplot

La función kdeplot() se usa cuando solo queremos la curva de densidad sin el histograma. Es ideal para obtener una visualización más suave de la distribución de los datos.

Ejemplo en Python

```
sns.kdeplot(data, shade=True, color="red")
plt.title("Distribución de Datos con KDE")
plt.xlabel("Valores")
plt.ylabel("Densidad")
plt.show()
```

Salida:

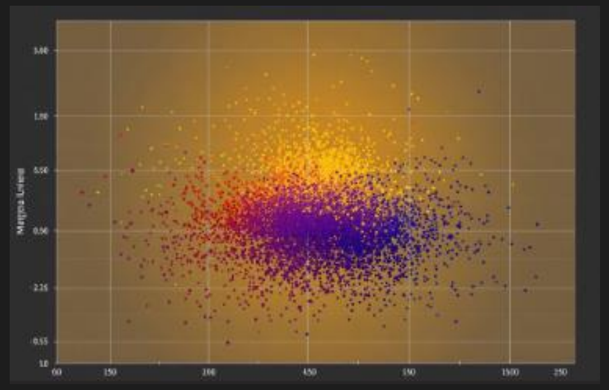
- Una curva KDE que estima la densidad de probabilidad de la variable.
- Sombra bajo la curva para visualizar mejor la distribución (shade=True).

Gráficos de Distribución

¿Cuándo usar histplot() y kdeplot()?

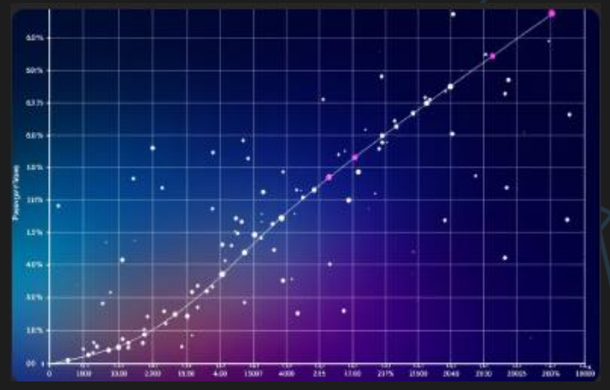
Método	Cuándo usarlo
histplot()	Para analizar frecuencia y distribución de datos discretos o continuos.
kdeplot()	Cuando queremos una visualización más suave de la distribución sin histograma.
Ambos combinados	Para obtener una visión más detallada y comparar distribuciones.

Gráficos de Dispersión y Correlación



Jointplot

El jointplot() combina un gráfico de dispersión con histogramas marginales, permitiendo observar la relación entre dos variables y la distribución individual de cada una. Es ideal para analizar correlaciones, detectar patrones y identificar valores atípicos.



Pairplot

El pairplot() genera una matriz de gráficos de dispersión para múltiples variables, permitiendo analizar visualmente todas las correlaciones en un conjunto de datos. Es perfecto para comparar relaciones entre múltiples variables en un solo gráfico.

Regplot

El regplot() es ideal para visualizar la relación lineal entre dos variables numéricas mediante una línea de regresión. Permite analizar tendencias, detectar patrones y probar la existencia de relaciones lineales entre variables.

Gráficos de Variables Categóricas

Barplot
El barplot() es útil para comparar la media de una variable numérica en diferentes categorías. Muestra la media de una variable numérica agrupada por una variable categórica e incluye barras de error para representar la dispersión de los datos.

Countplot
El countplot() es ideal para visualizar la frecuencia de cada categoría dentro de una variable categórica. Muestra la cantidad de observaciones en cada categoría y permite comparar la distribución de frecuencias en distintos grupos.

Boxplot
El boxplot() (gráfico de caja y bigotes) es excelente para analizar la distribución, asimetría y outliers en los datos. Visualiza la mediana, rango intercuartil (IQR) y valores atípicos de una variable numérica en diferentes categorías.

Violinplot
El violinplot() combina el boxplot con una estimación de densidad para visualizar mejor la distribución de los datos. Muestra la forma de la distribución en cada categoría y permite comparar la dispersión de los valores en diferentes grupos.

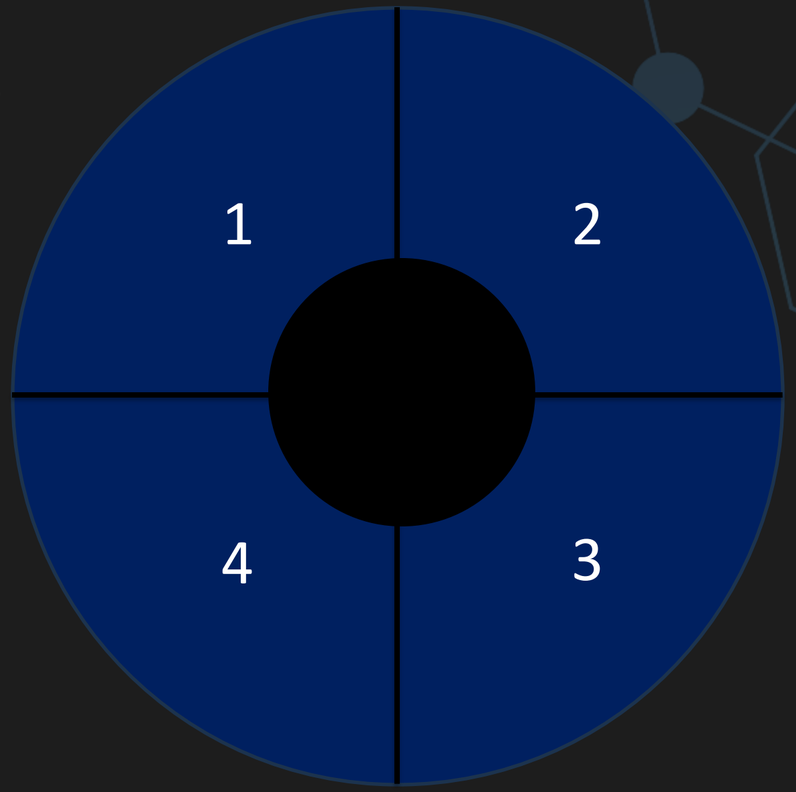
Heatmap: Visualización de Matrices

Interpretación

Identificar patrones y correlaciones

Personalización

Ajustar colores, anotaciones y formato



Creación

Generar matriz de correlación y aplicar sns.heatmap().

Preparación

Importar librerías y preparar datos

Grillas de Gráficos: PairGrid y FacetGrid

1

Crear objeto Grid

Inicializar PairGrid o FacetGrid con los datos y variables a visualizar. PairGrid para relaciones entre múltiples variables y FacetGrid para segmentar por categorías.

2

Mapear funciones

Aplicar diferentes tipos de gráficos a las distintas partes de la grilla usando métodos como map(), map_diag() y map_offdiag() para PairGrid o map() para FacetGrid.

3

Personalizar

Ajustar títulos, etiquetas, leyendas y otros elementos visuales para mejorar la interpretación de los datos y la presentación general.

4

Visualizar

Mostrar la grilla completa con plt.show() o guardarla como imagen para incluirla en informes o presentaciones.

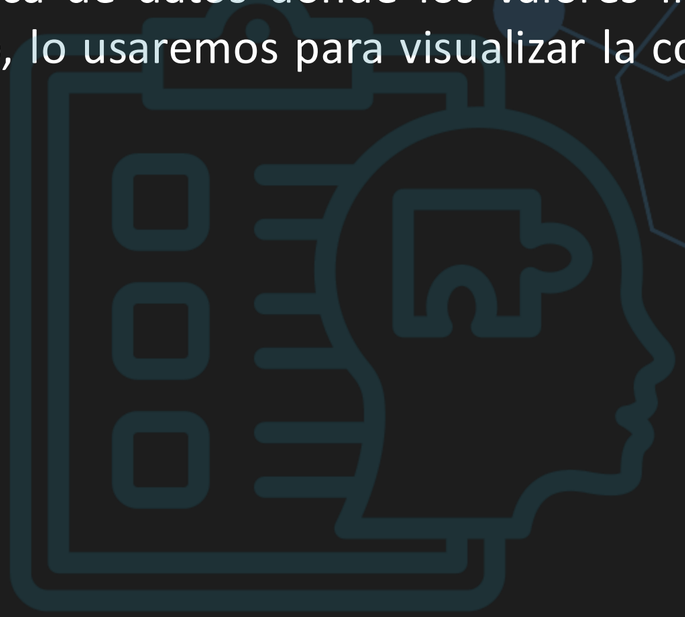
Comparación entre PairGrid y FacetGrid

Característica	PairGrid	FacetGrid
Tipo de datos	Numéricos	Numéricos o categóricos
Relación entre variables	Muestra combinaciones entre múltiples variables	Separa los gráficos por categorías
Personalización	Muy flexible (distintos tipos de gráficos en diagonal y fuera de la diagonal)	Se centra en un solo tipo de gráfico por categoría
Uso recomendado	Análisis de correlaciones y tendencias entre varias variables numéricas	Comparaciones dentro de subgrupos categóricos

Seaborn es una herramienta poderosa para la visualización de datos en Python. En esta sesión, hemos cubierto una amplia gama de gráficos, desde distribuciones y correlaciones hasta gráficos categóricos y matrices. Con estas herramientas, podrás explorar y comunicar insights de manera efectiva en tus proyectos de ciencia de datos.

Actividad Práctica Guiada: Generar Un Heatmap para Visualizar la Correlación Entre Variables

En este ejercicio práctico, aprenderás a generar un heatmap utilizando la librería Seaborn en Python. Un heatmap es una representación gráfica de datos donde los valores individuales contenidos en una matriz se representan con colores. En este caso, lo usaremos para visualizar la correlación entre variables en un conjunto de datos.



Requerimientos

1. **Importar Librerías**
2. **Crear un Conjunto de Datos**
3. **Calcular la Matriz de Correlación**
4. **Generar el Heatmap**
5. **Interpretación del Heatmap**
6. **Personalización del Heatmap**
7. **Visualización.**

El detalle de la actividad se encuentra en la guía de estudio de la sesión.

Preguntas

Sección de preguntas

Análisis Exploratorio

de Datos

Continúe con las actividades