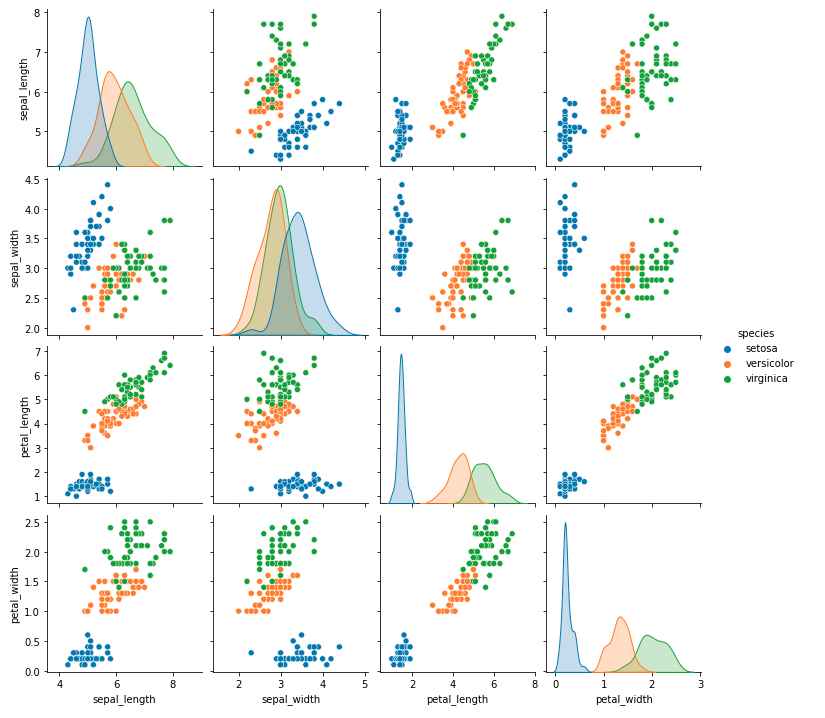
**THƯ VIỆN SEABORN - PYTHON**

**1. Seaborn là gì?**

**Seaborn** là một thư viện Python nổi tiếng dành cho trực quan hoá dữ liệu, được xây dựng trên nền tảng của **Matplotlib**. Thư viện này cung cấp một giao diện thân thiện với người dùng, giúp tạo ra các biểu đồ thống kê đẹp mắt và dễ hiểu. Seaborn được thiết kế để làm việc mượt mà với các DataFrame của **Pandas**, nhờ đó người dùng có thể trực quan hoá và khám phá dữ liệu một cách nhanh chóng và hiệu quả.

Seaborn cung cấp nhiều công cụ mạnh mẽ để trực quan hoá dữ liệu như biểu đồ phân tán, biểu đồ đường, biểu đồ cột, bản đồ nhiệt và nhiều loại biểu đồ khác. Thư viện cũng hỗ trợ các phân tích thống kê nâng cao như hồi quy tuyến tính, biểu đồ phân phối và biểu đồ theo nhóm danh mục.



**2. Tính năng chính của Seaborn**

**- Xây dựng trên nền tảng Matplotlib:** Seaborn được phát triển dựa trên Matplotlib, kế thừa toàn bộ sức mạnh từ thư viện gốc nhưng cung cấp giao diện cấp cao hơn, giúp đơn giản hoá việc tạo biểu đồ thống kê.

**- Tích hợp mượt mà với Pandas và Numpy:** Thư viện Seaborn hoạt động rất tốt với DataFrame của Pandas và mảng Numpy, cho phép người dùng thao tác và trực quan hoá dữ liệu trực tiếp mà không cần chuyển đổi định dạng.

**- Hỗ trợ biểu đồ thống kê đa dạng:** Cung cấp nhiều loại biểu đồ như:

+ Biểu đồ phân tán (scatter plot)

+ Biểu đồ đường (line plot)

+ Biểu đồ cột (bar plot)

+ Biểu đồ phân phối (distribution plot)

+ Bản đồ nhiệt (heatmap)

+ Biểu đồ hộp (boxplot), violin plot, catplot,…

**- Phù hợp cho phân tích dữ liệu thăm dò (EDA) và trình bày thống kê:** Seaborn có sẵn nhiều hàm phục vụ phân tích hồi quy tuyến tính, so sánh phân phối dữ liệu, kiểm tra sự khác biệt giữa các nhóm danh mục,… mà không cần viết nhiều dòng lệnh xử lý phức tạp.

**- Tuỳ chỉnh giao diện dễ dàng:** Cung cấp nhiều chủ đề (themes) và bảng màu (color palettes) mặc định đẹp mắt, chuyên nghiệp, có thể dễ dàng tuỳ chỉnh theo nhu cầu. Điều này giúp tăng tính thẩm mỹ cho biểu đồ mà không tốn nhiều công sức.

**- Hỗ trợ tạo biểu đồ nhiều phần (multi-plot):** Với các công cụ như FaceGrid, PairPlot hoặc JointPlot, người dùng có thể tạo lưới biểu đồ để so sánh nhiều biến hoặc nhóm dữ liệu khác nhau một cách trực quan.

**- Tự động hoá nhiều tác vụ trực quan hoá:** Seaborn giúp tự động xử lý các bước như phân nhóm dữ liệu, thêm nhãn, tính toán trung bình - giúp giảm thiểu mã nguồn cần viết, phù hợp với cả người mới bắt đầu.

**- Tương thích ngược với Matplotlib:** Dù sử dụng Seaborn, người dùng vẫn có thể truy cập và sử dụng các chức năng của Matplotlib để tuỳ chỉnh biểu đồ nâng cao nếu cần.

**3. Ưu điểm và nhược điểm của Seaborn**

**- Ưu điểm của Seaborn:**

+ Dễ dàng xử lý và quản lý dữ liệu có cấu trúc.

+ Trực quan hoá dữ liệu nhanh chóng mà không cần quan tâm đến các chi tiết nội bộ phức tạp.

+ Có thể thay đổi cách biểu diễn dữ liệu bằng cách thay đổi tham số type trong hàm relplot().

+ Hỗ trợ tốt cho việc học tập và trình bày dữ liệu theo cách trực quan, dễ hiểu cho người dùng cuối.

+ Tương thích tốt với Matplotlib và khi cài Seaborn, các thư viện liên quan như Matplotlib cũng được cài đặt thêm.

**- Nhược điểm của Seaborn:**

+ Seaborn không được tích hợp sẵn trong Python, nên cần cài đặt thủ công trước khi sử dụng.

+ Người bước cài đặt, không có hạn chế nghiêm trọng nào khác. Việc cài đặt này chỉ là một bước bổ sung cần thiết trong quá trình làm việc với dữ liệu.

**4. Các hàm vẽ biểu đồ trong Seaborn**

**4.1. Barplot (Biểu đồ cột)**

- Biểu đồ cột trong Seaborn chủ yếu được sử dụng để tổng hợp dữ liệu phân loại theo một phương pháp nhất định, phổ biến là giá trị trung bình (mean), tuy nhiên bạn cũng có thể dùng các phép tính khác. Biểu đồ này có thể được hiểu như một biểu diễn của các nhóm dựa trên hành vi của chúng.

- Để tạo biểu đồ cột, bạn chọn một cột dạng số cho trục y và một cột phân loại cho trục x. Biểu đồ sẽ hiển thị giá trị trung bình (hoặc giá trị được chỉ định) cho mỗi nhóm phân loại.

- Cú pháp: barplot([x, y, hue, data, order, hue\_order, ...])

- Ví dụ:

# set the background style of the plot

sns.set\_style('darkgrid')

# plot the graph using the default estimator mean

sns.barplot(x ='sex', y ='total\_bill', data = df, palette ='plasma')

# or

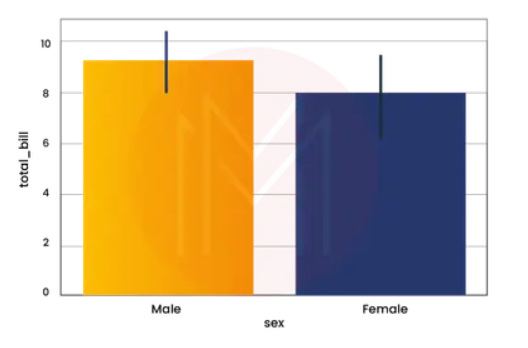
import numpy as np

# change the estimator from mean to standard deviation

sns.barplot(x ='sex', y ='total\_bill', data = df,

palette ='plasma', estimator = np.std)

**Output:**



**Giải thích:** Biểu đồ cho thấy hoá đơn trung bình của nam cao hơn nữ. Điều này giúp rút ra nhận định về sự khác biệt giữa các nhóm giới tính. palette dùng để xác định bảng màu của biểu đồ, còn estimator là hàm thống kê dùng để tính toán giá trị cho từng nhóm.

**4.2. Countplot (Biểu đồ tần suất)**

- Biểu đồ tần suất (count plot) trong Seaborn đếm số lần xuất hiện của từng hạng mục trong một cột phân loại và hiển thị kết quả. Đây là một trong những biểu đồ đơn giản nhất trong Seaborn.

- Cú pháp: countplot([x, y, hue, data, order, …])

- Ví dụ:

sns.countplot(x='sex', data=df)

**Output:**

A graph showing a person and person

AI-generated content may be incorrect.

**Giải thích:** Biểu đồ cho thấy số lượng nam trong dữ liệu lớn hơn nữ. Bạn chỉ cần cung cấp tên cột phân loại thông qua tham số x là có thể tạo biểu đồ.

**4.3. Boxplot (Biểu đồ hộp)**

- Boxplot (còn gọi là biểu đồ hộp và râu) là biểu đồ mô tả phân phối dữ liệu số, được dùng để so sánh giữa các nhóm. Phần hộp thể hiện 40% dữ liệu nằm giữa tứ phân vị thứ nhất và thứ ba, còn “râu” thể hiện các giá trị ngoại vi.

- Cú pháp: boxplot([x, y, hue, data, order, hue\_order, ...])

- Ví dụ:

sns.boxplot(x='day', y='total\_bill', data=df, hue='smoker')

**Output:**

A graph with different colored squares

AI-generated content may be incorrect.

**Giải thích:** Cột x là cột phân loại, y là cột số liệu. hue cho phép phân biệt dữ liệu theo một nhóm phụ khác. Biểu đồ cho thấy người không hút thuốc thường có hoá đơn cao hơn vào thứ Sáu so với người hút thuốc.

**4.4. Violin Plot (Biểu đồ violin)**

- Violin plot tương tự boxplot nhưng có khả năng thể hiện trực quan chi tiết hơn nhờ sử dụng phương pháp ước lượng mật độ (kernel density estimation).

- Cú pháp: violinplot([x, y, hue, data, order, ...])

- Ví dụ:

sns.violinplot(x='day', y='total\_bill', data=df, hue='sex', split=True)

**Output:**

A graph showing the days of the week

AI-generated content may be incorrect.

**Giải thích:** Tham số split=True giúp biểu đồ chia đôi mỗi “violin” theo từng giá trị trong hue. Điều này giúp so sánh trực tiếp giữa hai nhóm dễ dàng hơn.

**4.5. Strip Plot (Biểu đồ điểm rời rạc)**

- Strip plot tạo biểu đồ phân tán đơn giản (scatter plot) dựa tren các giá trị phân loại.

- Cú pháp: stripplot([x, y, hue, data, order, ...])

- Ví dụ:

sns.stripplot(x ='day', y ='total\_bill', data = df, jitter = True, hue ='smoker', dodge = True)

**Output:**

A graph of different colored dots

AI-generated content may be incorrect.

**Giải thích:**

Chúng ta sử dụng tham số jitter để thêm một chút ngẫu nhiên vào vị trí các điểm dữ liệu, vì trong biểu đồ strip plot, các điểm thường chồng lên nhau và rất khó quan sát. Khi có nhiều điểm trùng vị trí, tham số jitter sẽ thêm độ lệch nhẹ (chỉ theo trục phân loại) nhằm làm rõ sự phân bố dữ liệu. Điều này giúp biểu đồ dễ đọc và dễ phân tích hơn.

Tham số hue được dùng để tách biệt các nhóm dữ liệu theo một tiêu chí phân loại phụ, giúp biểu đồ thể hiện nhiều khía cạnh hơn. Nếu bạn muốn tạo các biểu đồ strip plot riêng biệt theo từng nhóm được chỉ định bởi hue, cần sử dụng thêm tham số split=True trong công cụ vẽ của bạn.

**4.6. Swarm Plot (Biểu đồ bầy điểm)**

- Một số người dùng kết hợp đặc điểm của violin plot và strip plot để tạo ra swarm plot trong Seaborn. Biểu đồ này về cơ bản giống strip plot, nhưng các điểm được tự động điều chỉnh vị trí sao cho không chồng lên nhau.

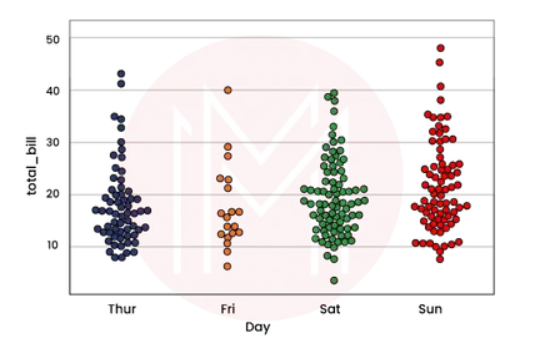
- Lưu ý: Swarm plot không phù hợp với tập dữ liệu lớn do tốn nhiều tài nguyên để sắp xếp các điểm. Thường nên kết hợp với biểu đồ violin để trực quan hiệu quả hơn.

- Cú pháp: swarmplot([x, y, hue, data, order, ...])

- Ví dụ:

sns.swarmplot(x='day', y='total\_bill', data=df)

**Output:**



**4.7. Factorplot (Biểu đồ tổng hợp theo loại)**

- Factorplot là một dạng biểu đồ tổng quát, cho phép bạn chọn loại biểu đồ bằng cách chỉ định tham số kind. Điều này giúp linh hoạt chuyển đổi giữa các kiểu biểu đồ như bar, violin, swarm mà không cần viết lại toàn bộ mã.

- Cú pháp: sns.factorplot([x, y, hue, data, row, col, ...])

- Ví dụ:

sns.factorplot(x='day', y='total\_bill', data=df, kind='bar')

**Output:**

A graph of different colored bars

AI-generated content may be incorrect.

**5. So sánh Seaborn và Matplotlib**

Matplotlib và Seaborn là hai thư viện trực quan hóa dữ liệu phổ biến trong Python, đều hỗ trợ tạo ra các biểu đồ chất lượng cao. Tuy nhiên, chúng khác nhau về mức độ trừu tượng, khả năng tùy chỉnh và mục đích sử dụng.

* **Mức độ trừu tượng:**
  + *Matplotlib* là một thư viện cấp thấp, cho phép kiểm soát chi tiết từng thành phần của biểu đồ như trục, nhãn, màu sắc, kiểu đường... Do đó, rất linh hoạt nhưng cũng phức tạp và đòi hỏi nhiều dòng mã.
  + *Seaborn* là thư viện cấp cao được xây dựng trên Matplotlib. Nó đơn giản hóa quá trình tạo biểu đồ, đặc biệt khi làm việc với dữ liệu dạng bảng như *Pandas DataFrame*, giúp tạo biểu đồ chỉ với vài dòng mã.
* **Khả năng thống kê:**
  + *Seaborn* tích hợp sẵn các hàm thống kê như biểu đồ phân phối, hồi quy tuyến tính, ma trận tương quan..., giúp người dùng dễ dàng kết hợp trực quan hóa với phân tích thống kê.
  + *Matplotlib* không có các công cụ thống kê tích hợp, nên cần kết hợp với các thư viện khác như NumPy, SciPy để thực hiện các phân tích.
* **Phong cách hiển thị mặc định:**
  + *Matplotlib* có giao diện mặc định khá cơ bản. Người dùng thường phải điều chỉnh thủ công để có biểu đồ đẹp mắt.
  + *Seaborn* cung cấp nhiều chủ đề và bảng màu hiện đại, được tối ưu cho việc thể hiện dữ liệu trực quan, giúp biểu đồ đẹp hơn mà ít cần tinh chỉnh.
* **Mục đích sử dụng phù hợp:**
  + Nếu bạn cần tạo biểu đồ tùy biến cao cho báo cáo kỹ thuật, xuất bản khoa học hay thuyết trình, *Matplotlib* là lựa chọn phù hợp.
  + Nếu bạn phân tích dữ liệu thường xuyên và muốn trực quan hóa nhanh, rõ ràng và đẹp mắt, *Seaborn* là công cụ lý tưởng.

Cuối cùng, vì Seaborn được xây dựng trên nền Matplotlib nên bạn hoàn toàn có thể kết hợp cả hai để tận dụng tối đa điểm mạnh của từng thư viện.