**Giới thiệu về Trực quan hóa Dữ liệu trong Python - Cách tạo biểu đồ sử dụng Matplotlib, Pandas và Seaborn**

**Trực quan hóa dữ liệu** là một lĩnh vực cố gắng hiểu dữ liệu bằng cách đặt nó vào một ngữ cảnh trực quan để những mô hình, xu hướng và sự tương quan có thể được phát hiện nếu không thì có thể bị bỏ qua. Python cung cấp nhiều thư viện vẽ đồ thị tuyệt vời đi kèm với rất nhiều tính năng khác nhau. Dù bạn muốn tạo các biểu đồ tương tác, trực tiếp hay tùy chỉnh cao, Python đều có một thư viện tuyệt vời dành cho bạn.

Để có một cái nhìn tổng quan, đây là một số thư viện trực quan hóa phổ biến:

**Matplotlib**: thấp cấp, cung cấp nhiều sự tự do

**Pandas Visualization**: giao diện dễ sử dụng, được xây dựng trên Matplotlib

**Seaborn**: giao diện cao cấp, có kiểu mặc định tuyệt vời

**ggplot**: dựa trên ggplot2 của R, sử dụng Grammar of Graphics

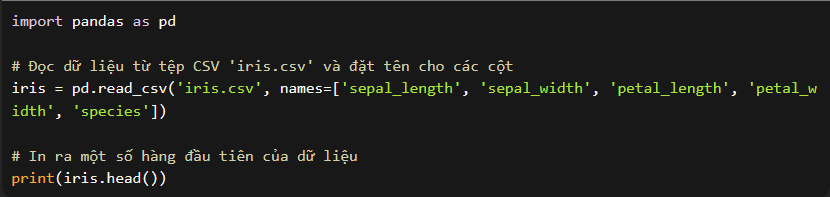
**Plotly**: có thể tạo các biểu đồ tương tác

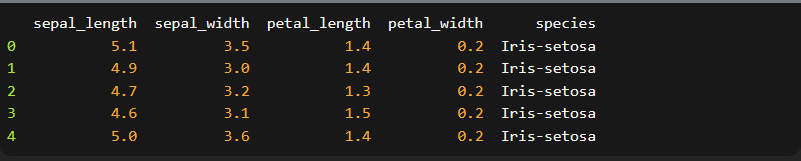
Trong bài viết này, chúng ta sẽ tìm hiểu cách tạo các biểu đồ cơ bản bằng Matplotlib, trực quan hóa Pandas và Seaborn cũng như cách sử dụng một số tính năng cụ thể của mỗi thư viện.

Bài viết này sẽ tập trung vào cú pháp và không tập trung vào việc giải thích các biểu đồ, điều này sẽ được đề cập trong một bài đăng blog khác.

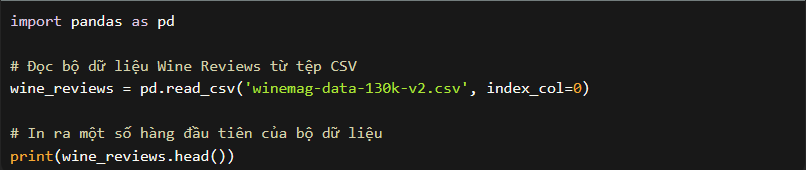
Trong các bài viết tiếp theo, tôi sẽ đi qua các công cụ trực quan hóa tương tác như Plotly, được xây dựng trên D3 và cũng có thể được sử dụng với JavaScript.

Trong bài viết này, chúng ta sẽ sử dụng hai bộ dữ liệu miễn phí. Đó là bộ dữ liệu Iris và bộ dữ liệu Wine Reviews, chúng ta có thể tải về bằng cách sử dụng phương thức read\_csv của pandas.





Hình 2: Đầu bảng dữ liệu Iris





Hình 3: Đầu bảng dữ liệu Wine Reviews

**Matplotlib**

Matplotlib là thư viện trực quan hóa Python phổ biến nhất. Đây là một thư viện cấp thấp với giao diện giống Matlab, cung cấp nhiều sự tự do nhưng đòi hỏi viết nhiều mã hơn.  
Để cài đặt Matplotlib, có thể sử dụng pip hoặc conda:

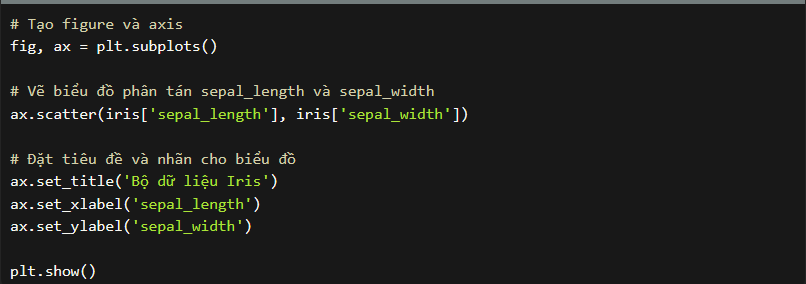
pip install matplotlib

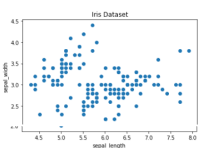
or conda install matplotlib

Matplotlib đặc biệt tốt cho việc tạo các biểu đồ cơ bản như biểu đồ đường, biểu đồ cột, biểu đồ histogram và nhiều biểu đồ khác. Thư viện có thể được nhập bằng cách gõ: import matplotlib.pyplot as plt

**Scatter Plot**

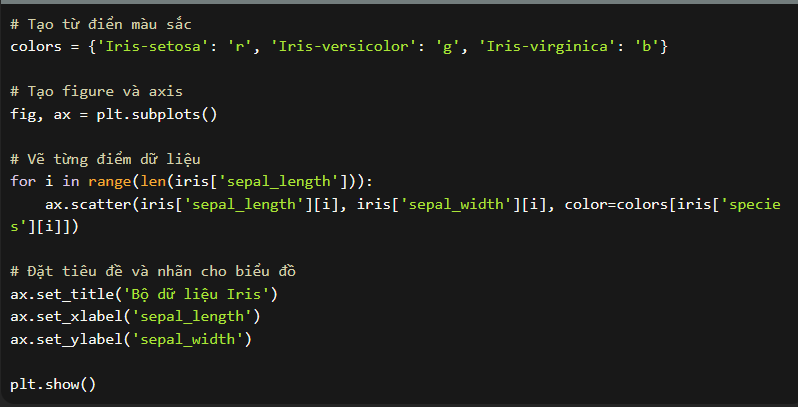
Để tạo một biểu đồ phân tán (scatter plot) trong Matplotlib, chúng ta có thể sử dụng phương thức scatter. Chúng ta cũng sẽ tạo một đối tượng figure và axis bằng cách sử dụng plt.subplots để đặt tiêu đề và nhãn cho biểu đồ của chúng ta.

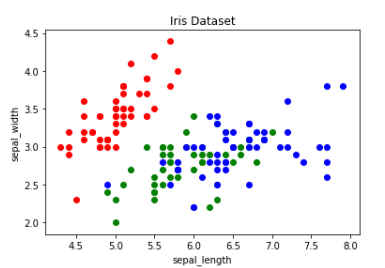




Hình 4: Biểu đồ phân tán trong Matplotlib

Chúng ta cũng có thể tăng tính ý nghĩa của biểu đồ bằng cách tô màu từng điểm dữ liệu theo lớp tương ứng. Điều này có thể được thực hiện bằng cách tạo một từ điển ánh xạ từ lớp sang màu sắc, sau đó sử dụng vòng lặp for để vẽ từng điểm dữ liệu với màu tương ứng.

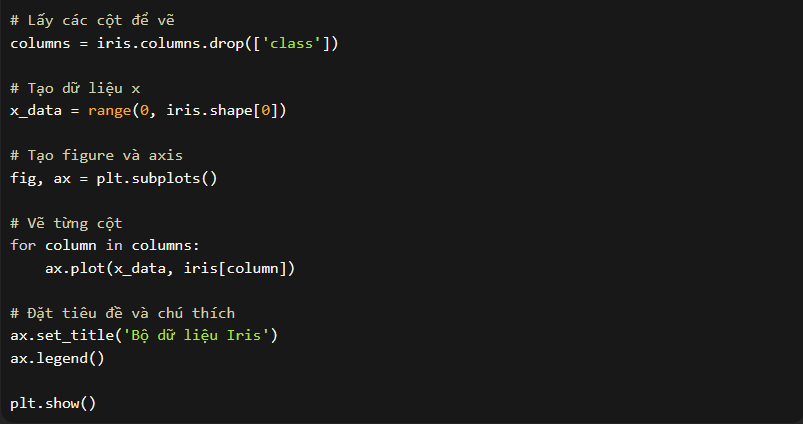


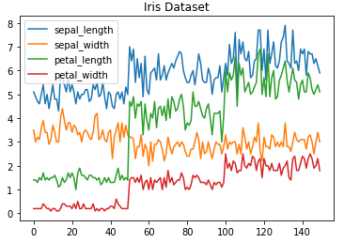


Hình 5: Biểu đồ phân tán được tô màu theo lớp

**Line Chart**

Trong Matplotlib, chúng ta có thể tạo một biểu đồ đường bằng cách gọi phương thức plot. Chúng ta cũng có thể vẽ nhiều cột trong cùng một biểu đồ bằng cách lặp qua các cột chúng ta muốn và vẽ từng cột trên cùng một trục.





Hình 6: Biểu đồ đường

**Histogram**

Trong Matplotlib, chúng ta có thể tạo một biểu đồ histogram bằng cách sử dụng phương thức hist. Nếu chúng ta truyền vào dữ liệu phân loại như cột 'points' từ bộ dữ liệu đánh giá rượu, nó sẽ tự động tính toán tần suất xuất hiện của mỗi lớp.

# Tạo figure và axis

fig, ax = plt.subplots()

# Vẽ biểu đồ histogram

ax.hist(wine\_reviews['points'])

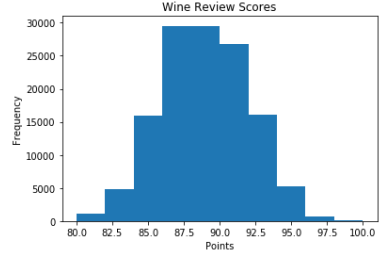
# Đặt tiêu đề và nhãn

ax.set\_title('Điểm Đánh giá Rượu')

ax.set\_xlabel('Điểm')

ax.set\_ylabel('Tần suất')

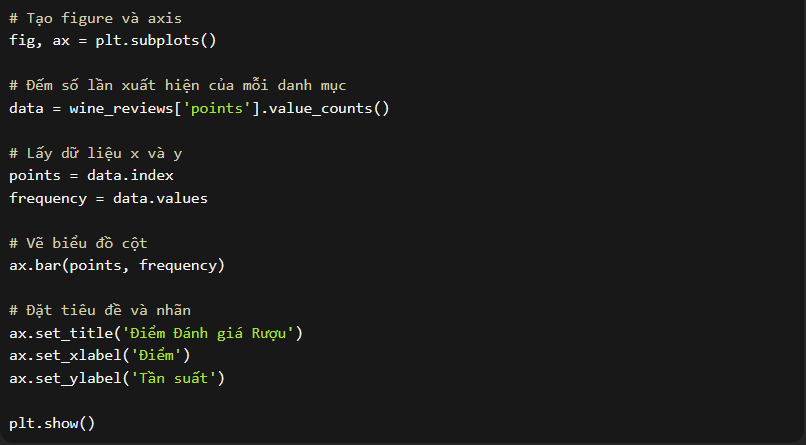
plt.show()

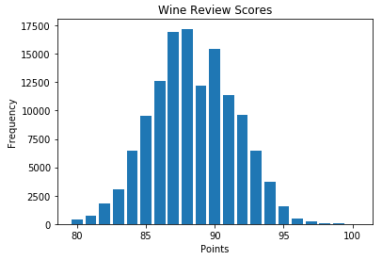
****

Hình 7: Biểu đồ histogram

**Bar Chart**

Chúng ta có thể tạo một biểu đồ cột bằng cách sử dụng phương thức bar. Biểu đồ cột không tự động tính toán tần suất của các danh mục, vì vậy chúng ta sẽ sử dụng hàm value\_counts của pandas để làm điều này. Biểu đồ cột thích hợp cho dữ liệu phân loại không có quá nhiều danh mục (ít hơn 30), vì nếu không, nó có thể trở nên rất lộn xộn.





Hình 8: Biểu đồ cột

**PandasVisualization**

Pandas là một thư viện mã nguồn mở, hiệu suất cao và dễ sử dụng cung cấp các cấu trúc dữ liệu như dataframe và các công cụ phân tích dữ liệu như các công cụ trực quan hóa chúng ta sẽ sử dụng trong bài viết này.

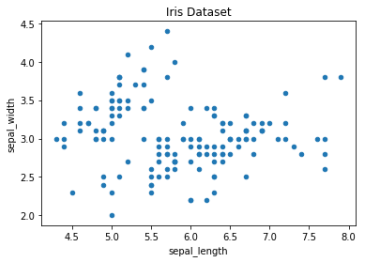
Pandas Visualization giúp tạo biểu đồ dễ dàng từ dataframe và series của pandas. Nó cũng có một API cấp cao hơn so với Matplotlib và do đó chúng ta cần ít mã hơn để đạt được cùng kết quả.

Pandas có thể được cài đặt bằng pip hoặc conda.

**Scatter Plot**

Để tạo một biểu đồ phân tán trong Pandas, chúng ta có thể gọi phương thức <tên\_bộ\_dữ\_liệu>.plot.scatter() và truyền hai đối số vào đó, tên cột x và tên cột y. Tuỳ chọn, chúng ta cũng có thể truyền vào một tiêu đề.





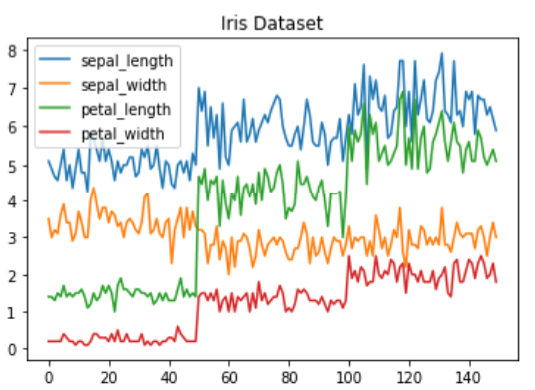
Hình 9: Biểu đồ phân tán

Như bạn có thể thấy trong hình ảnh, Pandas tự động đặt nhãn x và y là tên của các cột tương ứng.

**Line Chart**

Để tạo một biểu đồ đường trong Pandas, chúng ta có thể gọi phương thức <dataframe>.plot.line(). Trong khi ở Matplotlib chúng ta cần lặp qua từng cột mà chúng ta muốn vẽ, trong Pandas chúng ta không cần làm điều này vì nó tự động vẽ tất cả các cột số có sẵn (ít nhất nếu chúng ta không chỉ định cột/cột cụ thể).





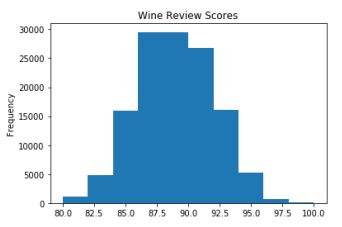
Hình 10: Biểu đồ đường

Nếu chúng ta có nhiều hơn một đặc trưng, Pandas sẽ tự động tạo một chú thích cho chúng ta, như có thể thấy trong hình ảnh trên.

**Histogram**

Trong Pandas, chúng ta có thể tạo một biểu đồ histogram bằng phương thức plot.hist. Không có đối số bắt buộc, nhưng chúng ta có thể truyền một số đối số tùy chọn như kích thước bin.

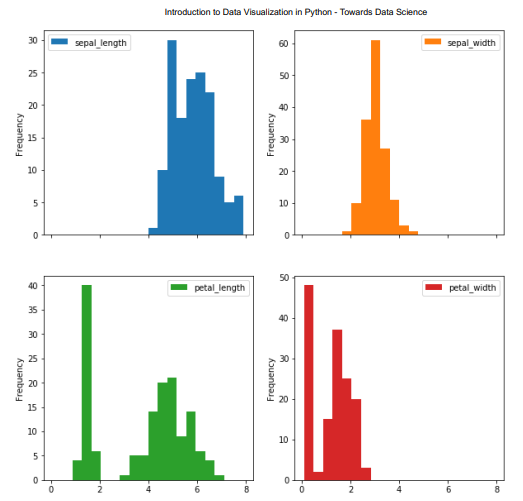




Hình 11: Biểu đồ histogram

Ngoài ra, việc tạo nhiều biểu đồ histogram cũng rất dễ dàng.





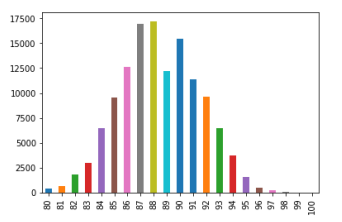
Hình 12: Nhiều biểu đồ histogram

Đối số subplots chỉ định rằng chúng ta muốn một biểu đồ riêng cho mỗi đặc trưng và layout xác định số biểu đồ trên mỗi hàng và cột.

**Bar Chart**

Để vẽ một biểu đồ cột, chúng ta có thể sử dụng phương thức plot.bar(), nhưng trước khi làm điều này, chúng ta cần lấy dữ liệu của mình. Đầu tiên, chúng ta sẽ đếm số lần xuất hiện bằng cách sử dụng phương thức value\_counts() và sau đó sắp xếp các lần xuất hiện từ nhỏ đến lớn bằng phương thức sort\_index().

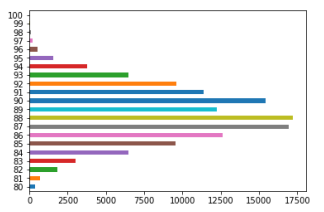




Hình 13: Biểu đồ cột dọc

Ngoài ra, việc tạo một biểu đồ cột ngang cũng rất đơn giản bằng cách sử dụng phương thức plot.barh().

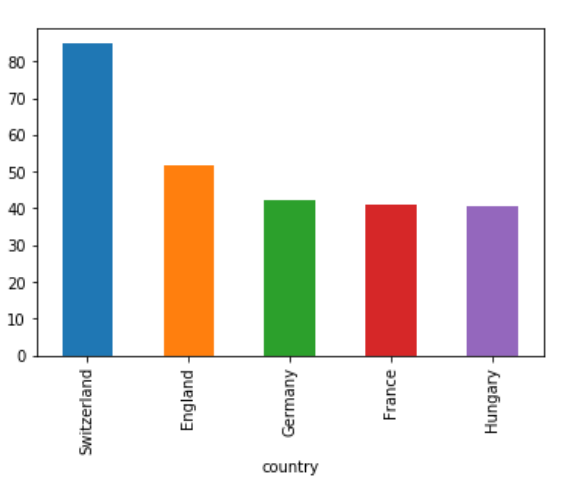
wine\_reviews['points'].value\_counts().sort\_index().plot.barh()



Hình 14: Biểu đồ cột ngang

Chúng ta cũng có thể vẽ các dữ liệu khác ngoài số lần xuất hiện.





Hình 15: Các quốc gia có rượu đắt nhất (trung bình)

Trong ví dụ trên, chúng ta nhóm dữ liệu theo quốc gia, sau đó lấy giá trung bình của rượu, sắp xếp nó và vẽ các quốc gia có giá trung bình rượu cao nhất.

**Seaborn**

Là một thư viện trực quan hóa dữ liệu Python dựa trên Matplotlib. Nó cung cấp một giao diện cấp cao để tạo ra các đồ thị hấp dẫn.

Seaborn có rất nhiều tính năng hữu ích. Bạn có thể tạo ra các đồ thị chỉ trong một dòng lệnh mà trong Matplotlib bạn có thể mất nhiều chục dòng lệnh. Thiết kế mặc định của Seaborn rất tuyệt vời và nó cũng cung cấp một giao diện thuận tiện để làm việc với các dataframe của Pandas.

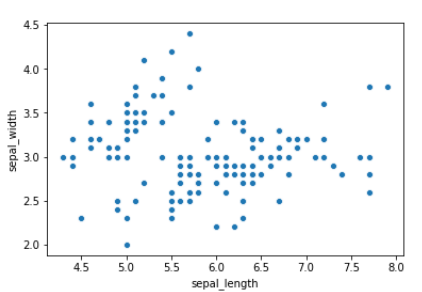
Để import Seaborn, bạn có thể gõ:



**Scatter plot**

Chúng ta có thể sử dụng phương thức .scatterplot để tạo một biểu đồ phân tán, và giống như trong Pandas, chúng ta cần truyền vào tên cột dữ liệu x và y, nhưng lần này chúng ta cũng cần truyền dữ liệu làm đối số bổ sung vì chúng ta không gọi hàm trực tiếp trên dữ liệu như chúng ta đã làm trong Pandas.

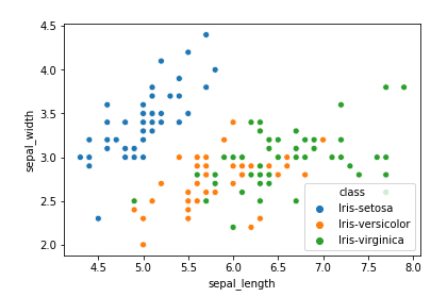




Hình 16: Biểu đồ phân tán

Chúng ta cũng có thể làm nổi bật các điểm theo lớp bằng cách sử dụng đối số hue, điều này dễ dàng hơn nhiều so với Matplotlib.

sns.scatterplot(x='sepal\_length', y='sepal\_width', hue='class', data=iris)

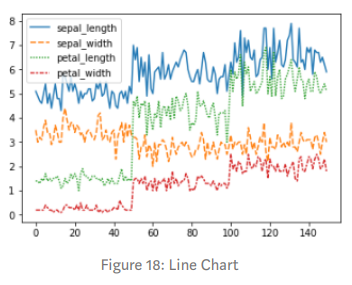


Hình 17: Biểu đồ phân tán được tô màu theo lớp

**Line chart**

Để tạo biểu đồ đường trong Seaborn, bạn có thể sử dụng phương thức sns.lineplot. Đối số bắt buộc chính là dữ liệu, trong trường hợp này là bốn cột số từ tập dữ liệu Iris. Hoặc bạn cũng có thể sử dụng phương thức sns.kdeplot, nó sẽ làm mờ các đường cong và tạo ra một biểu đồ sạch hơn nếu tập dữ liệu của bạn có nhiều giá trị ngoại lệ.



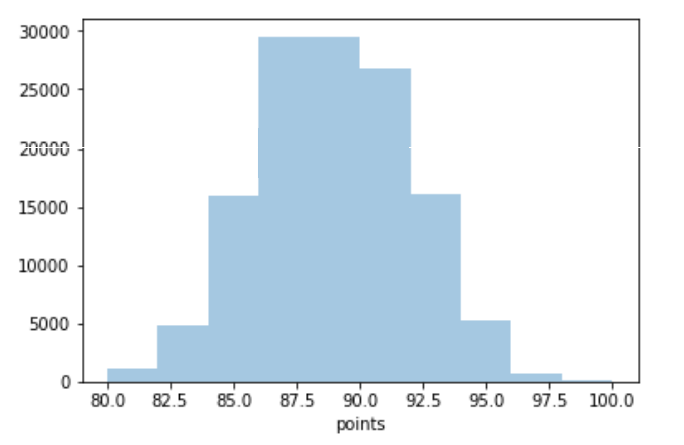


Hình 18: Biểu đồ đường

**Histogram**

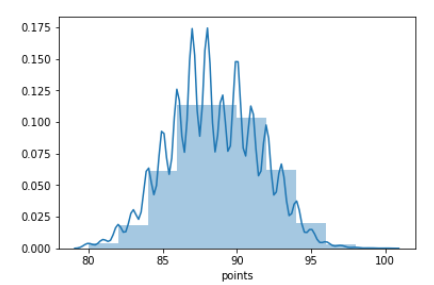
Để tạo một biểu đồ histogram trong Seaborn, bạn có thể sử dụng phương thức sns.distplot. Bạn cần truyền cột mà bạn muốn vẽ và nó sẽ tự động tính toán số lần xuất hiện. Bạn cũng có thể chỉ định số bin và xem có muốn bao gồm ước lượng mật độ nhân khẩu học Gaussian trong biểu đồ hay không.





Hình 19: Biểu đồ histogram



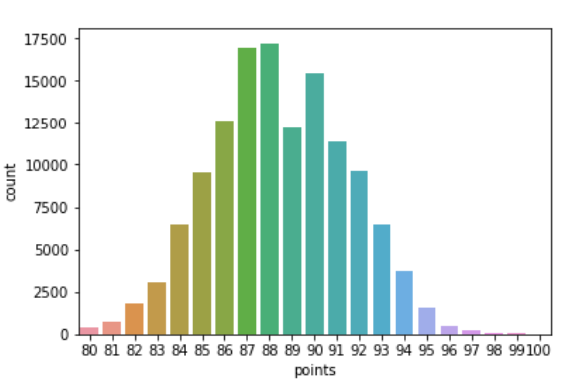


Hình 20: Biểu đồ histogram với ước lượng mật độ nhân khẩu học Gaussian

**Bar chart**

Trong Seaborn, bạn có thể tạo một biểu đồ cột bằng cách sử dụng phương thức sns.countplot và truyền vào dữ liệu.





Hình 21: Biểu đồ cột

**Other graphs**  
Bây giờ bạn đã hiểu cú pháp cơ bản của Matplotlib, Trực quan hóa Pandas và Seaborn, tôi muốn chỉ cho bạn một số loại đồ thị khác mà hữu ích để trích xuất thông tin.

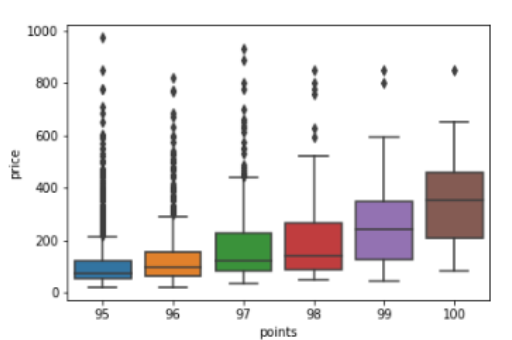
Đối với hầu hết các loại đồ thị này, Seaborn là thư viện được chọn vì giao diện cấp cao của nó cho phép tạo ra các đồ thị đẹp chỉ trong vài dòng mã.

Biểu đồ hộp (Box plots)  
Biểu đồ hộp là một phương pháp đồ họa để hiển thị bản tóm tắt năm số. Chúng ta có thể tạo biểu đồ hộp bằng cách sử dụng phương thức sns.boxplot của Seaborn và truyền dữ liệu cũng như tên cột x và y.

**Box plots**

Biểu đồ hộp là một phương pháp đồ họa để hiển thị bản tóm tắt năm số. Chúng ta có thể tạo biểu đồ hộp bằng cách sử dụng phương thức sns.boxplot của Seaborn và truyền dữ liệu cũng như tên cột x và y.





Hình 22: Biểu đồ hộp

Biểu đồ hộp, giống như biểu đồ cột, rất phù hợp cho dữ liệu có chỉ một vài loại, nhưng có thể trở nên lộn xộn nhanh chóng nếu có quá nhiều loại dữ liệu.

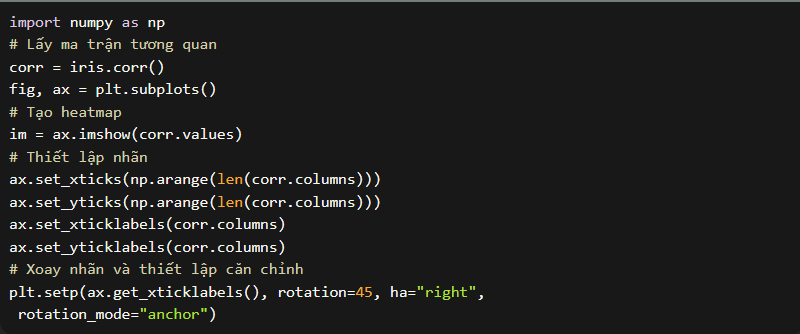
**Heatmap**

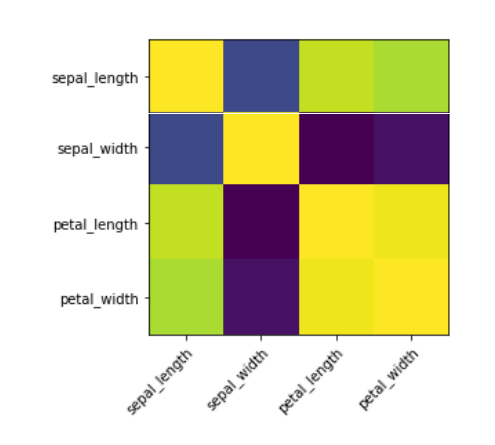
Một Heatmap là một biểu đồ biểu diễn đồ họa của dữ liệu, trong đó các giá trị cá nhân trong ma trận được biểu diễn dưới dạng màu sắc. Heatmap rất phù hợp để khám phá mối tương quan giữa các đặc trưng trong tập dữ liệu.

Để lấy được sự tương quan giữa các đặc trưng trong một tập dữ liệu, chúng ta có thể gọi phương thức <tập dữ liệu>.corr(), đây là một phương thức dataframe của Pandas. Điều này sẽ cho chúng ta ma trận tương quan.

Bây giờ chúng ta có thể sử dụng Matplotlib hoặc Seaborn để tạo heatmap.

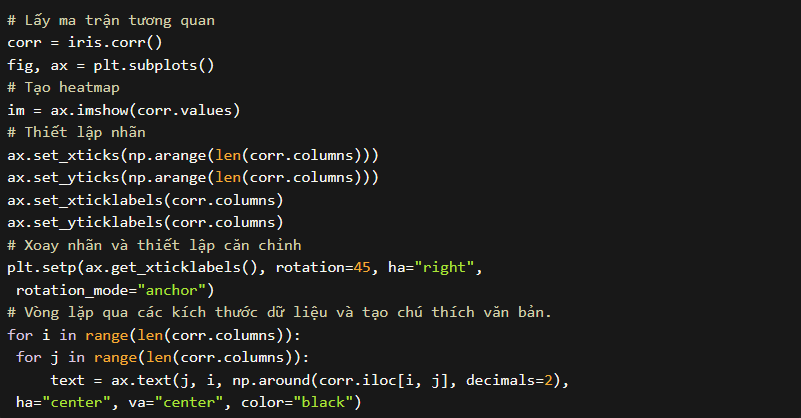
Sử dụng Matplotlib:

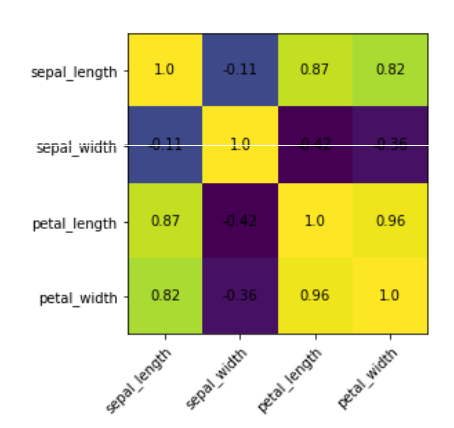




Hình 23: Heatmap không có chú thích

Để thêm chú thích vào heatmap, chúng ta cần thêm hai vòng lặp for:

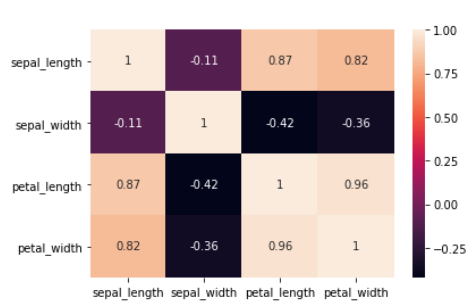




Hình 24: Heatmap với chú thích

Seaborn làm cho việc tạo heatmap và thêm chú thích dễ dàng hơn nhiều:





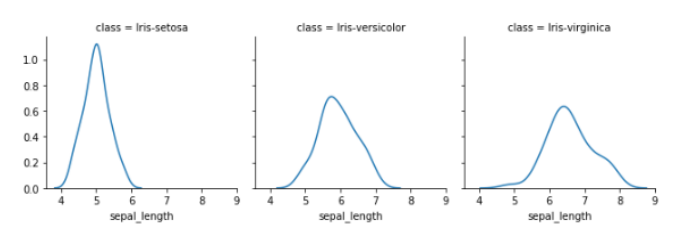
Hình 25: Heatmap với chú thích

**Faceting**

Là việc chia các biến dữ liệu thành nhiều subplot và kết hợp các subplot đó vào một hình ảnh duy nhất. Faceting thực sự hữu ích nếu bạn muốn khám phá nhanh tập dữ liệu của mình.

Để sử dụng một loại faceting trong Seaborn, chúng ta có thể sử dụng FacetGrid. Trước tiên, chúng ta cần xác định FacetGrid và truyền vào dữ liệu cũng như một hàng hoặc cột, sẽ được sử dụng để chia tách dữ liệu. Sau đó, chúng ta cần gọi hàm map trên đối tượng FacetGrid của chúng ta và xác định loại đồ thị chúng ta muốn sử dụng, cũng như cột chúng ta muốn vẽ đồ thị.





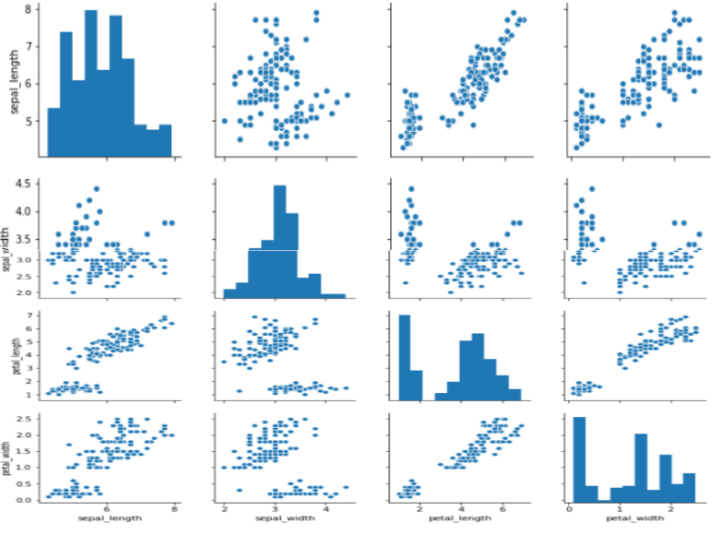
Hình 26: Facet-plot

Bạn có thể tạo ra các đồ thị lớn hơn và phức tạp hơn so với ví dụ trên. Bạn có thể tìm thấy một số ví dụ tại đây.

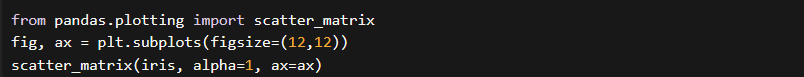
**Pairplot**

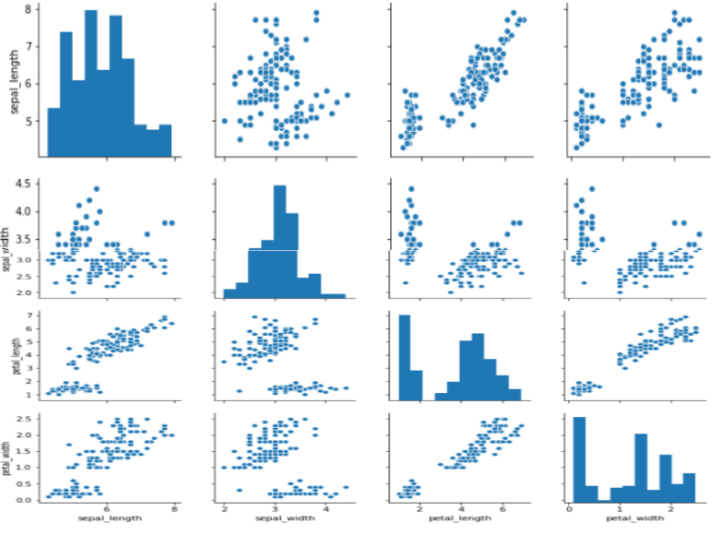
Cuối cùng, tôi sẽ chỉ cho bạn cách sử dụng pairplot của Seaborn và scatter\_matrix của Pandas, cho phép bạn vẽ một lưới các mối quan hệ cặp trong tập dữ liệu.





Hình 27: Pairplot





Hình 28: Scatter matrix

Như bạn có thể thấy trong các hình ảnh trên, các kỹ thuật này luôn vẽ hai đặc trưng với nhau. Đường chéo của đồ thị được điền bằng các biểu đồ histogram và các biểu đồ khác là biểu đồ phân tán.

**Kết luận**

Trực quan hóa dữ liệu là một lĩnh vực quan trọng để hiểu dữ liệu bằng cách đặt nó trong ngữ cảnh hình ảnh để phát hiện ra các mẫu, xu hướng và mối tương quan mà không thể nhận thấy nếu không có trực quan hóa. Python cung cấp nhiều thư viện đồ họa mạnh mẽ với nhiều tính năng khác nhau. Trong bài viết này, chúng ta đã tìm hiểu về Matplotlib, trực quan hóa dữ liệu của Pandas và Seaborn.

Nếu bạn thích bài viết này, hãy đăng ký kênh Youtube của tôi và theo dõi tôi trên mạng xã hội.

Mã nguồn được đề cập trong bài viết này có sẵn trên Github Repository.

Nếu bạn có bất kỳ câu hỏi, gợi ý hoặc đánh giá, bạn có thể liên hệ với tôi qua Twitter hoặc phần bình luận.