**BÁO CÁO NHỎ DUY**

1. **Vẽ biểu đồ miền** 
   1. **Phần code vẽ biểu đồ miền**

# Vẽ biểu đồ miền

def ve\_bieu\_do\_mien():

for widget in khung\_4.winfo\_children():

widget.destroy()

if 'Topic' in du\_lieu.columns:

data = du\_lieu['Topic'].value\_counts()

fig, ax = plt.subplots(figsize=(8, 5))

ax.fill\_between(data.index, data.values, color="skyblue", alpha=0.4)

ax.plot(data.index, data.values, color="Slateblue", alpha=0.6, linewidth=2)

ax.set\_title("Biểu đồ miền topic")

ax.set\_xlabel("Topic")

ax.set\_ylabel("Số lượng")

ax.tick\_params(axis='x', rotation=45)

canvas = FigureCanvasTkAgg(fig, master=khung\_4)

canvas.draw()

canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

tk.Button(khung\_4, text="Phóng to", command=lambda: hien\_thi\_full\_screen(fig)).pack(pady=10)

* 1. **Giải thích chi tiết**
     1. **def ve\_bieu\_do\_mien():**

Hàm này dùng để vẽ một biểu đồ miền (area plot) thể hiện tần suất xuất hiện của các giá trị trong cột Topic từ dữ liệu du\_lieu.

* + 1. **Xóa nội dung khung hiển thị:**
* for widget in khung\_4.winfo\_children():

widget.destroy()

**Mục đích:** Xóa tất cả các widget con (nếu có) bên trong khung khung\_4 trước khi vẽ biểu đồ mới. Điều này đảm bảo rằng khung được làm sạch trước khi hiển thị biểu đồ.

**Cách hoạt động:**

khung\_4.winfo\_children(): Lấy danh sách tất cả các widget con của khung\_4.

widget.destroy(): Xóa từng widget trong danh sách đó.

* + 1. **Kiểm tra dữ liệu:**
* if 'Topic' in du\_lieu.columns:

data = du\_lieu['Topic'].value\_counts()

**Mục đích:** Kiểm tra xem cột Topic có tồn tại trong DataFrame du\_lieu hay không. Nếu tồn tại:

Đếm số lần xuất hiện của mỗi giá trị duy nhất trong cột Topic.

Lưu kết quả vào biến data (một Series với các giá trị là tần suất xuất hiện).

**Cách hoạt động:**

du\_lieu['Topic']: Truy cập vào cột Topic của DataFrame.

value\_counts(): Đếm số lần xuất hiện của mỗi giá trị duy nhất trong cột.

* + 1. **Tạo biểu đồ:**
* fig, ax = plt.subplots(figsize=(8, 5))

**Mục đích:** Tạo một biểu đồ mới với kích thước cụ thể (8x5 inch).

**Cách thức hoạt động:**

plt.subplots(): Tạo một figure (fig) và một trục (ax) để vẽ biểu đồ.

figsize=(8, 5): Xác định kích thước của biểu đồ

* ax.fill\_between(data.index, data.values, color="skyblue", alpha=0.4)

**Mục đích:**Tô màu miền phía dưới đường biểu đồ.

**Cách hoạt động:**

data.index: Trục hoành (x) là các giá trị duy nhất trong cột Topic.

data.values: Trục tung (y) là tần suất xuất hiện của các giá trị đó.

color="skyblue": Chọn màu xanh nhạt cho miền.

alpha=0.4: Đặt độ mờ (trong suốt) của miền.

* ax.plot(data.index, data.values, color="Slateblue", alpha=0.6, linewidth=2)

**Mục đích:**Vẽ đường biểu đồ trên miền vừa tô màu.

**Cách hoạt động:**

color="Slateblue": Chọn màu xanh đậm cho đường.

alpha=0.6: Đặt độ mờ của đường.

linewidth=2: Đặt độ dày của đường là 2.

* + 1. **Thiết lập biểu đồ:**
* ax.set\_title("Biểu đồ miền topic")

ax.set\_xlabel("Topic")

ax.set\_ylabel("Số lượng")

**Mục đích:** Đặt tiêu đề và nhãn cho các trục của biểu đồ.

**Cách hoạt động:**

set\_title: Đặt tiêu đề cho biểu đồ.

set\_xlabel: Đặt nhãn cho trục x (trục hoành).

set\_ylabel: Đặt nhãn cho trục y (trục tung).

* ax.tick\_params(axis='x', rotation=45)

**Mục đích:**Xoay các nhãn trên trục x để dễ đọc.

**Cách hoạt động:**

axis='x': Áp dụng thay đổi cho trục x.

rotation=45: Xoay các nhãn 45 độ.

* + 1. **Hiển thị biểu đồ trên giao diện:**
* canvas = FigureCanvasTkAgg(fig, master=khung\_4)

canvas.draw()

canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

**Mục đích:** Chuyển biểu đồ được tạo bằng Matplotlib sang widget Tkinter và hiển thị trong khung\_4.

**Cách hoạt động:**

FigureCanvasTkAgg(fig, master=khung\_4): Chuyển fig (biểu đồ) thành một widget tương thích với Tkinter.

canvas.draw(): Vẽ biểu đồ.

canvas.get\_tk\_widget(): Lấy widget Tkinter chứa biểu đồ.

pack(fill=tk.BOTH, expand=True): Hiển thị widget, lấp đầy không gian trong khung khung\_4.

* + 1. **Thêm nút phóng to:**
* tk.Button(khung\_4, text="Phóng to", command=lambda: hien\_thi\_full\_screen(fig)).pack(pady=10)

**Mục đích:** Thêm một nút "Phóng to" vào khung\_4 để hiển thị biểu đồ ở chế độ toàn màn hình.

**Cách hoạt động:**

tk.Button: Tạo một nút.

text="Phóng to": Đặt nhãn cho nút.

command=lambda: hien\_thi\_full\_screen(fig) : Gọi hàm hien\_thi\_full\_screen(fig) khi bấm nút.

pack(pady=10): Hiển thị nút với khoảng cách dọc (padding) là 10 pixel.

1. **Cập nhật dữ liệu được chọn**
   1. **Phần code cập nhật dữ liệu được chọn**

# Cập nhật dữ liệu được chọn

def cap\_nhat\_du\_lieu():

selected\_items = tree.selection()

if not selected\_items:

messagebox.showwarning("Cảnh báo", "Chọn mục để cập nhật.")

return

updated\_data = [entry.get() for entry in danh\_sach\_o\_nhap]

for item in selected\_items:

index = tree.index(item)

du\_lieu.iloc[index] = updated\_data

du\_lieu.to\_csv(FILE\_NAME, sep=";", index=False, encoding="utf-8")

hien\_thi\_du\_lieu()

* 1. **Giải thích chi tiết**
     1. **def cap\_nhat\_du\_lieu():**

Hàm cap\_nhat\_du\_lieu() thực hiện cập nhật dữ liệu trong giao diện và lưu thay đổi vào file dữ liệu.

* + 1. **Lấy các mục được chọn từ tree**

Lấy danh sách các hàng được chọn từ bảng dữ liệu trong giao diện (Treeview).

* selected\_items = tree.selection()

**Mục đích:** tree.selection(): Lấy danh sách các mục (hàng) đang được chọn trong widget dạng Treeview (thường dùng để hiển thị dữ liệu dạng bảng trong Tkinter).

**Kết quả:** selected\_items chứa danh sách các ID của các mục được chọn trong Treeview. Nếu không có mục nào được chọn, danh sách này sẽ rỗng.

* + 1. **Kiểm tra xem có mục nào được chọn hay không**

Nếu không có mục nào được chọn, hiển thị cảnh báo và thoát hàm.

* if not selected\_items:

messagebox.showwarning("Cảnh báo", "Chọn mục để cập nhật.")

Return

**Mục đích :**

Nếu danh sách selected\_items rỗng (không có mục nào được chọn), hiển thị một hộp thoại cảnh báo (messagebox.showwarning) với nội dung "Chọn mục để cập nhật".

Sau đó, hàm thoát khỏi bằng câu lệnh return.

Đảm bảo rằng người dùng đã chọn ít nhất một mục để tiến hành cập nhật.

* + 1. **Lấy dữ liệu mới từ các ô nhập liệu**

Thu thập dữ liệu người dùng nhập vào từ các ô nhập liệu.

* updated\_data = [entry.get() for entry in danh\_sach\_o\_nhap]

**Mục đích:**

danh\_sach\_o\_nhap: Một danh sách chứa các widget nhập liệu (ví dụ: Entry của Tkinter) mà người dùng dùng để nhập dữ liệu mới.

entry.get(): Lấy giá trị hiện tại (chuỗi) từ mỗi ô nhập liệu.

Dùng list comprehension để tạo danh sách updated\_data chứa dữ liệu mới mà người dùng đã nhập.

**Kết quả:** updated\_data chứa dữ liệu mới từ các ô nhập liệu, dưới dạng danh sách các chuỗi.

* + 1. **Cập nhật dữ liệu trong du\_lieu**

Xác định vị trí của các hàng được chọn trong DataFrame du\_lieu.

Ghi đè dữ liệu của các hàng này bằng dữ liệu mới từ ô nhập liệu.

* for item in selected\_items:

index = tree.index(item)

du\_lieu.iloc[index] = updated\_data

**Mục đích:**

Lặp qua từng mục (item) trong selected\_items (các hàng được chọn trong Treeview).

Lấy chỉ số hàng trong dữ liệu: tree.index(item): Trả về vị trí (chỉ số) của mục được chọn trong Treeview.

Cập nhật dữ liệu: du\_lieu.iloc[index] = updated\_data: Thay thế hàng thứ index trong DataFrame du\_lieu bằng danh sách updated\_data.

**Lý do:** Đồng bộ hóa dữ liệu từ giao diện với DataFrame du\_lieu (đại diện dữ liệu gốc trong chương trình).

* + 1. **Lưu dữ liệu vào file CSV**

Ghi nội dung DataFrame du\_lieu đã cập nhật vào file CSV.

* du\_lieu.to\_csv(FILE\_NAME, sep=";", index=False, encoding="utf-8")

**Mục đích:** Ghi nội dung của DataFrame du\_lieu vào file CSV được chỉ định bởi FILE\_NAME.

**Các tham số:**

sep=";": Sử dụng ký tự ; làm dấu phân cách giữa các cột.

index=False: Không ghi chỉ số hàng vào file.

encoding="utf-8": Đảm bảo mã hóa file theo ch

uẩn UTF-8 để hỗ trợ tiếng Việt và các ký tự đặc biệt.

**Lý do:** Lưu các thay đổi dữ liệu để đảm bảo chúng được lưu trữ lâu dài.

* + 1. **Hiển thị lại dữ liệu trong giao diện**

Làm mới nội dung trong bảng giao diện để phản ánh các thay đổi mới.

* hien\_thi\_du\_lieu()

**Mục đích:** Gọi hàm hien\_thi\_du\_lieu(), có nhiệm vụ cập nhật lại nội dung của widget Treeview để phản ánh các thay đổi mới từ DataFrame du\_lieu.

**Lý do:** Sau khi cập nhật dữ liệu, cần làm mới giao diện để hiển thị nội dung mới.

1. **Tạo giao diện ứng dụng**
   1. **Đoạn code tạo giao diện ứng dụng**

ung\_dung = tk.Tk()

ung\_dung.title("Quản lý dữ liệu với Đồ thị")

ung\_dung.state('zoomed')

* 1. **Giải thích chi tiết** 
     1. ****Khởi tạo cửa sổ chính của ứng dụng:****
* ung\_dung = tk.Tk()

tk.Tk() là một đối tượng chính của thư viện Tkinter, đại diện cho cửa sổ chính của ứng dụng.

Đoạn mã này tạo ra cửa sổ chính của ứng dụng và lưu nó vào biến ung\_dung.

Sau khi đối tượng này được tạo, chúng ta có thể thực hiện các thao tác với cửa sổ chính như thay đổi kích thước, tiêu đề, màu sắc, thêm các widget (các thành phần giao diện) vào trong cửa sổ này.

* + 1. **Thiết lập tiêu đề cửa sổ:**
* ung\_dung.title("Quản lý dữ liệu với Đồ thị")

ung\_dung.title(...) dùng để đặt tiêu đề cho cửa sổ chính của ứng dụng.

"Quản lý dữ liệu với Đồ thị" là tên cửa sổ mà người dùng sẽ thấy ở thanh tiêu đề của cửa sổ ứng dụng.

* + 1. **Đặt trạng thái cửa sổ là 'zoomed':**
* ung\_dung.state('zoomed')

ung\_dung.state('zoomed') dùng để đặt trạng thái của cửa sổ chính là zoomed. Điều này có nghĩa là cửa sổ sẽ được mở rộng tối đa ra toàn màn hình khi người dùng khởi chạy ứng dụng.

state() là phương thức của Tkinter dùng để thiết lập trạng thái cửa sổ.

Tham số 'zoomed' có tác dụng làm cửa sổ mở rộng hết màn hình (tương tự như khi người dùng nhấn nút "phóng to" trên cửa sổ).

Nếu muốn cửa sổ có kích thước mặc định, có thể thay 'zoomed' bằng 'normal'.

1. **Tạo giao diện ứng dụng**
   1. **Đoạn code tạo giao diện ứng dụng**

# Tạo giao diện ứng dụng

khung\_1 = tk.Frame(ung\_dung)

khung\_1.grid(row=0, column=0, columnspan=3, sticky="nsew", pady=10)

khung\_2 = tk.LabelFrame(ung\_dung, text="Nhập dữ liệu", padx=10, pady=10)

khung\_2.grid(row=1, column=0, sticky="nsew", padx=10, pady=10)

khung\_4 = tk.LabelFrame(ung\_dung, text="Đồ thị", padx=10, pady=10)

khung\_4.grid(row=1, column=1, columnspan=2, sticky="nsew", padx=5, pady=5)

khung\_3 = tk.Frame(ung\_dung)

khung\_3.grid(row=3, column=0, columnspan=3, sticky="nsew", padx=10, pady=10)

* 1. **Giải thích chi tiết**

Đoạn mã này tạo ra 4 khung (frame) trong cửa sổ chính (ung\_dung), mỗi khung có nhiệm vụ riêng:

**khung\_1:** Là khung đầu tiên, có thể chứa các widget liên quan đến tiêu đề hoặc thông tin.

**khung\_2:** Là khung chứa phần "Nhập dữ liệu", có thể chứa các widget nhập liệu.

**khung\_4:** Là khung chứa phần "Đồ thị", có thể chứa các biểu đồ hoặc hình ảnh.

**khung\_3:** Là khung cuối cùng, có thể chứa các thành phần khác hoặc là một phần của giao diện điều khiển.

Tất cả các khung này được sắp xếp bằng grid layout, nơi mỗi khung được chỉ định vị trí (hàng, cột), độ rộng (columnspan), và các thuộc tính bố trí khác (như kéo dài, padding).