Tool by Nguyễn Vạn Phúc Huy

Matplotlib

📊 1. Vẽ Đồ Thị Cơ Bản

- `import matplotlib.pyplot as plt` : Nhập thư viện pyplot.
- `plt.style.use("style_name")` : Chọn kiểu hiển thị (vd: `"ggplot"`, `"seaborn-v0_8"`, `"dark_background"`).
- `plt.plot(y)` : Vẽ đồ thị với `y` là tung độ, hoành độ là chỉ số (index).
- `plt.plot(x, y)` : Vẽ đồ thị với `x` là hoành độ, `y` là tung độ.
- `plt.show()` : Hiển thị đồ thị vừa vẽ.

📐 2. Tùy Chỉnh Trục & Tiêu Đề

- `plt.xlim(min, max)` : Đặt giới hạn cho trục hoành.
- `plt.ylim(min, max)` : Đặt giới hạn cho trục tung (có thể dùng `None` cho một giới hạn).
- `plt.title("Tiêu đề")` : Thêm tiêu đề cho đồ thị.
- `plt.xlabel("Nhãn trục X")` : Thêm nhãn cho trục hoành.
- `plt.ylabel("Nhãn trục Y")` : Thêm nhãn cho trục tung.
- `plt.grid(True)` : Hiển thị lưới (hoặc `False` để ẩn).
- `plt.xticks(ticks_list, labels_list, rotation=45)` : Tùy chỉnh các vạch chia (ticks) và nhãn (labels) trên trục X, có thể xoay nhãn.
- `plt.yticks(ticks_list, labels_list)` : Tương tự cho trục Y.

✨ 3. Định Dạng Đường, Màu & Marker

- `plt.plot(x, y, format_string)` : Định dạng nhanh (ít dùng).
  - *Ví dụ:* `'b--'` (blue dashed line - xanh dương, nét đứt).
  - *Ví dụ:* `'r:'` (red dotted line - đỏ, nét chấm).
  - *Ví dụ:* `'g^'` (green triangles - xanh lá, marker hình tam giác).
- **Cách dùng rõ ràng (nên dùng):**
  - `color='blue'` (hoặc `'#0000FF'`, `'b'`): Chỉ định màu sắc.
  - `linestyle='--'` (hoặc `ls`): Kiểu đường (`'-'`, `'--'`, `':'`, `'-.'`).
  - `linewidth=2.5` (hoặc `lw`): Độ dày của đường.
  - `marker='o'`: Kiểu điểm đánh dấu (`'o'`, `'^'`, `'s'`, `'+'`, `'x'`).
  - `markersize=8` (hoặc `ms`): Kích thước của marker.
- *Ví dụ đầy đủ:* `plt.plot(x, y, color='green', linestyle=':', marker='o', lw=2)`

🖼️ 4. Kích Thước & Đồ Thị Con (Cách Cơ Bản)

- `plt.figure(figsize=(width, height))` : Tạo một Figure mới với kích thước tùy chỉnh (đơn vị inch). (Lưu ý: `plt.subplots()` ở mục 8 là cách làm gọn hơn).
- `plt.subplot(rows, cols, index)` : Chia figure thành lưới `rows` x `cols` và chọn đồ thị con thứ `index` (bắt đầu từ 1) để vẽ lệnh `plt.plot()` tiếp theo.
  - *Ví dụ:* `plt.subplot(2, 2, 1)` (Lưới 2x2, chọn ô thứ 1 - trái trên).
  - *Ví dụ:* `plt.subplot(2, 1, 2)` (Lưới 2x1, chọn ô thứ 2 - dưới).
- `plt.tight_layout()` : Tự động căn chỉnh các đồ thị con cho vừa vặn, tránh chồng chéo.

📝 5. Thêm Chữ, Chú Thích & Nhãn

- `plt.text(x, y, "Nội dung")` : Thêm chữ tại tọa độ (x, y).
  - *Tham số tùy chọn:* `fontsize`, `color`, `horizontalalignment='center'`, `rotation`.
- `plt.plot(..., label="Tên đường")` : Đặt nhãn cho một đường vẽ (dùng cho `legend`).
- `plt.legend(loc="vị_trí")` : Hiển thị bảng chú giải các nhãn.
  - *Ví dụ:* `loc="lower right"`, `loc="best"` (tự động chọn vị trí tốt nhất).
- `plt.annotate("Chú thích", xy=(x_point, y_point), xytext=(x_text, y_text), arrowprops=dict(facecolor='black'))` : Thêm chú thích có mũi tên chỉ từ chữ (`xytext`) đến điểm (`xy`).

📈 6. Các Loại Đồ Thị Khác

- `plt.hist(data, bins=n)` : Vẽ biểu đồ tần suất (histogram) với `n` khoảng (bins).
- `plt.scatter(x, y)` : Vẽ biểu đồ phân tán (scatter plot).
  - *Tham số tùy chọn:* `s` (kích thước điểm), `c` (màu sắc, có thể là 1 mảng giá trị), `alpha` (độ trong suốt).
- `plt.boxplot(data)` : Vẽ biểu đồ hộp (boxplot), `data` có thể là list các mảng.
- `plt.bar(categories, values)` : Vẽ biểu đồ cột đứng.
- `plt.barh(categories, values)` : Vẽ biểu đồ cột ngang.
- `plt.pie(sizes, labels=labels_list, autopct='%1.1f%%')` : Vẽ biểu đồ tròn.
- `plt.imshow(matrix_data, cmap='viridis')` : Hiển thị ma trận 2D (heatmap, ảnh).
- `plt.colorbar()` : Thêm thanh màu (thường dùng với `imshow` hoặc `scatter` khi `c` là một mảng giá trị).

💾 7. Lưu Hình Vẽ

- `plt.savefig("tên_file.png")` : Lưu đồ thị thành file ảnh.
  - *Tham số tùy chọn:*
  - `dpi=300` : (Dots Per Inch) Tăng độ phân giải cho ảnh (chuẩn in ấn).
  - `transparent=True` : Nền trong suốt.
  - `bbox_inches='tight'` : Cắt bỏ các khoảng trắng thừa xung quanh hình.
- *Ví dụ:* `plt.savefig("my_figure.png", dpi=300, bbox_inches='tight')`

👑 8. API Hướng Đối Tượng (OO) - Cách Dùng Chuẩn

Đây là cách dùng linh hoạt và được khuyên dùng, đặc biệt khi xử lý nhiều đồ thị con. Thay vì `plt.plot()`, `plt.title()`, ta gọi phương thức từ đối tượng `ax` (Axes).

- **Figure (`fig`)**: Toàn bộ cửa sổ chứa hình.
- **Axes (`ax`)**: Từng ô đồ thị con (nơi thực sự vẽ).

A. Một đồ thị con (Cách dùng chuẩn)

```
# Tạo 1 Figure và 1 Axes
fig, ax = plt.subplots(figsize=(10, 6))

# Vẽ trên ax
ax.plot(x, y, label="Dữ liệu", color='red')

# Tùy chỉnh trên ax (thường có 'set_' ở đầu)
ax.set_title("Tiêu đề (cách OO)")
ax.set_xlabel("Nhãn X")
ax.set_ylabel("Nhãn Y")
ax.set_xlim(0, 10)
ax.grid(True)
ax.legend()

plt.show() # Vẫn dùng plt.show() để hiển thị
```

Pandas

📥 1. Nhập & Xuất Dữ Liệu

- `pd.read_csv('file.csv')`: Đọc file CSV.
  - *Tham số:* `sep=','`, `header=0`, `names=['a', 'b']`, `index_col='col_name'`
- `pd.read_excel('file.xlsx', sheet_name='Sheet1')`: Đọc file Excel.
- `pd.read_json('file.json')`: Đọc file JSON.
- `pd.read_sql(query, connection)`: Đọc từ database SQL.
- `pd.DataFrame(data_dict)`: Tạo DataFrame từ dictionary (ví dụ: `{'col1': [1,2], 'col2': [3,4]}`).
- `df.to_csv('file.csv', index=False)`: Lưu DataFrame thành file CSV (không lưu cột index).
- `df.to_excel('file.xlsx', sheet_name='Data')`: Lưu DataFrame thành file Excel.
- `df.to_json('file.json', orient='records')`: Lưu DataFrame thành file JSON.

🔍 2. Khám Phá Nhanh (Inspection)

- `df.head(n)`: Xem `n` dòng đầu tiên (mặc định là 5).
- `df.tail(n)`: Xem `n` dòng cuối cùng (mặc định là 5).
- `df.sample(n)`: Lấy ngẫu nhiên `n` dòng.
- `df.shape`: Trả về (số dòng, số cột).
- `df.info()`: Thông tin tổng quan (kiểu dữ liệu, số lượng non-null, bộ nhớ).
- `df.describe()`: Thống kê mô tả cho các cột số (count, mean, std, min, max, quartiles).
- `df.describe(include='object')`: Thống kê mô tả cho các cột object/categorical (count, unique, top, freq).
- `df.columns`: Hiển thị danh sách các tên cột.
- `df.index`: Hiển thị thông tin về index (hàng).
- `df.dtypes`: Hiển thị kiểu dữ liệu (dtype) của mỗi cột.
- `df['col'].value_counts()`: Đếm các giá trị duy nhất trong một cột (Series).

🎯 3. Chọn & Lọc Dữ Liệu (Selection & Filtering)

- **Chọn Cột:**
  - `df['col']` hoặc `df.col`: Chọn một cột (trả về Series).
  - `df[['col1', 'col2']]`: Chọn nhiều cột (trả về DataFrame).
- **Chọn Dòng (theo nhãn/vị trí):**
  - `df.loc[label]`: Chọn một dòng theo nhãn index (ví dụ: `df.loc[0]`).
  - `df.loc['label1':'label5']`: Chọn một khoảng dòng theo nhãn (slice, bao gồm cả điểm cuối).
  - `df.iloc[vị_trí]`: Chọn một dòng theo vị trí số nguyên (ví dụ: `df.iloc[0]`).
  - `df.iloc[0:5]`: Chọn 5 dòng đầu tiên (slice, không bao gồm điểm cuối, giống Python list).
- **Chọn Dòng & Cột:**
  - `df.loc[row_labels, col_labels]`: Chọn theo nhãn.
    - *Ví dụ:* `df.loc[0:5, ['name', 'age']]`
  - `df.iloc[row_positions, col_positions]`: Chọn theo vị trí.
    - *Ví dụ:* `df.iloc[0:5, [0, 2]]`
- **Lọc theo điều kiện (Boolean Indexing):**
  - `df[df['age'] > 30]`: Lọc các dòng thỏa mãn điều kiện.
  - `df[df['country'].isin(['USA', 'Canada'])]`: Lọc các dòng có giá trị nằm trong một danh sách.
  - `df[(df['age'] > 30) & (df['gender'] == 'Male')]`: Lọc với nhiều điều kiện (dùng `&` (AND), `|` (OR), `~` (NOT)).
  - `df.query('age > 30 and gender == "Male"')`: Cách lọc khác dùng chuỗi truy vấn (query string).

🔧 4. Làm Sạch & Biến Đổi Dữ Liệu (Cleaning & Transformation)

- **Xử lý NaN (Missing Values):**
  - `df.isna().sum()`: Đếm số lượng giá trị NaN trong mỗi cột.
  - `df.dropna()`: Xóa tất cả các dòng có chứa NaN.
  - `df.dropna(axis=1)`: Xóa tất cả các cột có chứa NaN.
  - `df.fillna(value)`: Điền giá trị (ví dụ: `0`, `df['col'].mean()`) cho các ô bị thiếu (NaN).
- **Thay Đổi Dữ Liệu:**
  - `df.rename(columns={'old_name': 'new_name'})`: Đổi tên cột.
  - `df.replace(old_value, new_value)`: Thay thế giá trị.
  - `df['col'].astype('new_type')`: Thay đổi kiểu dữ liệu của cột (ví dụ: `'int'`, `'float'`, `'category'`).
- **Xóa Dòng/Cột:**
  - `df.drop(columns=['col_name'])`: Xóa cột.
  - `df.drop(labels=[index_label])`: Xóa dòng theo index.
- **Duplicates:**
  - `df.duplicated()`: Kiểm tra các dòng trùng lặp (trả về Boolean Series).
  - `df.drop_duplicates(subset=['col_name'], keep='first')`: Xóa các dòng trùng lặp (chỉ giữ lại dòng đầu tiên).
- **Áp dụng hàm (Apply/Map):**
  - `df['col'].map(function_or_dict)`: Áp dụng hàm hoặc dictionary cho từng phần tử của một Cột (Series).
  - `df.apply(function, axis=0)`: Áp dụng hàm lên từng cột (trả về 1 giá trị cho mỗi cột).
  - `df.apply(function, axis=1)`: Áp dụng hàm lên từng dòng (trả về 1 giá trị cho mỗi dòng).
    - *Ví dụ (lambda):* `df['new_col'] = df['old_col'].apply(lambda x: x * 2)`

🔀 5. Sắp Xếp & Sửa Index

- `df.sort_values(by='col_name', ascending=True)`: Sắp xếp DataFrame theo giá trị của một cột.
- `df.sort_values(by=['col1', 'col2'])`: Sắp xếp theo nhiều cột.
- `df.sort_index()`: Sắp xếp DataFrame theo index.
- `df.reset_index()`: Chuyển index hiện tại thành một cột mới và dùng index số nguyên mặc định.
- `df.set_index('col_name')`: Đặt một cột làm index của DataFrame.

📊 6. Thống Kê & Groupby

- **Thống Kê Cơ Bản:**
  - `.mean()`, `.std()`, `.median()`: Trung bình, độ lệch chuẩn, trung vị.
  - `.sum()`, `.count()`: Tổng, số lượng (không tính NaN).
  - `.max()`, `.min()`: Giá trị lớn nhất, nhỏ nhất.
  - `.nunique()`: Đếm số lượng giá trị duy nhất.
  - `.corr()`: Tính ma trận tương quan giữa các cột số.
  - `.quantile(x)`: Tính tứ phân vị ở x%
- **Grouping:**
  - `df.groupby('col')`: Nhóm dữ liệu theo các giá trị của một cột.
  - `df.groupby(['col1', 'col2'])`: Nhóm theo nhiều cột.
  - `df.groupby('col').sum()`: Tính tổng cho mỗi nhóm.
  - `df.groupby('col')['data_col'].mean()`: Tính trung bình của cột 'data_col' cho mỗi nhóm.
  - `df.groupby('col').agg(func_dict)`: Áp dụng nhiều hàm thống kê cùng lúc.
    - *Ví dụ:* `df.groupby('group').agg({'data1': 'mean', 'data2': 'sum'})`

🔗 7. Hợp Nhất & Kết Nối (Merge & Concat)

- `pd.concat([df1, df2])` : Nối các DataFrame theo chiều dọc (stack rows, `axis=0`).
- `pd.concat([df1, df2], axis=1)` : Nối các DataFrame theo chiều ngang (ghép cột, `axis=1`).
- `pd.merge(df_left, df_right, on='key_col')` : Kết nối (join) hai DataFrame dựa trên một cột chung (key).
  - *Tham số `how` (cách join):*
    - `how='inner'` (mặc định): Chỉ giữ lại các hàng có key chung ở cả hai bảng.
    - `how='left'` : Giữ tất cả các hàng của bảng trái, điền NaN nếu không có key ở bảng phải.
    - `how='right'` : Giữ tất cả các hàng của bảng phải.
    - `how='outer'` : Giữ tất cả các hàng của cả hai bảng.

🔄 8. Pivot Tables

- `df.pivot_table(values='data_col', index='row_col', columns='col_col', aggfunc='mean')` :
  - Tạo một bảng tổng hợp (giống Excel Pivot Table).
  - `values` : Cột dữ liệu để tính toán.
  - `index` : Cột để dùng làm các hàng của bảng pivot.
  - `columns` : Cột để dùng làm các cột của bảng pivot.
  - `aggfunc` : Hàm thống kê (ví dụ: `'mean'`, `'sum'`, `'count'`).

⏳ 9. Dữ Liệu Thời Gian (Time Series)

- `pd.to_datetime(df['date_col'])` : Chuyển đổi cột sang kiểu dữ liệu datetime.
- *Sau khi đặt cột datetime làm index (dùng `df.set_index('date_col')`):*
- `df['2025']` : Chọn tất cả dữ liệu trong năm 2025.
- `df['2025-10']` : Chọn tất cả dữ liệu trong tháng 10/2025.
- `df.resample('M').mean()` : Tính trung bình dữ liệu theo mỗi tháng ('M'), ('D' - ngày, 'W' - tuần, 'Q' - quý).

📈 10. Trực Quan Hóa (Tích hợp)

- `df.plot(kind='line')` : Vẽ biểu đồ đường (mặc định).
- `df.plot(kind='bar')` : Vẽ biểu đồ cột đứng.
- `df.plot(kind='barh')` : Vẽ biểu đồ cột ngang.
- `df.plot(kind='scatter', x='col1', y='col2')` : Vẽ biểu đồ phân tán.
- `df['col'].hist(bins=20)` : Vẽ biểu đồ histogram cho một cột.
- `df.boxplot()` : Vẽ biểu đồ hộp cho các cột số.

Phân Phối Xác Suất (Scipy Stats)

1. 📖 Giới Thiệu

- **Xác Suất**: Đại lượng dùng để đo khả năng xảy ra của một sự kiện (event), có giá trị từ 0 đến 1.
- **Phân Phối Xác Suất (PPXS)**: Thể hiện xác suất xuất hiện của các giá trị của một biến ngẫu nhiên. Giúp hiểu quy luật của dữ liệu, dự đoán giá trị, so sánh dữ liệu với các PPXS đã biết.
- **Biến Ngẫu Nhiên (BNN):** Các đặc trưng dữ liệu được giả định là biến ngẫu nhiên.
  - **Rời rạc (Discrete):** Nhận các giá trị đếm được (vd: số lần tung xúc xắc, số câu đúng, số khách hàng). Mô tả bằng **Hàm khối xác suất (PMF)**.
  - **Liên tục (Continuous):** Nhận các giá trị trong một khoảng (vd: chiều cao, nhiệt độ, thời gian). Mô tả bằng **Hàm mật độ xác suất (PDF)**.

---

2. 📚 Thư Viện `scipy.stats`

- Thư viện chính trong Python để làm việc với các PPXS.
- Import thư viện chung và các phân phối cụ thể:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import binom    # Nhị thức
from scipy.stats import poisson  # Poisson
from scipy.stats import uniform  # Đều
from scipy.stats import norm     # Chuẩn
from scipy.stats import expon    # Mũ
from scipy.stats import t        # Student
from scipy.stats import chi2     # Chi-bình phương
```

- Một số phân phối thông dụng:

Scipy Package	Distribution Name	Tên Phân Phối	Ký hiệu
`binom`	Binomial Distribution	Phân Phối Nhị Thức	$B(n, p)$
`poisson`	Poisson Distribution	Phân Phối Poisson	$P(\lambda)$
`uniform`	Uniform Distribution	Phân Phối Đều	$U(a, b)$
`norm`	Normal/Gauss Distribution	Phân Phối Chuẩn	$N(\mu, \sigma^2)$
`expon`	Exponential Distribution	Phân Phối Mũ	$Exp(\lambda)$
`t`	Student t-Distribution	Phân Phối Student	$t(df)$
`chi2`	Chi-Square Distribution	Phân Phối Chi Bình Phương	$\chi^2(df)$

---

3. ⚙️ Các Hàm & Tham Số Chung

A. Các Hàm Tính Xác Suất & Lấy Mẫu (Function Details)

- `<dist>.rvs(...)` : Tạo mẫu ngẫu nhiên (Random Variates).
  - `size` : Số lượng mẫu cần tạo.
  - Các tham số đặc trưng của phân phối (vd: `n`, `p` cho `binom`).
- `<dist>.pmf(k, ...)` : **Hàm khối xác suất** (Probability Mass Function).
  - Dùng cho BNN **Rời rạc**.
  - Tính xác suất $P(X = k)$. (Hàm độ lớn f(x) cho phân phối rời rạc).
- `<dist>.pdf(x, ...)` : **Hàm mật độ xác suất** (Probability Density Function).
  - Dùng cho BNN **Liên tục**.
  - $f(x)$ (Hàm mật độ f(x) cho phân phối liên tục. Lưu ý: $f(x) \geq 0$ và tích phân toàn miền bằng 1, $f(x)$ không phải là xác suất tại điểm x).
- `<dist>.cdf(x, ...)` : **Hàm phân phối tích lũy** (Cumulative Distribution Function).
  - Tính xác suất $P(X \leq x)$ (Hàm F(x)).
- `<dist>.ppf(q, ...)` : **Hàm điểm phần trăm** (Percent Point Function).
  - Hàm ngược của CDF (Inverse of CDF), còn gọi là Quantile Function.
  - Tìm x sao cho $P(X \leq x) = q$ (với q là xác suất). (Tìm điểm phân vị).
- `<dist>.sf(x, ...)` : **Hàm sống sót** (Survival Function).
  - Tính xác suất $P(X > x)$ (bằng `1 - cdf(x)`). (Hàm Survival).
- `<dist>.isf(q, ...)` : **Hàm ngược của hàm sống sót** (Inverse Survival Function).
  - Tìm x sao cho $P(X > x) = q$. (Hàm ngược của hàm Survival).

B. Các Tham Số Chung (loc, scale)

Hầu hết các phân phối liên tục đều có 2 tham số chuẩn hóa:

- `loc` : (Location) Dịch chuyển vị trí phân phối.

- Với `norm`, `loc` chính là trung bình μ.
- Với `uniform`, `loc` là điểm bắt đầu a.

- `scale` : (Scale) Co dãn phân phối (quy định độ rộng).
  - Với `norm`, `scale` chính là độ lệch chuẩn σ.
  - Với `uniform`, `scale` là độ rộng $b - a$.
  - Với `expon`, `scale` là $1/\lambda$ (nghịch đảo của rate λ, tức là giá trị trung bình).

C. Các Hàm Thống Kê

- `<dist>.mean(...)` : Trả về kỳ vọng (trung bình) của phân phối.
- `<dist>.median(...)` : Trả về trung vị.
- `<dist>.var(...)` : Trả về phương sai (σ^2).
- `<dist>.std(...)` : Trả về độ lệch chuẩn (σ).

---

4. Phân Phối Nhị Thức (`binom`) - Rời Rạc

- **Ký hiệu**: $B(n, p)$
  - `n` : Số lần thực hiện phép thử Bernoulli độc lập.
  - `p` : Xác suất thành công trong một lần thử.
- **Biến ngẫu nhiên X**: Số lần thành công trong `n` phép thử.
- **Ví dụ**: Một bài thi trắc nghiệm gồm 10 câu, mỗi câu có 4 lựa chọn (p=0.25). Bạn đánh lụi tất cả. X là số câu trả lời đúng. $X \sim B(10, 0.25)$.

```
from scipy.stats import binom
import matplotlib.pyplot as plt
import numpy as np

n, p = 10, 0.25

# 1. Phát sinh mẫu ngẫu nhiên (rvs)
# Phát sinh 1 giá trị (1 bạn làm bài)
mau_1 = binom.rvs(n, p)
print(f"Một kết quả ngẫu nhiên: {mau_1}")

# Phát sinh 1000 giá trị (1000 bạn làm bài)
mau_1000 = binom.rvs(n, p, size=1000)

# Vẽ histogram của mẫu
plt.figure(figsize=(10, 5))
dorongcot = 0.4
offset = dorongcot / 2
plt.hist(mau_1000, bins=np.arange(n + 2) - offset, width=dorongcot, density=True, label='Histogram từ mẫu RVS')
plt.xticks(range(n + 1))
plt.xlabel('Số câu đúng (k)')
plt.ylabel('Tần suất tương đối / Mật độ')
plt.title(f'Histogram Phân phối Nhị thức (n={n}, p={p}) từ 1000 mẫu')

# 2. Tính và vẽ hàm PMF (xác suất lý thuyết P(X=k))
k_values = np.arange(0, n + 1)
pmf_values = binom.pmf(k_values, n, p)
plt.stem(k_values, pmf_values, linefmt='r-', markerfmt='ro', basefmt=' ', label='PMF Lý thuyết')
plt.legend()
plt.show()

# 3. Tính và vẽ hàm CDF (xác suất tích lũy P(X <= k))
plt.figure(figsize=(10, 5))
cdf_values = binom.cdf(k_values, n, p)
plt.plot(k_values, cdf_values, marker='o', drawstyle='steps-post', label='CDF') # steps-post cho đồ thị bậc thang
plt.title(f'Hàm Phân phối Tích lũy (CDF) Nhị thức (n={n}, p={p})')
plt.xlabel('Số câu đúng (k)')
plt.ylabel('P(X <= k)')
plt.xticks(k_values)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.legend()
plt.show()

# 4. Tính toán xác suất cụ thể
# P(X = 3) - Xác suất đúng chính xác 3 câu
p_eq_3 = binom.pmf(3, n, p)
print(f"P(X = 3) = {p_eq_3:.4f}")
# P(X <= 2) - Xác suất đúng tối đa 2 câu
p_le_2 = binom.cdf(2, n, p)
print(f"P(X <= 2) = {p_le_2:.4f}")
# P(X > 5) - Xác suất đúng nhiều hơn 5 câu
p_gt_5 = binom.sf(5, n, p) # <=> 1 - binom.cdf(5, n, p)
print(f"P(X > 5) = {p_gt_5:.4f}")
# Tìm k nhỏ nhất sao cho P(X <= k) >= 0.9 (Phân vị thứ 90)
k_q90 = binom.ppf(0.9, n, p)
print(f"Phân vị thứ 90 (k sao cho P(X<=k)>=0.9): {k_q90}")
```

---

5. Phân Phối Poisson (`poisson`) - Rời Rạc

- **Ký hiệu**: $P(\lambda)$
  - `lambda` (λ): Trung bình số lần xảy ra sự kiện trong một khoảng (thời gian, không gian). Trong `scipy` tham số này tên là `mu`.
- **Biến ngẫu nhiên X**: Số lần sự kiện xảy ra.
- **Ví dụ**: Số khách hàng đến siêu thị trong 1 giờ (trung bình $\lambda = 20$).

```
from scipy.stats import poisson

# Trung bình 20 khách/giờ
mu_lambda = 20

# P(X = 20) - Xác suất có đúng 20 khách
p_eq_20 = poisson.pmf(20, mu=mu_lambda)
print(f"P(X = 20) = {p_eq_20:.4f}")
# P(X <= 15) - Xác suất có tối đa 15 khách
p_le_15 = poisson.cdf(15, mu=mu_lambda)
print(f"P(X <= 15) = {p_le_15:.4f}")
# P(X > 30) - Xác suất có hơn 30 khách
p_gt_30 = poisson.sf(30, mu=mu_lambda) # <=> 1 - poisson.cdf(30, mu=mu_lambda)
print(f"P(X > 30) = {p_gt_30:.4f}")

# Vẽ PMF và CDF tương tự như Binomial
k_values_poi = np.arange(0, 41) # Chọn khoảng giá trị hợp lý quanh lambda
pmf_poi = poisson.pmf(k_values_poi, mu=mu_lambda)
cdf_poi = poisson.cdf(k_values_poi, mu=mu_lambda)

fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].stem(k_values_poi, pmf_poi, linefmt='b-', markerfmt='bo', basefmt=' ')
axes[0].set(xlabel='Số khách (k)', title=f'Poisson PMF ($\lambda$={mu_lambda})')
axes[1].plot(k_values_poi, cdf_poi, marker='.', drawstyle='steps-post')
axes[1].set(xlabel='Số khách (k)', title=f'Poisson CDF ($\lambda$={mu_lambda})')
plt.tight_layout()
plt.show()
```

---

6. Phân Phối Chuẩn (`norm`) - Liên Tục

- **Ký hiệu**: $N(\mu, \sigma^2)$
  - `loc` (μ): Kỳ vọng (trung bình).
  - `scale` (σ): Độ lệch chuẩn. (Lưu ý: tham số là σ, không phải σ^2).
- **Biến ngẫu nhiên X**: Biến liên tục, đồ thị hình chuông đối xứng qua μ.
- **Ví dụ**: Chiều cao nam thanh niên trưởng thành ($\mu = 170$ cm, $\sigma = 3$ cm). $X \sim N(170, 3^2)$.

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 170, 3

# 1. Phát sinh mẫu ngẫu nhiên (rvs) và vẽ histogram
plt.figure(figsize=(10, 5))
mau_10k = norm.rvs(loc=mu, scale=sigma, size=10000)
plt.hist(mau_10k, bins=100, density=True, alpha=0.6, color='g', label='Histogram từ mẫu RVS')

# 2. Tính và vẽ hàm PDF (hàm mật độ xác suất) lý thuyết
x_values = np.linspace(mu - 4*sigma, mu + 4*sigma, 200) # Khoảng giá trị quanh mu (~99.99%)
pdf_values = norm.pdf(x_values, loc=mu, scale=sigma)
plt.plot(x_values, pdf_values, color='red', lw=2, label=f'PDF Lý thuyết $N({mu}, {sigma}^2)$')
plt.title(f'Phân phối Chuẩn (Normal) - PDF và Histogram Mẫu')
plt.xlabel('Chiều cao (cm)')
plt.ylabel('Mật độ')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

# 3. Tính toán với Phân phối Chuẩn Tắc (Z ~ N(0, 1))
print("\n--- Tính toán với Phân phối Chuẩn Tắc Z ~ N(0, 1) ---")
# P(Z <= -1)
p_le_neg1 = norm.cdf(-1) # Mặc định loc=0, scale=1
print(f"P(Z <= -1) = {p_le_neg1:.5f}")
# P(Z > 1) = 1 - P(Z <= 1)
p_gt_1 = 1 - norm.cdf(1)
# Hoặc dùng hàm sf: P(Z > 1)
p_gt_1_sf = norm.sf(1)
print(f"P(Z > 1) = {p_gt_1:.5f} (dùng 1-cdf) = {p_gt_1_sf:.5f} (dùng sf)")
# P(-1 <= Z <= 1)
p_between = norm.cdf(1) - norm.cdf(-1)
print(f"P(-1 <= Z <= 1) = {p_between:.5f}")

# 4. Tìm điểm phân vị (Quantile) với Phân phối Chuẩn Tắc
print("\n--- Tìm điểm phân vị (Quantile) với Z ~ N(0, 1) ---")
# Tìm z sao cho P(Z <= z) = 0.02275
z_val_ppf = norm.ppf(0.02275)
print(f"z sao cho P(Z <= z) = 0.02275 là: {z_val_ppf:.3f}") # Gần bằng -2
# Tìm z sao cho P(Z > z) = 0.02275
z_val_isf = norm.isf(0.02275)
print(f"z sao cho P(Z > z) = 0.02275 là: {z_val_isf:.3f}") # Gần bằng 2

# 5. Tính toán với phân phối chuẩn bất kỳ (ví dụ X ~ N(16, 4^2))
print("\n--- Tính toán với Phân phối Chuẩn X ~ N(16, 4^2) ---")
mu_x, sigma_x = 16, 4
# P(X <= 8)
p_le_8 = norm.cdf(8, loc=mu_x, scale=sigma_x)
print(f"P(X <= 8) = {p_le_8:.5f}")
# Tìm x sao cho P(X <= x) = 0.02275
x_val_ppf = norm.ppf(0.02275, loc=mu_x, scale=sigma_x)
print(f"x sao cho P(X <= x) = 0.02275 là: {x_val_ppf:.3f}") # = 16 + (-2)*4 = 8

# Vẽ minh họa P(-1 <= Z <= 1)
plt.figure(figsize=(8, 4))
x_z = np.linspace(-4, 4, 200)
pdf_z = norm.pdf(x_z)
plt.plot(x_z, pdf_z, label='PDF của Z ~ N(0,1)')
x_fill = np.linspace(-1, 1, 100)
plt.fill_between(x_fill, norm.pdf(x_fill), color='blue', alpha=0.35, label='P(-1 <= Z <= 1)')
plt.title('Minh họa diện tích dưới đường cong PDF chuẩn tắc')
plt.text(-0.4, 0.1, f'{p_between:.3f}', fontsize=12)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

---

7. Phân Phối Đều (`uniform`) - Liên Tục

- **Ký hiệu**: $U(a, b)$
  - `a`: Giá trị nhỏ nhất.
  - `b`: Giá trị lớn nhất.
- **Ý nghĩa**: Mọi giá trị trong khoảng $[a, b]$ đều có cùng khả năng xảy ra (hàm mật độ là hằng số trong khoảng này).
- **Tham số `scipy`**: `loc = a`, `scale = b - a`.

```
from scipy.stats import uniform

a, b = 10, 30
# loc = 10, scale = 30 - 10 = 20
dist_u = uniform(loc=a, scale=(b-a))

# P(X <= 15)
p_u_le_15 = dist_u.cdf(15)
print(f"P(X <= 15) = {p_u_le_15:.2f}") # (15-10) / (30-10) = 0.25
# P(20 <= X <= 25)
p_u_between = dist_u.cdf(25) - dist_u.cdf(20)
print(f"P(20 <= X <= 25) = {p_u_between:.2f}") # (25-20) / (30-10) = 0.25

# Lấy mẫu và vẽ histogram
mau_deu = dist_u.rvs(10000)
plt.hist(mau_deu, bins=20, density=True, alpha=0.6, label='Histogram Mẫu')
# Vẽ PDF lý thuyết
x_u = np.linspace(a - 5, b + 5, 100) # Vẽ rộng hơn khoảng [a,b]
pdf_u = dist_u.pdf(x_u)
plt.plot(x_u, pdf_u, 'r-', lw=2, label='PDF Lý thuyết')
plt.title(f'Phân phối Đều U({a},{b})')
plt.legend()
plt.show()
```

---

8. Phân Phối Mũ (`expon`) - Liên Tục

- **Ký hiệu**: $Exp(\lambda)$
  - λ: Tốc độ (rate) trung bình của sự kiện (ví dụ: số sự kiện / đơn vị thời gian).
- **Ý nghĩa**: Mô tả thời gian chờ cho đến khi sự kiện tiếp theo xảy ra (trong quá trình Poisson).
- **Tham số `scipy`**: `scale = 1 / \lambda`.
  - **Chú ý**: `scale` chính là **thời gian chờ trung bình** giữa các sự kiện. Nếu trung bình thời gian chờ là 5 phút ($1/\lambda = 5$), thì `scale=5`.

```
from scipy.stats import expon

# Ví dụ: Trung bình thời gian chờ xe bus là 10 phút.
# 1/lambda = 10 (thời gian chờ trung bình)
# Ta dùng scale = 10
avg_wait_time = 10
dist_e = expon(scale=avg_wait_time) # loc=0 là mặc định (thời gian không âm)

# P(X <= 5) - Xác suất chờ tối đa 5 phút
p_e_le_5 = dist_e.cdf(5)
print(f"P(X <= 5) = {p_e_le_5:.4f}")
# P(X > 10) - Xác suất chờ hơn 10 phút (hơn thời gian trung bình)
p_e_gt_10 = dist_e.sf(10)
print(f"P(X > 10) = {p_e_gt_10:.4f}") # Khoảng 0.368 (1/e)

# Vẽ PDF và CDF
x_e = np.linspace(0, avg_wait_time * 4, 200) # Vẽ đến khoảng 4 lần trung bình
pdf_e = dist_e.pdf(x_e)
cdf_e = dist_e.cdf(x_e)

fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].plot(x_e, pdf_e, label=f'PDF ($1/\lambda$={avg_wait_time})')
axes[0].set(xlabel='Thời gian chờ (phút)', title='Exponential PDF')
axes[0].legend()
axes[0].grid(True, linestyle='--', alpha=0.7)
```

```
axes[1].plot(x_e, cdf_e, label=f'CDF ($1/\lambda${avg_wait_time})')
axes[1].set(xlabel='Thời gian chờ (phút)', title='Exponential CDF')
axes[1].legend()
axes[1].grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

---

9. Phân Phối Student (`t`) - Liên Tục

- **Ký hiệu**: $t(df)$
  - `df`: Bậc tự do (Degrees of Freedom).
- **Ý nghĩa**: Tương tự phân phối Chuẩn nhưng có "đuôi nặng" (heavy tails), nghĩa là các giá trị cực đoan (xa trung bình) dễ xảy ra hơn, đặc biệt khi `df` nhỏ.
- **Sử dụng**: Khi ước lượng trung bình tổng thể từ mẫu nhỏ (thường $n < 30$) và chưa biết phương sai σ^2 tổng thể. Khi `df` lớn (ví dụ `df > 30`), phân phối `t` xấp xỉ phân phối `norm`.

```
from scipy.stats import t

df_low = 3     # Bậc tự do thấp (đuôi nặng)
df_medium = 10
df_high = 50   # Bậc tự do cao (gần giống normal)

x_t = np.linspace(-5, 5, 200)

plt.figure(figsize=(10, 6))
plt.plot(x_t, norm.pdf(x_t), 'k-', lw=2.5, alpha=0.8, label='Normal (Z)')
plt.plot(x_t, t.pdf(x_t, df=df_low), 'r--', lw=1.5, label=f't (df={df_low})')
plt.plot(x_t, t.pdf(x_t, df=df_medium), 'g-.', lw=1.5, label=f't (df={df_medium})')
plt.plot(x_t, t.pdf(x_t, df=df_high), 'b:', lw=1.5, label=f't (df={df_high})')
plt.legend()
plt.title('So sánh Phân phối t với Phân phối Chuẩn tắc')
plt.xlabel('Giá trị')
plt.ylabel('Mật độ')
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

---

10. 🔬 Ước Lượng Tham Số (Fitting)

- **Mục đích**: Tìm các tham số của một phân phối lý thuyết (ví dụ: μ, σ của `norm`) sao cho phân phối đó khớp (fit) nhất với dữ liệu mẫu ta có.
- **Hàm**: `<dist>.fit(data)` - Trả về các tham số ước lượng được.

```
# 1. Tạo dữ liệu mẫu (ví dụ: từ N(15, 2^2))
np.random.seed(42) # Để kết quả có thể tái lập
mu_true, sigma_true = 15, 2
data_sample = norm.rvs(loc=mu_true, scale=sigma_true, size=1000)

# 2. Giả sử không biết mu, sigma. Dùng .fit() để ước lượng từ data_sample
mu_fit, sigma_fit = norm.fit(data_sample)

print(f"Tham số gốc:      mu = {mu_true:.4f}, sigma = {sigma_true:.4f}")
print(f"Tham số ước lượng: mu = {mu_fit:.4f}, sigma = {sigma_fit:.4f}")

# 3. Vẽ biểu đồ so sánh histogram của dữ liệu và PDF với tham số ước lượng
plt.figure(figsize=(10, 6))
plt.hist(data_sample, bins=30, density=True, alpha=0.6, label='Histogram Dữ liệu Mẫu')
x_fit = np.linspace(data_sample.min(), data_sample.max(), 200)
pdf_fit = norm.pdf(x_fit, loc=mu_fit, scale=sigma_fit)
plt.plot(x_fit, pdf_fit, 'r-', lw=2, label=f'PDF Ước lượng N({mu_fit:.2f}, {sigma_fit:.2f}^2)')
plt.title('Ước lượng tham số Phân phối Chuẩn từ Dữ liệu Mẫu')
plt.xlabel('Giá trị')
plt.ylabel('Mật độ')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

Kiểm tra giữa kỳ

Dưới đây là tóm tắt khung bài làm và các hàm chính được sử dụng trong bài kiểm tra, dựa trên đề bài (`[TKMT] GK.docx`) và đáp án (`DapAn.ipynb`).

🧰 Các Thư Viện Cần Thiết

Khung bài làm tiêu chuẩn luôn bắt đầu bằng việc nhập các thư viện cốt lõi:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import poisson, binom, norm
```

🚗 Bài 1: Phân Tích Dữ Liệu (Dataset: `mpg.csv`)

Bài này tập trung vào kỹ năng thao tác, phân tích và trực quan hóa dữ liệu bằng **Pandas** và **Seaborn**.

1. Đọc và Làm Sạch Dữ Liệu

- [cite_start]**Mục tiêu:** Tải dữ liệu và xử lý các giá trị thiếu hoặc trùng lặp[cite: 47, 48, 49].
- **Cách làm:**
  1. **Đọc file:** `mpg_df = pd.read_csv('mpg.csv')`
  2. **Kiểm tra thiếu:** `mpg_df.isnull().sum()` (Đáp án cho thấy cột 'horsepower' có 6 giá trị thiếu).
  3. **Xử lý thiếu:** `mpg_df.dropna(inplace=True)` (Xóa các dòng có giá trị thiếu).
  4. **Xử lý trùng lặp:** `mpg_df.drop_duplicates(inplace=True)` (Xóa các dòng bị trùng).

2. Tính Thống Kê Cơ Bản

- [cite_start]**Mục tiêu:** Tính các chỉ số thống kê mô tả cho các cột cụ thể[cite: 50, 51].
- **Cách làm:**
  - [cite_start]**Trung bình (Mean):** `mean_values = mpg_df[['cylinders', 'horsepower', 'weight']].mean()` [cite: 50]
  - [cite_start]**Trung vị (Median):** `median_values = mpg_df[['cylinders', 'horsepower', 'weight']].median()` [cite: 50]
  - [cite_start]**Miền giá trị (Range):** `mpg_range = (mpg_df['mpg'].min(), mpg_df['mpg'].max())` [cite: 51]
  - [cite_start]**Phân vị (Quartiles):** [cite: 51]
    - Q1: `mpg_q1 = mpg_df['mpg'].quantile(0.25)`
    - Q3: `mpg_q3 = mpg_df['mpg'].quantile(0.75)`
  - [cite_start]**Nhận xét:** Dựa vào miền giá trị và Q1, Q3 để xem có giá trị nào quá xa không (outlier)[cite: 52].

3. Phân Tích Theo Nhóm (Groupby)

- [cite_start]**Mục tiêu:** Tính giá trị trung bình của 'mpg' cho từng nhóm 'origin' [cite: 53] [cite_start]và tìm nhóm tiêu hao ít nhiên liệu nhất (MPG cao nhất) hoặc nhiều nhất (MPG thấp nhất)[cite: 54].
- **Cách làm:**
  1. **Tính trung bình theo nhóm:** `mpg_by_origin = mpg_df.groupby('origin')['mpg'].mean()`
  2. **Tìm khu vực tiêu hao ít nhiên liệu nhất (MPG cao nhất):** `mpg_by_origin.idxmax()`
  3. [cite_start]**Tìm khu vực tiêu hao nhiều nhiên liệu nhất (MPG thấp nhất):** `mpg_by_origin.idxmin()` (Đáp án dùng `idxmin` để tìm xe tiêu hao *ít* nhiên liệu nhất, điều này có vẻ ngược với đề[cite: 37, 54]. Lưu ý: `mpg` (Miles/Gallon) càng cao thì càng *ít* tốn nhiên liệu. Do đó, `idxmax()` mới là xe *ít* tiêu hao nhiên liệu nhất).

4. Trực Quan Hóa (Phân Bố & So Sánh)

- [cite_start]**Mục tiêu:** Vẽ Histogram của 'weight' [cite: 55] [cite_start]và Boxplot so sánh 'horsepower' giữa các thập niên[cite: 56].
- **Cách làm:**
  - **Histogram (với Seaborn):**

```
sns.histplot(mpg_df['weight'], bins=20, kde=True)
plt.title("Phân bố trọng lượng xe")
plt.xlabel("Trọng lượng (pound)")
plt.ylabel("Số lượng xe")
plt.show()
```

- **Boxplot (với Seaborn):** * *Mẹo:* Để chia 'model_year' (70, 71, ..., 80, 81, 82) thành thập niên 70 và 80, ta dùng phép chia lấy nguyên `// 10 * 10`.

```
# Tạo cột thập niên (70 hoặc 80)
decade = mpg_df['model_year'] // 10 * 10
sns.boxplot(x=decade, y=mpg_df['horsepower'])
plt.title("So sánh công suất giữa thập niên 70 và 80")
plt.xlabel("Thập niên sản xuất")
plt.ylabel("Công suất (hp)")
plt.show()
```

- [cite_start]**Nhận xét:** Nhìn vào boxplot để so sánh trung vị, Q1, Q3 và các điểm ngoại lai (outlier) giữa hai nhóm[cite: 57].

5. Trực Quan Hóa (Mối Quan Hệ)

- [cite_start]**Mục tiêu:** Vẽ 4 biểu đồ Scatter Plot để xem mối quan hệ giữa 'mpg' và các biến khác[cite: 58].
- **Cách làm:**
  - Sử dụng `plt.subplots` để tạo một lưới 2x2.
  - Vẽ từng biểu đồ scatter vào mỗi `ax` (trục) tương ứng.

```
fig, axes = plt.subplots(2, 2, figsize=(12, 10)) # Tạo lưới 2x2

# Biểu đồ 1: MPG vs Cylinders
sns.scatterplot(x=mpg_df['cylinders'], y=mpg_df['mpg'], ax=axes[0, 0])
axes[0, 0].set_title("MPG vs Cylinders")

# Biểu đồ 2: MPG vs Horsepower
sns.scatterplot(x=mpg_df['horsepower'], y=mpg_df['mpg'], ax=axes[0, 1])
axes[0, 1].set_title("MPG vs Horsepower")

# Biểu đồ 3: MPG vs Acceleration
sns.scatterplot(x=mpg_df['acceleration'], y=mpg_df['mpg'], ax=axes[1, 0])
axes[1, 0].set_title("MPG vs Acceleration")

# Biểu đồ 4: MPG vs Model Year
sns.scatterplot(x=mpg_df['model_year'], y=mpg_df['mpg'], ax=axes[1, 1])
axes[1, 1].set_title("MPG vs Model Year")

plt.tight_layout() # Tự động căn chỉnh
plt.show()
```

- [cite_start]**Nhận xét:** Đánh giá yếu tố ảnh hưởng lớn nhất [cite: 59] bằng cách xem biểu đồ nào có xu hướng (tuyến tính, log,...) rõ ràng nhất (ví dụ: 'horsepower' và 'cylinders' có vẻ tương quan nghịch mạnh với 'mpg').

🏭 Bài 2: Phân Phối Xác Suất (Scipy.stats)

Bài này tập trung vào việc áp dụng 3 phân phối xác suất cơ bản để giải quyết bài toán thực tế.

(a) Phân Phối Poisson (Số lần xuất hiện trung bình)

- [cite_start]**Mục tiêu:** Tính xác suất có **nhiều hơn 3 lỗi** ($P(X > 3)$), biết trung bình là 2 lỗi ($\lambda = 2$)[cite: 62, 63].
- **Phân tích:** $P(X > 3) = 1 - P(X \leq 3)$.
- **Hàm:** Sử dụng `poisson.cdf` (Hàm phân phối tích lũy).
  - `k = 3` (số lỗi)
  - `mu = 2` (tham số λ trong `scipy.stats` là `mu`)
- **Cách làm:**

```
# P(X > 3) = 1 - P(X <= 3)
prob_more_than_3 = 1 - poisson.cdf(3, mu=2)
print(f"Xác suất có hơn 3 lỗi: {prob_more_than_3:.4f}") # Kết quả: 0.1429
```

(b) Phân Phối Nhị Thức (Số lần thành công trong `n` phép thử)

- [cite_start]**Mục tiêu:** Tính xác suất có **chính xác 18** vi mạch đạt chuẩn ($P(X = 18)$)[cite: 65].
- **Phân tích:**
  - [cite_start]Số phép thử (tổng số vi mạch): $n = 20$ [cite: 64]
  - [cite_start]Xác suất thành công (đạt chuẩn): $p = 0.95$ [cite: 64]
  - [cite_start]Số lần thành công mong muốn: $k = 18$ [cite: 65]
- **Hàm:** Sử dụng `binom.pmf` (Hàm khối xác suất - cho giá trị chính xác).
- **Cách làm:**

```
prob_18_success = binom.pmf(18, n=20, p=0.95)
print(f"Xác suất có đúng 18 vi mạch đạt: {prob_18_success:.4f}") # Kết quả: 0.1887
```

(c) Phân Phối Chuẩn (Biến liên tục)

- [cite_start]**Mục tiêu 1:** Tính xác suất hoàn thành **trong vòng 45 phút** ($P(X \leq 45)$)[cite: 67].
- **Phân tích:**
  - [cite_start]Trung bình: $\mu = 50$ phút [cite: 66]
  - [cite_start]Độ lệch chuẩn: $\sigma = 5$ phút [cite: 66]
  - [cite_start]Giá trị cần tính: $x = 45$ [cite: 67]
- **Hàm:** Sử dụng `norm.cdf` (Hàm phân phối tích lũy - tính diện tích bên trái của `x`).
- **Cách làm (Mục tiêu 1):**

```
# P(X <= 45) với mu=50, sigma=5
prob_under_45 = norm.cdf(45, loc=50, scale=5)
print(f"Xác suất hoàn thành trong 45 phút: {prob_under_45:.4f}") # Kết quả: 0.1587
```

- [cite_start]**Mục tiêu 2:** Tìm thời gian x để nằm trong **top 5% nhanh nhất** ($P(X \leq x) = 0.05$)[cite: 68].
- **Phân tích:** Đây là bài toán tìm điểm phân vị (ngược của CDF).
  - Xác suất (quantile): $q = 0.05$ (vì là 5% nhanh nhất, tức là 5% ở đuôi bên trái)
- **Hàm:** Sử dụng `norm.ppf` (Hàm điểm phần trăm - Percent Point Function).
- **Cách làm (Mục tiêu 2):**

```
# Tìm x sao cho P(X <= x) = 0.05
thresh_top_5 = norm.ppf(0.05, loc=50, scale=5)
print(f"Thời gian cần để vào top 5% nhanh nhất: {thresh_top_5:.2f} phút") # Kết quả: 41.78 phút
```