Họ tên: Mai Thanh Duy

MSSV: 3123410055

Thống Kê Mô Tả, Xử Lý và Trực Quan Hóa Dữ Liệu

**1.1. THỐNG KÊ MÔ TẢ**

**1.1.1. Ôn tập lý thuyết**

**❓ Thống kê mô tả là gì? Nó khác gì với thống kê suy luận (inferential statistics)?**

**Thống kê mô tả (Descriptive Statistics):**

* Là phương pháp tóm tắt, mô tả và trình bày dữ liệu một cách có ý nghĩa
* Sử dụng các chỉ số như trung bình, trung vị, độ lệch chuẩn, biểu đồ
* Chỉ mô tả dữ liệu hiện có, không đưa ra kết luận về tổng thể
* **Ví dụ:** Tính điểm trung bình của lớp học, vẽ biểu đồ phân bố tuổi

**Thống kê suy luận (Inferential Statistics):**

* Sử dụng dữ liệu mẫu để đưa ra kết luận về tổng thể
* Bao gồm kiểm định giả thuyết, ước lượng khoảng tin cậy
* Có tính dự đoán và khái quát hóa
* **Ví dụ:** Từ khảo sát 1000 người, suy luận về toàn bộ dân số

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Thống kê mô tả** | **Thống kê suy luận** |
| Mục đích | Mô tả dữ liệu | Dự đoán, suy luận |
| Phạm vi | Chỉ dữ liệu hiện có | Mở rộng ra tổng thể |
| Công cụ | Mean, median, std, charts | Hypothesis testing, CI |

**❓ Các thước đo thống kê mô tả chính được sử dụng để làm gì? Khi nào nên dùng trung vị thay vì trung bình?**

**Các thước đo chính:**

**1. Thước đo xu hướng trung tâm (Central Tendency):**

* **Trung bình (Mean):** Giá trị trung bình cộng, nhạy cảm với outliers
* **Trung vị (Median):** Giá trị ở giữa khi sắp xếp, không bị ảnh hưởng bởi outliers
* **Mode:** Giá trị xuất hiện nhiều nhất

**2. Thước đo độ phân tán (Dispersion):**

* **Phương sai (Variance):** Đo lường mức độ phân tán của dữ liệu
* **Độ lệch chuẩn (Standard Deviation):** Căn bậc hai của phương sai
* **Range:** Khoảng cách giữa giá trị lớn nhất và nhỏ nhất
* **IQR (Interquartile Range):** Q3 - Q1, đo độ phân tán của 50% dữ liệu giữa

**⚠️ Khi nào dùng Trung vị thay vì Trung bình?**

* **Có outliers:** Thu nhập (vài người siêu giàu làm méo trung bình)
* **Phân bố lệch:** Giá nhà, lương (thường lệch phải)
* **Dữ liệu thứ tự:** Xếp hạng, đánh giá sao
* **Ví dụ:** Thu nhập trung bình 100 triệu/tháng (có 1 tỷ phú) vs thu nhập trung vị 15 triệu/tháng (thực tế hơn)

**❓ Làm thế nào để xác định phân bố của một tập dữ liệu? Các loại phân bố phổ biến là gì?**

**Cách xác định phân bố:**

1. **Trực quan hóa:** Vẽ histogram, density plot, Q-Q plot
2. **Thống kê mô tả:** So sánh mean, median, mode
3. **Kiểm định thống kê:** Shapiro-Wilk test, Kolmogorov-Smirnov test
4. **Skewness & Kurtosis:** Đo độ lệch và độ nhọn

**Các loại phân bố phổ biến:**

**1. Phân bố chuẩn (Normal Distribution):**

* Hình chuông đối xứng, mean = median = mode
* 68% dữ liệu trong ±1σ, 95% trong ±2σ, 99.7% trong ±3σ
* **Ví dụ:** Chiều cao, cân nặng, điểm thi

**2. Phân bố lệch phải (Right-skewed/Positive skew):**

* Đuôi dài bên phải, mean > median > mode
* **Ví dụ:** Thu nhập, giá nhà, thời gian phản hồi

**3. Phân bố lệch trái (Left-skewed/Negative skew):**

* Đuôi dài bên trái, mode > median > mean
* **Ví dụ:** Tuổi thọ, điểm thi dễ (nhiều người đạt điểm cao)

**4. Phân bố đều (Uniform Distribution):**

* Tất cả giá trị có xác suất như nhau
* **Ví dụ:** Kết quả tung xúc xắc

**💡 Code kiểm tra phân bố:**

import numpy as np

from scipy import stats

import matplotlib.pyplot as plt

# Tạo dữ liệu mẫu

data = np.random.normal(100, 15, 1000)

# Kiểm tra skewness và kurtosis

skewness = stats.skew(data)

kurtosis = stats.kurtosis(data)

print(f"Skewness: {skewness:.3f}") # Gần 0 = đối xứng

print(f"Kurtosis: {kurtosis:.3f}") # Gần 0 = phân bố chuẩn

# Kiểm định Shapiro-Wilk

statistic, p\_value = stats.shapiro(data)

print(f"P-value: {p\_value:.3f}")

if p\_value > 0.05:

print("Dữ liệu có phân bố chuẩn")

else:

print("Dữ liệu không có phân bố chuẩn")

**❓ Độ lệch chuẩn và phạm vi (range) có ý nghĩa gì trong việc đánh giá sự phân tán của dữ liệu?**

**Độ lệch chuẩn (Standard Deviation - σ):**

* **Ý nghĩa:** Đo lường mức độ dữ liệu phân tán xung quanh giá trị trung bình
* **Công thức:** σ = √(Σ(xi - μ)² / N)
* **Ưu điểm:**
  + Sử dụng tất cả các điểm dữ liệu
  + Có ý nghĩa thống kê rõ ràng trong phân bố chuẩn
  + Đơn vị giống với dữ liệu gốc
* **Nhược điểm:** Nhạy cảm với outliers
* **Ví dụ:** Điểm thi trung bình 70, σ=5 (điểm tập trung) vs σ=20 (điểm phân tán)

**Phạm vi (Range):**

* **Ý nghĩa:** Khoảng cách giữa giá trị lớn nhất và nhỏ nhất
* **Công thức:** Range = Max - Min
* **Ưu điểm:**
  + Dễ tính toán và hiểu
  + Cho biết khoảng giá trị của dữ liệu
* **Nhược điểm:**
  + Rất nhạy cảm với outliers
  + Chỉ sử dụng 2 điểm dữ liệu
  + Không phản ánh phân bố thực tế

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Độ lệch chuẩn** | **Range** |
| Độ tin cậy | Cao (dùng tất cả dữ liệu) | Thấp (chỉ dùng 2 điểm) |
| Ảnh hưởng outliers | Trung bình | Rất cao |
| Sử dụng | Phân tích chuyên sâu | Khảo sát nhanh |

**💡 Lời khuyên:** Nên sử dụng cả hai cùng với IQR để có cái nhìn toàn diện về độ phân tán. IQR (Q3-Q1) ít bị ảnh hưởng bởi outliers hơn.

**❓ Sự khác biệt giữa các thước đo như Q1, Q2, Q3 trong biểu đồ hộp (boxplot) là gì?**

**Các Quartile (Tứ phân vị):**

* **Q1 (Quartile 1 - Tứ phân vị thứ nhất):**
  + Giá trị ở vị trí 25% của dữ liệu đã sắp xếp
  + 25% dữ liệu nhỏ hơn Q1
* **Q2 (Quartile 2 - Trung vị/Median):**
  + Giá trị ở vị trí 50% của dữ liệu
  + 50% dữ liệu nhỏ hơn Q2
  + Chia dữ liệu thành 2 phần bằng nhau
* **Q3 (Quartile 3 - Tứ phân vị thứ ba):**
  + Giá trị ở vị trí 75% của dữ liệu
  + 75% dữ liệu nhỏ hơn Q3

**Các thành phần khác trong Boxplot:**

* **IQR (Interquartile Range):** Q3 - Q1 (độ phân tán của 50% dữ liệu giữa)
* **Whiskers (Râu):**
  + Lower whisker: Q1 - 1.5×IQR
  + Upper whisker: Q3 + 1.5×IQR
* **Outliers:** Các điểm nằm ngoài whiskers

**💡 Ví dụ thực tế:**

Dữ liệu điểm thi: [45, 52, 58, 63, 68, 72, 75, 78, 82, 85, 88, 92, 95]

* Q1 = 63 (25% học sinh điểm ≤ 63)
* Q2 = 75 (50% học sinh điểm ≤ 75)
* Q3 = 88 (75% học sinh điểm ≤ 88)
* IQR = 88 - 63 = 25

**💻 Code tạo Boxplot:**

import matplotlib.pyplot as plt

import numpy as np

# Dữ liệu mẫu

data = [45, 52, 58, 63, 68, 72, 75, 78, 82, 85, 88, 92, 95, 120]

# Tính các quartile

q1 = np.percentile(data, 25)

q2 = np.percentile(data, 50) # median

q3 = np.percentile(data, 75)

iqr = q3 - q1

print(f"Q1: {q1}, Q2: {q2}, Q3: {q3}, IQR: {iqr}")

# Vẽ boxplot

plt.figure(figsize=(10, 6))

plt.boxplot(data, vert=False)

plt.title('Boxplot - Phân tích Quartiles')

plt.xlabel('Điểm số')

plt.grid(True, alpha=0.3)

plt.show()

**❓ Làm thế nào để xử lý giá trị thiếu (missing values) trước khi tính toán các chỉ số thống kê mô tả?**

**Các phương pháp xử lý Missing Values:**

**1. Xóa dữ liệu (Deletion):**

* **Listwise deletion:** Xóa toàn bộ hàng có giá trị thiếu
  + ✅ Ưu điểm: Đơn giản, không làm sai lệch dữ liệu
  + ❌ Nhược điểm: Mất nhiều dữ liệu, giảm kích thước mẫu
  + 📌 Khi nào dùng: Tỷ lệ missing < 5%, dữ liệu thiếu ngẫu nhiên
* **Pairwise deletion:** Chỉ xóa khi tính toán cụ thể
  + ✅ Giữ được nhiều dữ liệu hơn
  + ❌ Kích thước mẫu khác nhau cho các phép tính

**2. Thay thế (Imputation):**

* **Mean/Median/Mode Imputation:**
  + Thay bằng trung bình (dữ liệu số, phân bố chuẩn)
  + Thay bằng trung vị (có outliers, phân bố lệch)
  + Thay bằng mode (dữ liệu phân loại)
  + ❌ Nhược điểm: Giảm phương sai, không phản ánh quan hệ giữa biến
* **Forward/Backward Fill:** Dùng cho dữ liệu chuỗi thời gian
* **Interpolation:** Nội suy tuyến tính hoặc đa thức
* **KNN Imputation:** Dùng K láng giềng gần nhất
* **Multiple Imputation:** Tạo nhiều bộ dữ liệu đã điền
* **Model-based:** Dùng ML để dự đoán giá trị thiếu

**3. Giữ nguyên (Keep as is):**

* Một số thuật toán ML xử lý được missing values (XGBoost, LightGBM)
* Tạo biến indicator cho missing (missing\_flag)

**💻 Code xử lý Missing Values:**

import pandas as pd

import numpy as np

from sklearn.impute import SimpleImputer, KNNImputer

# Tạo dữ liệu mẫu

df = pd.DataFrame({

'age': [25, 30, np.nan, 35, 40, np.nan, 45],

'salary': [50000, 60000, 55000, np.nan, 70000, 65000, 75000],

'city': ['HN', 'HCM', 'HN', np.nan, 'DN', 'HCM', 'HN']

})

print("Dữ liệu gốc:")

print(df)

print(f"\nSố lượng missing:\n{df.isnull().sum()}")

print(f"\nTỷ lệ missing:\n{df.isnull().sum() / len(df) \* 100}%")

# Phương pháp 1: Xóa hàng có missing

df\_dropped = df.dropna()

print(f"\nSau khi xóa: {len(df\_dropped)} hàng")

# Phương pháp 2: Thay bằng mean/median

df\_mean = df.copy()

df\_mean['age'].fillna(df\_mean['age'].mean(), inplace=True)

df\_mean['salary'].fillna(df\_mean['salary'].median(), inplace=True)

# Phương pháp 3: Thay bằng mode (cho categorical)

df\_mean['city'].fillna(df\_mean['city'].mode()[0], inplace=True)

print("\nSau khi điền mean/median/mode:")

print(df\_mean)

# Phương pháp 4: KNN Imputation

imputer = KNNImputer(n\_neighbors=2)

df\_knn = df.copy()

df\_knn[['age', 'salary']] = imputer.fit\_transform(df\_knn[['age', 'salary']])

print("\nSau khi dùng KNN Imputation:")

print(df\_knn)

# Phương pháp 5: Forward fill (cho time series)

df\_ffill = df.fillna(method='ffill')

print("\nSau khi forward fill:")

print(df\_ffill)

**⚠️ Lưu ý quan trọng:**

* Luôn phân tích nguyên nhân missing trước khi xử lý
* Missing ngẫu nhiên (MAR) vs Missing có hệ thống (MNAR)
* Ghi nhận phương pháp xử lý để tái tạo được kết quả
* Kiểm tra phân bố trước và sau khi xử lý

**❓ Bạn có thể giải thích cách đọc và diễn giải một biểu đồ histogram hoặc boxplot từ dữ liệu thực tế không?**

**HISTOGRAM - Biểu đồ tần suất:**

**Cách đọc Histogram:**

1. **Trục X:** Các khoảng giá trị (bins)
2. **Trục Y:** Tần suất (số lượng) hoặc mật độ
3. **Chiều cao cột:** Số lượng quan sát trong khoảng đó

**Diễn giải Histogram:**

* **Hình dạng phân bố:**
  + Đối xứng → Phân bố chuẩn
  + Lệch phải → Nhiều giá trị nhỏ, ít giá trị lớn
  + Lệch trái → Nhiều giá trị lớn, ít giá trị nhỏ
  + Nhiều đỉnh → Có thể có nhiều nhóm con
* **Xu hướng trung tâm:** Vị trí của đỉnh cao nhất
* **Độ phân tán:** Độ rộng của phân bố
* **Outliers:** Các cột đơn lẻ xa trung tâm

**BOXPLOT - Biểu đồ hộp:**

**Cách đọc Boxplot:**

* **Hộp (Box):** Chứa 50% dữ liệu giữa (từ Q1 đến Q3)
* **Đường giữa hộp:** Trung vị (Q2)
* **Râu dưới:** Kéo dài đến giá trị nhỏ nhất trong phạm vi hợp lệ
* **Râu trên:** Kéo dài đến giá trị lớn nhất trong phạm vi hợp lệ
* **Các điểm ngoài:** Outliers

**Diễn giải Boxplot:**

* **Độ phân tán:** Chiều cao hộp (IQR) càng lớn → dữ liệu càng phân tán
* **Độ lệch:**
  + Trung vị gần Q1 → Lệch phải
  + Trung vị gần Q3 → Lệch trái
  + Trung vị ở giữa → Đối xứng
* **Outliers:** Số lượng và vị trí các điểm ngoại lai
* **So sánh nhóm:** Dễ so sánh nhiều nhóm cùng lúc

**💻 Code tạo và diễn giải biểu đồ:**

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

# Tạo dữ liệu mẫu: Điểm thi của 3 lớp

np.random.seed(42)

class\_A = np.random.normal(75, 10, 100) # Trung bình 75, std 10

class\_B = np.random.normal(70, 15, 100) # Phân tán hơn

class\_C = np.concatenate([np.random.normal(65, 8, 80),

np.random.normal(90, 5, 20)]) # 2 nhóm

# Tạo figure với 2 subplots

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# HISTOGRAM cho từng lớp

axes[0, 0].hist(class\_A, bins=15, color='skyblue', edgecolor='black', alpha=0.7)

axes[0, 0].set\_title('Histogram - Lớp A\n(Phân bố chuẩn, tập trung)')

axes[0, 0].set\_xlabel('Điểm số')

axes[0, 0].set\_ylabel('Tần suất')

axes[0, 0].axvline(np.mean(class\_A), color='red', linestyle='--', label='Mean')

axes[0, 0].axvline(np.median(class\_A), color='green', linestyle='--', label='Median')

axes[0, 0].legend()

axes[0, 1].hist(class\_B, bins=15, color='lightcoral', edgecolor='black', alpha=0.7)

axes[0, 1].set\_title('Histogram - Lớp B\n(Phân tán rộng)')

axes[0, 1].set\_xlabel('Điểm số')

axes[0, 1].axvline(np.mean(class\_B), color='red', linestyle='--', label='Mean')

axes[0, 1].axvline(np.median(class\_B), color='green', linestyle='--', label='Median')

axes[0, 1].legend()

axes[0, 2].hist(class\_C, bins=15, color='lightgreen', edgecolor='black', alpha=0.7)

axes[0, 2].set\_title('Histogram - Lớp C\n(Hai đỉnh - bimodal)')

axes[0, 2].set\_xlabel('Điểm số')

axes[0, 2].axvline(np.mean(class\_C), color='red', linestyle='--', label='Mean')

axes[0, 2].axvline(np.median(class\_C), color='green', linestyle='--', label='Median')

axes[0, 2].legend()

# BOXPLOT cho từng lớp

data\_all = [class\_A, class\_B, class\_C]

axes[1, 0].boxplot(class\_A, vert=True)

axes[1, 0].set\_title('Boxplot - Lớp A\n(Ít outliers, đối xứng)')

axes[1, 0].set\_ylabel('Điểm số')

axes[1, 1].boxplot(class\_B, vert=True)

axes[1, 1].set\_title('Boxplot - Lớp B\n(IQR lớn, phân tán)')

axes[1, 1].set\_ylabel('Điểm số')

axes[1, 2].boxplot(class\_C, vert=True)

axes[1, 2].set\_title('Boxplot - Lớp C\n(Nhiều outliers)')

axes[1, 2].set\_ylabel('Điểm số')

plt.tight\_layout()

plt.show()

# In thống kê mô tả

print("THỐNG KÊ MÔ TẢ:")

print("\nLớp A:")

print(f" Mean: {np.mean(class\_A):.2f}")

print(f" Median: {np.median(class\_A):.2f}")

print(f" Std: {np.std(class\_A):.2f}")

print(f" Q1: {np.percentile(class\_A, 25):.2f}")

print(f" Q3: {np.percentile(class\_A, 75):.2f}")

print("\nLớp B:")

print(f" Mean: {np.mean(class\_B):.2f}")

print(f" Median: {np.median(class\_B):.2f}")

print(f" Std: {np.std(class\_B):.2f}")

print("\nLớp C:")

print(f" Mean: {np.mean(class\_C):.2f}")

print(f" Median: {np.median(class\_C):.2f}")

print(f" Std: {np.std(class\_C):.2f}")

**💡 Mẹo diễn giải:**

* **Histogram:** Tốt cho xem hình dạng phân bố chi tiết
* **Boxplot:** Tốt cho so sánh nhiều nhóm và phát hiện outliers
* Nên dùng cả hai để có cái nhìn toàn diện
* Kết hợp với thống kê mô tả để diễn giải chính xác

**❓ Khi gặp một tập dữ liệu có giá trị ngoại lai (outliers), bạn sẽ xử lý chúng như thế nào trước khi thực hiện thống kê mô tả?**

**Bước 1: PHÁT HIỆN OUTLIERS**

**Các phương pháp phát hiện:**

* **1. Phương pháp IQR (Interquartile Range):**
  + Outlier nếu: x < Q1 - 1.5×IQR hoặc x > Q3 + 1.5×IQR
  + ✅ Ưu điểm: Không giả định phân bố, dễ hiểu
  + 📌 Phổ biến nhất trong thực tế
* **2. Phương pháp Z-score:**
  + Outlier nếu: |z| > 3 (hoặc 2.5)
  + z = (x - μ) / σ
  + ✅ Tốt cho phân bố chuẩn
* **3. Phương pháp Modified Z-score (MAD):**
  + Sử dụng median thay vì mean
  + ✅ Robust hơn với outliers
* **4. Trực quan hóa:**
  + Boxplot, scatter plot
  + Dễ nhận biết bằng mắt

**Bước 2: PHÂN TÍCH NGUYÊN NHÂN**

* **Lỗi nhập liệu:** Sai số, typo → Cần sửa hoặc xóa
* **Lỗi đo lường:** Thiết bị hỏng → Xóa
* **Giá trị thực tế:** Hiện tượng đặc biệt → Giữ lại hoặc phân tích riêng
* **Phân bố tự nhiên:** Một phần của dữ liệu → Giữ lại

**Bước 3: XỬ LÝ OUTLIERS**

**Các phương pháp xử lý:**

* **1. Giữ nguyên (Keep):**
  + Khi outliers là giá trị thực, có ý nghĩa
  + Dùng thống kê robust (median, IQR)
  + Dùng ML models robust với outliers
* **2. Xóa (Remove):**
  + Khi chắc chắn là lỗi
  + Tỷ lệ outliers < 5%
  + ❌ Mất thông tin, giảm kích thước mẫu
* **3. Capping/Winsorization:**
  + Thay outliers bằng giá trị ngưỡng (percentile 1% và 99%)
  + ✅ Giữ được kích thước mẫu
  + ✅ Giảm ảnh hưởng của outliers
* **4. Transformation:**
  + Log transform: log(x)
  + Square root: √x
  + Box-Cox transformation
  + ✅ Giảm ảnh hưởng outliers, chuẩn hóa phân bố
* **5. Binning:**
  + Chia thành các nhóm/khoảng
  + ✅ Giảm ảnh hưởng outliers
* **6. Imputation:**
  + Thay bằng mean, median, hoặc dự đoán từ model
* **7. Phân tích riêng:**
  + Tạo segment riêng cho outliers
  + Phân tích cả hai nhóm

**💻 Code xử lý Outliers:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

# Tạo dữ liệu mẫu với outliers

np.random.seed(42)

data = np.concatenate([

np.random.normal(100, 15, 95), # Dữ liệu bình thường

[200, 210, 5, 10] # Outliers

])

df = pd.DataFrame({'value': data})

print("=== BƯỚC 1: PHÁT HIỆN OUTLIERS ===\n")

# Phương pháp 1: IQR

Q1 = df['value'].quantile(0.25)

Q3 = df['value'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

outliers\_iqr = df[(df['value'] < lower\_bound) | (df['value'] > upper\_bound)]

print(f"Phương pháp IQR:")

print(f" Q1: {Q1:.2f}, Q3: {Q3:.2f}, IQR: {IQR:.2f}")

print(f" Ngưỡng: [{lower\_bound:.2f}, {upper\_bound:.2f}]")

print(f" Số outliers: {len(outliers\_iqr)}")

print(f" Giá trị outliers: {outliers\_iqr['value'].values}\n")

# Phương pháp 2: Z-score

z\_scores = np.abs(stats.zscore(df['value']))

outliers\_z = df[z\_scores > 3]

print(f"Phương pháp Z-score (|z| > 3):")

print(f" Số outliers: {len(outliers\_z)}")

print(f" Giá trị outliers: {outliers\_z['value'].values}\n")

# Trực quan hóa

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Boxplot gốc

axes[0, 0].boxplot(df['value'])

axes[0, 0].set\_title('Boxplot - Dữ liệu gốc')

axes[0, 0].set\_ylabel('Giá trị')

# Histogram gốc

axes[0, 1].hist(df['value'], bins=20, edgecolor='black')

axes[0, 1].set\_title('Histogram - Dữ liệu gốc')

axes[0, 1].set\_xlabel('Giá trị')

print("=== BƯỚC 2: XỬ LÝ OUTLIERS ===\n")

# Phương pháp 1: Xóa outliers

df\_removed = df[(df['value'] >= lower\_bound) & (df['value'] <= upper\_bound)]

print(f"1. Xóa outliers:")

print(f" Kích thước: {len(df)} → {len(df\_removed)}")

print(f" Mean: {df['value'].mean():.2f} → {df\_removed['value'].mean():.2f}")

print(f" Std: {df['value'].std():.2f} → {df\_removed['value'].std():.2f}\n")

axes[1, 0].boxplot(df\_removed['value'])

axes[1, 0].set\_title('Sau khi xóa outliers')

# Phương pháp 2: Capping (Winsorization)

df\_capped = df.copy()

df\_capped['value'] = df\_capped['value'].clip(lower=lower\_bound, upper=upper\_bound)

print(f"2. Capping (Winsorization):")

print(f" Mean: {df['value'].mean():.2f} → {df\_capped['value'].mean():.2f}")

print(f" Std: {df['value'].std():.2f} → {df\_capped['value'].std():.2f}\n")

axes[1, 1].boxplot(df\_capped['value'])

axes[1, 1].set\_title('Sau khi Capping')

# Phương pháp 3: Log transformation

df\_log = df.copy()

df\_log['value'] = np.log1p(df\_log['value']) # log1p = log(1+x) để tránh log(0)

print(f"3. Log Transformation:")

print(f" Mean: {df['value'].mean():.2f} → {df\_log['value'].mean():.2f}")

print(f" Std: {df['value'].std():.2f} → {df\_log['value'].std():.2f}\n")

axes[1, 2].boxplot(df\_log['value'])

axes[1, 2].set\_title('Sau khi Log Transform')

# Scatter plot để so sánh

axes[0, 2].scatter(range(len(df)), df['value'], alpha=0.6)

axes[0, 2].axhline(y=lower\_bound, color='r', linestyle='--', label='Lower bound')

axes[0, 2].axhline(y=upper\_bound, color='r', linestyle='--', label='Upper bound')

axes[0, 2].set\_title('Scatter plot - Outliers')

axes[0, 2].legend()

plt.tight\_layout()

plt.show()

# So sánh thống kê

print("=== SO SÁNH THỐNG KÊ ===\n")

comparison = pd.DataFrame({

'Gốc': df['value'].describe(),

'Xóa': df\_removed['value'].describe(),

'Capping': df\_capped['value'].describe(),

'Log': df\_log['value'].describe()

})

print(comparison)

**⚠️ Quy tắc vàng khi xử lý Outliers:**

1. **Hiểu trước khi xử lý:** Tại sao có outliers?
2. **Ghi chép:** Luôn document phương pháp xử lý
3. **So sánh:** Phân tích trước và sau xử lý
4. **Domain knowledge:** Tham khảo chuyên gia lĩnh vực
5. **Thử nhiều phương pháp:** So sánh kết quả
6. **Báo cáo:** Nêu rõ trong báo cáo phân tích

|  |  |  |  |
| --- | --- | --- | --- |
| **Phương pháp** | **Ưu điểm** | **Nhược điểm** | **Khi nào dùng** |
| Giữ nguyên | Giữ nguyên thông tin | Ảnh hưởng kết quả | Outliers có ý nghĩa |
| Xóa | Đơn giản, hiệu quả | Mất dữ liệu | Chắc chắn là lỗi |
| Capping | Giữ kích thước mẫu | Thay đổi phân bố | Cần giữ số lượng |
| Transform | Chuẩn hóa phân bố | Khó diễn giải | Phân bố lệch |

**1.2. XỬ LÝ VÀ TRỰC QUAN HÓA DỮ LIỆU**

**1.2.1. Ôn tập lý thuyết**

**❓ Trực quan hóa dữ liệu có vai trò gì trong phân tích dữ liệu? Tại sao nó quan trọng trong khám phá dữ liệu (EDA)?**

**Vai trò của Trực quan hóa dữ liệu:**

**1. Khám phá và Hiểu dữ liệu (Exploration):**

* Nhận diện patterns, trends, và relationships nhanh chóng
* Phát hiện outliers và anomalies
* Hiểu phân bố và đặc điểm của dữ liệu
* **Ví dụ:** Histogram cho thấy phân bố lệch phải của thu nhập

**2. Truyền đạt thông tin (Communication):**

* Dễ hiểu hơn bảng số
* Kể câu chuyện từ dữ liệu (data storytelling)
* Thuyết phục stakeholders
* **Ví dụ:** Dashboard cho CEO dễ hiểu hơn báo cáo 100 trang

**3. Phát hiện vấn đề (Problem Detection):**

* Missing values, duplicates
* Data quality issues
* Imbalanced data
* **Ví dụ:** Heatmap correlation phát hiện multicollinearity

**4. Xác nhận giả thuyết (Hypothesis Validation):**

* Kiểm tra assumptions của model
* Validate kết quả phân tích
* **Ví dụ:** Q-Q plot kiểm tra tính chuẩn của residuals

**5. So sánh và Đối chiếu (Comparison):**

* So sánh nhiều nhóm, thời kỳ
* Benchmark performance
* **Ví dụ:** Bar chart so sánh doanh thu các chi nhánh

**Tại sao quan trọng trong EDA (Exploratory Data Analysis)?**

**🎯 Nguyên tắc "A picture is worth a thousand words"**

Não bộ xử lý hình ảnh nhanh hơn 60,000 lần so với text!

* **Tốc độ:** Nhận diện patterns trong giây lát thay vì phân tích từng số
* **Trực quan:** Dễ nhớ và hiểu hơn bảng số
* **Toàn diện:** Nhìn thấy big picture và chi tiết cùng lúc
* **Tương tác:** Có thể zoom, filter, drill-down
* **Phát hiện bất ngờ:** Tìm ra insights không ngờ tới

**📊 Ví dụ thực tế - Anscombe's Quartet:**

4 datasets có cùng mean, variance, correlation nhưng phân bố hoàn toàn khác nhau!

import matplotlib.pyplot as plt

import seaborn as sns

# Load Anscombe's quartet

df = sns.load\_dataset("anscombe")

# Tính thống kê cho mỗi dataset

for dataset in ['I', 'II', 'III', 'IV']:

data = df[df['dataset'] == dataset]

print(f"\nDataset {dataset}:")

print(f" Mean X: {data['x'].mean():.2f}")

print(f" Mean Y: {data['y'].mean():.2f}")

print(f" Std X: {data['x'].std():.2f}")

print(f" Std Y: {data['y'].std():.2f}")

print(f" Correlation: {data['x'].corr(data['y']):.3f}")

# Vẽ scatter plots

sns.lmplot(x="x", y="y", col="dataset", hue="dataset",

data=df, col\_wrap=2, ci=None, height=4)

plt.suptitle("Anscombe's Quartet - Cùng thống kê, khác phân bố!", y=1.02)

plt.show()

# Kết luận: Chỉ nhìn số liệu không đủ, PHẢI trực quan hóa!

**Quy trình EDA với Trực quan hóa:**

1. **Understand:** Hiểu cấu trúc dữ liệu (df.info(), df.describe())
2. **Visualize:** Vẽ biểu đồ cho từng biến (univariate)
3. **Analyze:** Phân tích mối quan hệ (bivariate, multivariate)
4. **Clean:** Xử lý missing, outliers dựa trên insights
5. **Transform:** Feature engineering dựa trên patterns
6. **Validate:** Kiểm tra lại sau xử lý

|  |  |  |
| --- | --- | --- |
| **Giai đoạn EDA** | **Biểu đồ sử dụng** | **Mục đích** |
| Univariate | Histogram, Boxplot, Bar chart | Hiểu phân bố từng biến |
| Bivariate | Scatter plot, Line plot | Mối quan hệ 2 biến |
| Multivariate | Heatmap, Pair plot | Quan hệ nhiều biến |
| Time series | Line plot, Area chart | Xu hướng theo thời gian |

**❓ Các loại biểu đồ phổ biến được sử dụng trong các trường hợp nào?**

**1. HISTOGRAM - Biểu đồ tần suất:**

* **Mục đích:** Hiển thị phân bố của biến số liên tục
* **Khi nào dùng:**
  + Xem hình dạng phân bố (chuẩn, lệch, bimodal)
  + Phát hiện outliers
  + Hiểu tần suất các giá trị
* **Ví dụ:** Phân bố tuổi khách hàng, điểm thi, thu nhập

**2. SCATTER PLOT - Biểu đồ phân tán:**

* **Mục đích:** Hiển thị mối quan hệ giữa 2 biến số
* **Khi nào dùng:**
  + Tìm correlation (tương quan)
  + Phát hiện patterns, clusters
  + Identify outliers trong 2D
* **Ví dụ:** Chiều cao vs cân nặng, giá nhà vs diện tích

**3. BOXPLOT - Biểu đồ hộp:**

* **Mục đích:** Hiển thị phân bố và outliers
* **Khi nào dùng:**
  + So sánh phân bố nhiều nhóm
  + Phát hiện outliers
  + Xem độ phân tán (IQR)
* **Ví dụ:** So sánh lương theo phòng ban, điểm thi theo lớp

**4. BAR CHART - Biểu đồ cột:**

* **Mục đích:** So sánh giá trị giữa các nhóm
* **Khi nào dùng:**
  + Dữ liệu phân loại (categorical)
  + So sánh số lượng, tỷ lệ
  + Ranking, top N
* **Ví dụ:** Doanh thu theo sản phẩm, số lượng khách hàng theo thành phố

**5. LINE PLOT - Biểu đồ đường:**

* **Mục đích:** Hiển thị xu hướng theo thời gian
* **Khi nào dùng:**
  + Time series data
  + Xu hướng, seasonality
  + So sánh nhiều chuỗi thời gian
* **Ví dụ:** Giá cổ phiếu, doanh thu theo tháng, nhiệt độ theo ngày

**6. PIE CHART - Biểu đồ tròn:**

* **Mục đích:** Hiển thị tỷ lệ phần trăm
* **Khi nào dùng:**
  + Ít hơn 5-6 categories
  + Tổng = 100%
  + Nhấn mạnh một phần
* **Ví dụ:** Market share, phân bố ngân sách
* **⚠️ Lưu ý:** Tránh dùng khi có nhiều categories (dùng bar chart thay thế)

**7. HEATMAP - Bản đồ nhiệt:**

* **Mục đích:** Hiển thị mối quan hệ nhiều biến bằng màu sắc
* **Khi nào dùng:**
  + Correlation matrix
  + Confusion matrix
  + Dữ liệu 2D với nhiều giá trị
* **Ví dụ:** Tương quan giữa các features, performance theo thời gian và khu vực

**8. VIOLIN PLOT:**

* **Mục đích:** Kết hợp boxplot và density plot
* **Khi nào dùng:**
  + Xem cả phân bố và outliers
  + So sánh nhiều nhóm
  + Phát hiện multimodal distribution
* **Ví dụ:** Phân bố lương theo giới tính và phòng ban

**9. PAIR PLOT - Biểu đồ cặp:**

* **Mục đích:** Hiển thị mối quan hệ tất cả các cặp biến
* **Khi nào dùng:**
  + EDA với nhiều biến số
  + Tìm correlation nhanh
  + Feature selection
* **Ví dụ:** Phân tích dataset Iris, Boston Housing

**10. COUNT PLOT:**

* **Mục đích:** Đếm số lượng mỗi category
* **Khi nào dùng:**
  + Dữ liệu categorical
  + Kiểm tra class imbalance
* **Ví dụ:** Số lượng khách hàng theo độ tuổi, phân bố nhãn trong classification

**💻 Code ví dụ cho các loại biểu đồ:**

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd

# Tạo dữ liệu mẫu

np.random.seed(42)

df = pd.DataFrame({

'age': np.random.normal(35, 10, 200),

'salary': np.random.normal(50000, 15000, 200),

'department': np.random.choice(['IT', 'Sales', 'HR', 'Finance'], 200),

'experience': np.random.randint(0, 20, 200),

'satisfaction': np.random.randint(1, 6, 200)

})

# Tạo figure với nhiều subplots

fig = plt.figure(figsize=(16, 12))

# 1. Histogram

plt.subplot(3, 3, 1)

plt.hist(df['age'], bins=20, edgecolor='black', color='skyblue')

plt.title('1. Histogram - Phân bố tuổi')

plt.xlabel('Tuổi')

plt.ylabel('Tần suất')

# 2. Scatter plot

plt.subplot(3, 3, 2)

plt.scatter(df['experience'], df['salary'], alpha=0.5, color='coral')

plt.title('2. Scatter Plot - Kinh nghiệm vs Lương')

plt.xlabel('Kinh nghiệm (năm)')

plt.ylabel('Lương')

# 3. Boxplot

plt.subplot(3, 3, 3)

df.boxplot(column='salary', by='department', ax=plt.gca())

plt.title('3. Boxplot - Lương theo phòng ban')

plt.suptitle('') # Remove default title

# 4. Bar chart

plt.subplot(3, 3, 4)

dept\_counts = df['department'].value\_counts()

plt.bar(dept\_counts.index, dept\_counts.values, color='lightgreen')

plt.title('4. Bar Chart - Số nhân viên theo phòng ban')

plt.xlabel('Phòng ban')

plt.ylabel('Số lượng')

plt.xticks(rotation=45)

# 5. Line plot (giả lập time series)

plt.subplot(3, 3, 5)

time\_data = pd.DataFrame({

'month': range(1, 13),

'revenue': np.cumsum(np.random.randn(12)) + 100

})

plt.plot(time\_data['month'], time\_data['revenue'], marker='o', color='purple')

plt.title('5. Line Plot - Doanh thu theo tháng')

plt.xlabel('Tháng')

plt.ylabel('Doanh thu')

plt.grid(True, alpha=0.3)

# 6. Pie chart

plt.subplot(3, 3, 6)

plt.pie(dept\_counts.values, labels=dept\_counts.index, autopct='%1.1f%%',

startangle=90, colors=['#ff9999','#66b3ff','#99ff99','#ffcc99'])

plt.title('6. Pie Chart - Tỷ lệ nhân viên')

# 7. Heatmap (correlation)

plt.subplot(3, 3, 7)

corr\_data = df[['age', 'salary', 'experience', 'satisfaction']].corr()

sns.heatmap(corr\_data, annot=True, cmap='coolwarm', center=0,

square=True, linewidths=1)

plt.title('7. Heatmap - Ma trận tương quan')

# 8. Violin plot

plt.subplot(3, 3, 8)

sns.violinplot(data=df, x='department', y='salary', palette='Set2')

plt.title('8. Violin Plot - Phân bố lương')

plt.xticks(rotation=45)

# 9. Count plot

plt.subplot(3, 3, 9)

sns.countplot(data=df, x='satisfaction', palette='viridis')

plt.title('9. Count Plot - Mức độ hài lòng')

plt.xlabel('Điểm hài lòng (1-5)')

plt.ylabel('Số lượng')

plt.tight\_layout()

plt.show()

|  |  |  |  |
| --- | --- | --- | --- |
| **Loại biểu đồ** | **Loại dữ liệu** | **Mục đích chính** | **Ví dụ sử dụng** |
| Histogram | Số liên tục | Phân bố | Tuổi, thu nhập |
| Scatter Plot | 2 biến số | Correlation | Chiều cao vs cân nặng |
| Boxplot | Số + Categorical | So sánh, outliers | Lương theo phòng ban |
| Bar Chart | Categorical | So sánh số lượng | Doanh thu sản phẩm |
| Line Plot | Time series | Xu hướng | Giá cổ phiếu |
| Heatmap | Matrix | Correlation | Feature correlation |

**❓ Làm thế nào để chọn loại biểu đồ phù hợp với đặc điểm của dữ liệu?**

**Quy trình chọn biểu đồ - Decision Tree:**

**🎯 Câu hỏi quan trọng nhất: "Tôi muốn truyền đạt thông tin gì?"**

**BƯỚC 1: Xác định loại dữ liệu**

* **Dữ liệu số (Numerical):**
  + Liên tục: Tuổi, chiều cao, thu nhập
  + Rời rạc: Số con, số phòng
* **Dữ liệu phân loại (Categorical):**
  + Nominal: Màu sắc, giới tính, thành phố
  + Ordinal: Học vị, xếp hạng (thấp/trung/cao)
* **Dữ liệu thời gian (Temporal):**
  + Date, datetime, timestamp

**BƯỚC 2: Xác định mục đích phân tích**

**A. So sánh (Comparison):**

* **So sánh giữa các nhóm:**
  + Bar chart (categorical)
  + Boxplot (numerical + categorical)
  + Violin plot (phân bố + so sánh)
* **Ví dụ:** So sánh doanh thu các chi nhánh, lương theo phòng ban

**B. Phân bố (Distribution):**

* **Một biến số:**
  + Histogram (phân bố tần suất)
  + Density plot (phân bố mượt)
  + Boxplot (tóm tắt phân bố)
* **Ví dụ:** Phân bố điểm thi, phân bố tuổi khách hàng

**C. Mối quan hệ (Relationship):**

* **2 biến số:**
  + Scatter plot (correlation)
  + Line plot (nếu có thứ tự)
* **Nhiều biến:**
  + Heatmap (correlation matrix)
  + Pair plot (tất cả các cặp)
* **Ví dụ:** Chiều cao vs cân nặng, correlation giữa features

**D. Xu hướng theo thời gian (Trend):**

* **Time series:**
  + Line plot (xu hướng)
  + Area chart (tích lũy)
  + Candlestick (giá cổ phiếu)
* **Ví dụ:** Doanh thu theo tháng, giá cổ phiếu, nhiệt độ theo ngày

**E. Tỷ lệ/Phần trăm (Proportion):**

* **Composition:**
  + Pie chart (< 5 categories)
  + Stacked bar chart (nhiều categories)
  + Treemap (hierarchical)
* **Ví dụ:** Market share, phân bố ngân sách

**F. Ranking:**

* **Xếp hạng:**
  + Horizontal bar chart (dễ đọc tên)
  + Lollipop chart (đẹp hơn)
* **Ví dụ:** Top 10 sản phẩm bán chạy, xếp hạng thành phố

**BƯỚC 3: Xem xét số lượng biến**

* **1 biến (Univariate):**
  + Số: Histogram, Boxplot, Density plot
  + Categorical: Bar chart, Pie chart, Count plot
* **2 biến (Bivariate):**
  + Số + Số: Scatter plot, Line plot
  + Số + Categorical: Boxplot, Violin plot, Bar chart
  + Categorical + Categorical: Stacked bar, Heatmap
* **3+ biến (Multivariate):**
  + Heatmap, Pair plot, 3D scatter
  + Facet grid (nhiều subplots)

**💡 Decision Tree - Chọn biểu đồ:**

Bắt đầu

│

├─ Mục đích là gì?

│ │

│ ├─ So sánh giữa các nhóm

│ │ └─ Bar chart / Boxplot

│ │

│ ├─ Xem phân bố

│ │ └─ Histogram / Density plot

│ │

│ ├─ Tìm mối quan hệ

│ │ └─ Scatter plot / Heatmap

│ │

│ ├─ Xu hướng thời gian

│ │ └─ Line plot / Area chart

│ │

│ └─ Tỷ lệ phần trăm

│ └─ Pie chart / Stacked bar

│

└─ Loại dữ liệu?

│

├─ Số liên tục

│ └─ Histogram, Scatter, Line

│

├─ Categorical

│ └─ Bar, Pie, Count plot

│

└─ Time series

└─ Line, Area chart

**💻 Code ví dụ - Chọn biểu đồ theo tình huống:**

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

# Tạo dữ liệu mẫu

np.random.seed(42)

df = pd.DataFrame({

'date': pd.date\_range('2024-01-01', periods=100),

'sales': np.cumsum(np.random.randn(100)) + 100,

'category': np.random.choice(['A', 'B', 'C'], 100),

'price': np.random.normal(50, 10, 100),

'quantity': np.random.randint(1, 20, 100)

})

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Tình huống 1: Xem phân bố giá

axes[0, 0].hist(df['price'], bins=20, edgecolor='black', color='skyblue')

axes[0, 0].set\_title('Tình huống 1: Phân bố giá\n→ Dùng Histogram')

axes[0, 0].set\_xlabel('Giá')

# Tình huống 2: So sánh doanh thu theo category

category\_sales = df.groupby('category')['sales'].sum()

axes[0, 1].bar(category\_sales.index, category\_sales.values, color=['#ff9999','#66b3ff','#99ff99'])

axes[0, 1].set\_title('Tình huống 2: So sánh doanh thu\n→ Dùng Bar Chart')

axes[0, 1].set\_xlabel('Category')

# Tình huống 3: Xu hướng doanh thu theo thời gian

axes[0, 2].plot(df['date'], df['sales'], color='purple')

axes[0, 2].set\_title('Tình huống 3: Xu hướng thời gian\n→ Dùng Line Plot')

axes[0, 2].tick\_params(axis='x', rotation=45)

# Tình huống 4: Mối quan hệ giá và số lượng

axes[1, 0].scatter(df['price'], df['quantity'], alpha=0.5, color='coral')

axes[1, 0].set\_title('Tình huống 4: Mối quan hệ 2 biến\n→ Dùng Scatter Plot')

axes[1, 0].set\_xlabel('Giá')

axes[1, 0].set\_ylabel('Số lượng')

# Tình huống 5: Phân bố giá theo category

df.boxplot(column='price', by='category', ax=axes[1, 1])

axes[1, 1].set\_title('Tình huống 5: So sánh phân bố\n→ Dùng Boxplot')

axes[1, 1].set\_xlabel('Category')

plt.sca(axes[1, 1])

plt.xticks(rotation=0)

# Tình huống 6: Tỷ lệ category

category\_counts = df['category'].value\_counts()

axes[1, 2].pie(category\_counts.values, labels=category\_counts.index,

autopct='%1.1f%%', startangle=90)

axes[1, 2].set\_title('Tình huống 6: Tỷ lệ phần trăm\n→ Dùng Pie Chart')

plt.tight\_layout()

plt.show()

|  |  |  |  |
| --- | --- | --- | --- |
| **Tình huống** | **Loại dữ liệu** | **Biểu đồ phù hợp** | **Lý do** |
| Xem phân bố điểm thi | Số liên tục | Histogram | Thấy hình dạng phân bố |
| So sánh doanh thu 4 chi nhánh | Categorical + Số | Bar chart | Dễ so sánh chiều cao |
| Tìm correlation chiều cao-cân nặng | 2 biến số | Scatter plot | Thấy rõ mối quan hệ |
| Xu hướng giá cổ phiếu | Time series | Line plot | Thấy xu hướng liên tục |
| Market share 3 công ty | Categorical (tỷ lệ) | Pie chart | Thấy tỷ lệ phần trăm |
| Phát hiện outliers lương | Số + Categorical | Boxplot | Thấy rõ outliers |

**⚠️ Những sai lầm thường gặp:**

* ❌ Dùng Pie chart khi có > 5 categories → Dùng Bar chart
* ❌ Dùng Line plot cho categorical data → Dùng Bar chart
* ❌ Dùng 3D charts không cần thiết → Khó đọc, gây nhầm lẫn
* ❌ Quá nhiều màu sắc → Gây rối mắt
* ❌ Không có title, labels → Khó hiểu

**❓ Sự khác biệt giữa các thư viện trực quan hóa trong Python như Matplotlib, Seaborn và Plotly là gì?**

**1. MATPLOTLIB - Thư viện cơ bản nhất:**

* **Đặc điểm:**
  + Thư viện trực quan hóa cơ bản và lâu đời nhất của Python
  + Low-level, kiểm soát chi tiết cao
  + Cú pháp phức tạp hơn nhưng linh hoạt
  + Static plots (không tương tác)
* **Ưu điểm:**
  + ✅ Kiểm soát hoàn toàn mọi chi tiết
  + ✅ Tích hợp tốt với NumPy, Pandas
  + ✅ Cộng đồng lớn, tài liệu phong phú
  + ✅ Nền tảng cho nhiều thư viện khác
* **Nhược điểm:**
  + ❌ Cú pháp dài dòng
  + ❌ Mặc định không đẹp
  + ❌ Không tương tác
* **Khi nào dùng:**
  + Cần customization chi tiết
  + Biểu đồ đơn giản, báo cáo tĩnh
  + Học cơ bản về visualization

**2. SEABORN - Matplotlib đẹp hơn:**

* **Đặc điểm:**
  + Built on top of Matplotlib
  + High-level interface, cú pháp đơn giản
  + Mặc định đẹp, themes sẵn có
  + Tích hợp tốt với Pandas DataFrame
  + Chuyên về statistical plots
* **Ưu điểm:**
  + ✅ Cú pháp ngắn gọn, dễ học
  + ✅ Biểu đồ đẹp mặc định
  + ✅ Nhiều plot types cho thống kê
  + ✅ Tự động xử lý DataFrame
* **Nhược điểm:**
  + ❌ Ít linh hoạt hơn Matplotlib
  + ❌ Không tương tác
  + ❌ Chậm hơn với dữ liệu lớn
* **Khi nào dùng:**
  + EDA nhanh
  + Statistical analysis
  + Cần biểu đồ đẹp với ít code

**3. PLOTLY - Biểu đồ tương tác:**

* **Đặc điểm:**
  + Interactive plots (zoom, hover, click)
  + Web-based (HTML/JavaScript)
  + Đẹp và hiện đại
  + Hỗ trợ 3D, animations
* **Ưu điểm:**
  + ✅ Tương tác cao (hover, zoom, filter)
  + ✅ Đẹp và professional
  + ✅ Dễ export sang HTML
  + ✅ Tốt cho dashboards
  + ✅ Hỗ trợ 3D và animations
* **Nhược điểm:**
  + ❌ File size lớn
  + ❌ Chậm với dữ liệu lớn
  + ❌ Cần internet (một số tính năng)
* **Khi nào dùng:**
  + Dashboards, web apps
  + Presentations tương tác
  + Cần 3D hoặc animations

**💻 So sánh code - Cùng một biểu đồ:**

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import pandas as pd

import numpy as np

# Dữ liệu mẫu

df = pd.DataFrame({

'x': np.random.randn(100),

'y': np.random.randn(100),

'category': np.random.choice(['A', 'B', 'C'], 100)

})

# ===== MATPLOTLIB =====

plt.figure(figsize=(8, 6))

for cat in df['category'].unique():

data = df[df['category'] == cat]

plt.scatter(data['x'], data['y'], label=cat, alpha=0.6)

plt.xlabel('X axis')

plt.ylabel('Y axis')

plt.title('Scatter Plot - Matplotlib')

plt.legend()

plt.grid(True, alpha=0.3)

plt.show()

# ===== SEABORN =====

plt.figure(figsize=(8, 6))

sns.scatterplot(data=df, x='x', y='y', hue='category', alpha=0.6)

plt.title('Scatter Plot - Seaborn')

plt.show()

# ===== PLOTLY =====

fig = px.scatter(df, x='x', y='y', color='category',

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **Matplotlib** | **Seaborn** | **Plotly** |
| Độ khó | Khó | Dễ | Trung bình |
| Tương tác | ❌ Không | ❌ Không | ✅ Có |
| Đẹp mặc định | ❌ | ✅ | ✅ |
| Tốc độ | Nhanh | Trung bình | Chậm |
| Customization | Cao nhất | Trung bình | Cao |
| Use case | Báo cáo tĩnh | EDA, thống kê | Dashboard, web |

title='Scatter Plot - Plotly (Interactive)')

fig.show()

# Kết luận:

# - Matplotlib: 8 dòng, kiểm soát chi tiết

# - Seaborn: 2 dòng, đẹp hơn

# - Plotly: 2 dòng, tương tác được