BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH
**KHOA CÔNG NGHỆ THÔNG TIN**


**TS. NGÔ HỮU DŨNG**

**BÀI GIẢNG**

# PHÂN TÍCH DỮ LIỆU VÀ TRÍ TUỆ NHÂN TẠO

**Dành cho bậc đại học**

***Bao gồm nền tảng lý thuyết, câu hỏi ôn tập, bài tập thực hành và các case study thực tiễn***

**THÀNH PHỐ HỒ CHÍ MINH - NĂM 2025**

# LỜI NÓI ĐẦU

Trong thời đại dữ liệu bùng nổ, khoa học dữ liệu (data science), trí tuệ nhân tạo (artificial intelligence – AI) và học máy (machine learning – ML) đã trở thành công cụ thiết yếu trong nghiên cứu khoa học, sản xuất, kinh doanh và đời sống. Sự phát triển mạnh mẽ của Internet, cảm biến, mạng xã hội và thương mại điện tử đã tạo ra khối lượng dữ liệu khổng lồ, vừa là thách thức vừa là cơ hội cho các nhà nghiên cứu và kỹ sư.

Nhằm đáp ứng nhu cầu học tập và nghiên cứu, giáo trình "Phân tích dữ liệu và trí tuệ nhân tạo" được biên soạn với mục tiêu:

- Trang bị kiến thức nền tảng về khoa học dữ liệu, phân tích dữ liệu và trí tuệ nhân tạo.
- Giúp người học làm quen với các công cụ, thư viện Python để thực hành xử lý, trực quan hóa và phân tích dữ liệu.
- Giới thiệu các kỹ thuật cơ bản trong học máy và trí tuệ nhân tạo, đồng thời khơi gợi tư duy phản biện về ứng dụng, xu hướng và thách thức.

Điểm nhấn của giáo trình là tính thực hành: mỗi chương đều có hướng dẫn chi tiết, case study và bài tập ứng dụng. Sinh viên không chỉ tiếp thu lý thuyết mà còn trực tiếp thao tác trên dữ liệu, xây dựng mô hình, trực quan hóa kết quả và rút ra nhận xét. Qua đó, người học rèn luyện tư duy phân tích, kỹ năng lập trình và khả năng ứng dụng vào thực tiễn.

Tài liệu tham khảo được chọn lọc từ các nguồn uy tín, bao gồm sách giáo trình quốc tế, báo cáo nghiên cứu và tài liệu chính thức của các thư viện Python. Bản thảo còn nhiều hạn chế; tác giả mong nhận được góp ý từ đồng nghiệp, sinh viên và bạn đọc để hoàn thiện hơn trong các lần tái bản.

Xin chân thành cảm ơn sự hỗ trợ của nhà trường, đồng nghiệp và sinh viên đã tham gia thảo luận, thực hành và phản biện, góp phần định hình nội dung giáo trình này.

Tác giả tin rằng với tinh thần học tập chủ động và sáng tạo, sinh viên sẽ không chỉ nắm vững kiến thức cơ bản mà còn hình thành kỹ năng phân tích dữ liệu, đặt nền móng vững chắc cho việc nghiên cứu và ứng dụng trí tuệ nhân tạo trong tương lai.

*Thành phố Hồ Chí Minh, ngày 11 tháng 9 năm 2025*

**Người biên soạn**

# MỤC LỤC

# CHƯƠNG 1 NỀN TẢNG KHOA HỌC DỮ LIỆU

Trong thời đại dữ liệu bùng nổ, khả năng **hiểu và phân tích dữ liệu** trở thành kỹ năng thiết yếu trong hầu hết các lĩnh vực, từ kinh doanh, y tế, giáo dục cho đến khoa học kỹ thuật. Chương này cung cấp nền tảng cơ bản của **khoa học dữ liệu (Data Science)**: khái niệm, chu trình phân tích dữ liệu, các công cụ thường dùng và vai trò ứng dụng trong thực tiễn.

Mục tiêu của chương là giúp người học:

- Diễn giải được **khoa học dữ liệu là gì** và tại sao nó quan trọng.
- Trình bày được **chu trình cơ bản** khi làm việc với dữ liệu.
- Làm quen với **các công cụ phổ biến** hỗ trợ phân tích dữ liệu.
- Nhận biết được **vai trò và ứng dụng** của khoa học dữ liệu trong các ngành nghề khác nhau.

Sau khi học xong chương này, người học sẽ có **cái nhìn tổng quan**, làm nền tảng cho các chương sau đi sâu vào kỹ thuật và công cụ cụ thể.

## 1.1 Khái niệm cơ bản

Khoa học dữ liệu (Data Science) là một lĩnh vực liên ngành (interdisciplinary field) kết hợp giữa toán học và thống kê (mathematics and statistics), khoa học máy tính (computer science) và kiến thức miền (domain knowledge) nhằm trích xuất thông tin (information extraction), tri thức (knowledge discovery) và giá trị (value creation) từ dữ liệu. Nói cách khác, khoa học dữ liệu không chỉ tập trung vào thu thập dữ liệu (data collection), mà còn biến dữ liệu thành tri thức hữu ích (useful knowledge) phục vụ cho việc ra quyết định (decision-making) [1], [3].

Trong bối cảnh hiện nay, sự phát triển mạnh mẽ của dữ liệu lớn (Big Data) và trí tuệ nhân tạo (Artificial Intelligence – AI) đã thúc đẩy khoa học dữ liệu trở thành một trong những lĩnh vực quan trọng bậc nhất. Nó không chỉ giới hạn trong nghiên cứu

học thuật (academic research) mà còn được ứng dụng rộng rãi trong kinh doanh (business), y tế (healthcare), tài chính (finance), giao thông (transportation) và cả quản lý xã hội (social governance).

Một số khái niệm liên quan thường gặp:

- Dữ liệu (Data): tập hợp các giá trị (values) được thu thập, có thể ở dạng số (numeric), văn bản (text), hình ảnh (images), âm thanh (audio) hoặc video.
- Thông tin (Information): dữ liệu đã được xử lý (processed) và gắn kết theo ngữ cảnh, giúp người dùng hiểu được ý nghĩa.
- Tri thức (Knowledge): sự tổng hợp (synthesis) và diễn giải (interpretation) từ thông tin, dùng để hỗ trợ ra quyết định.

Điểm cốt lõi của khoa học dữ liệu là tư duy dựa trên dữ liệu (data-driven thinking). Thay vì ra quyết định chỉ dựa vào kinh nghiệm (experience) hoặc trực giác (intuition), các tổ chức và cá nhân ngày càng phụ thuộc vào dữ liệu để đảm bảo khách quan (objectivity), minh bạch (transparency) và hiệu quả (efficiency).

Như vậy, khoa học dữ liệu vừa là một khoa học (science) vì nó cung cấp phương pháp luận (methodology) để khám phá tri thức, vừa là một nghệ thuật (art) vì nó đòi hỏi khả năng sáng tạo (creativity) trong việc đặt câu hỏi, chọn công cụ và diễn giải kết quả. Đây chính là nền tảng để hiểu rõ các chương tiếp theo trong giáo trình.

## 1.2 Chu trình phân tích dữ liệu

Phân tích dữ liệu không phải là một hành động đơn lẻ, mà là một chu trình liên tục (iterative cycle) gồm nhiều bước gắn kết với nhau. Mỗi bước đóng vai trò quan trọng trong việc chuyển hóa dữ liệu thô (raw data) thành tri thức hữu ích (useful knowledge). Chu trình này thường bắt đầu từ việc xác định vấn đề (problem definition): ta cần trả lời câu hỏi gì, giải quyết bài toán nào, hoặc hỗ trợ ra quyết định ra sao. Việc hiểu rõ mục tiêu ngay từ đầu sẽ định hướng toàn bộ quá trình tiếp theo [6].

Sau khi xác định mục tiêu, bước tiếp theo là thu thập dữ liệu (data collection) từ các nguồn khác nhau. Dữ liệu có thể đến từ cơ sở dữ liệu (databases), tệp văn bản (text files), API dịch vụ web (web APIs), cảm biến IoT (IoT sensors), hay khảo sát trực tiếp. Khâu thu thập thường kéo theo việc kiểm tra tính đầy đủ (completeness) và độ tin cậy (reliability) của dữ liệu, bởi dữ liệu sai lệch hoặc thiếu hụt có thể ảnh hưởng nghiêm trọng đến kết quả phân tích.

Tiếp đến là giai đoạn tiền xử lý dữ liệu (data preprocessing). Đây thường là phần tốn nhiều công sức nhất trong chu trình, bao gồm làm sạch dữ liệu (data cleaning), xử lý giá trị thiếu (missing value handling), chuẩn hóa định dạng (data normalization), biến đổi dữ liệu (data transformation) và tích hợp dữ liệu (data integration) từ nhiều nguồn khác nhau. Dữ liệu sau khi tiền xử lý mới có thể sử dụng cho các phân tích hoặc mô hình hóa.

Bước kế tiếp là khám phá và phân tích dữ liệu (data exploration and analysis). Ở giai đoạn này, các phương pháp thống kê mô tả (descriptive statistics) và kỹ thuật trực quan hóa dữ liệu (data visualization) được áp dụng để tìm hiểu phân bố, mối quan hệ giữa các biến, xu hướng nổi bật hoặc điểm bất thường (outliers). Quá trình khám phá giúp hình thành giả thuyết ban đầu và định hướng cho việc áp dụng các mô hình nâng cao hơn.

Sau đó, nếu mục tiêu yêu cầu, dữ liệu sẽ được đưa vào các mô hình phân tích (analytical models) hoặc mô hình học máy (machine learning models). Đây là giai đoạn dự báo (prediction) hoặc phát hiện mẫu ẩn (pattern recognition) trong dữ liệu. Các mô hình có thể được đánh giá bằng những thước đo (metrics) như độ chính xác (accuracy), độ bao phủ (recall), hay sai số trung bình (mean error) để đảm bảo tính tin cậy trước khi ứng dụng vào thực tế.

Cuối cùng, kết quả phân tích cần được diễn giải (interpretation) và truyền đạt (communication) tới người ra quyết định hoặc công chúng. Việc trình bày bằng báo cáo (reports), dashboard trực quan (dashboards), biểu đồ (charts/plots) hoặc công cụ

tương tác giúp người sử dụng dễ dàng hiểu, chấp nhận và áp dụng kết quả. Đây là bước biến dữ liệu thành giá trị thực tiễn (practical value).

Điều quan trọng là chu trình này mang tính lặp lại (iterative). Sau khi ứng dụng, có thể xuất hiện câu hỏi mới hoặc phát hiện ra vấn đề cần điều chỉnh. Khi đó, quá trình quay lại từ bước đầu, tạo thành một vòng xoáy phát triển liên tục, giúp phân tích dữ liệu ngày càng chính xác và phù hợp hơn với thực tế.

## 1.3 Công cụ trong phân tích dữ liệu

Công cụ trong phân tích dữ liệu (tools in data analysis) là tập hợp các ngôn ngữ lập trình, môi trường và phần mềm hỗ trợ xử lý, trực quan hóa và mô hình hóa dữ liệu. Chúng giúp nhà khoa học dữ liệu (data scientists) và nhà phân tích dữ liệu (data analysts) làm việc hiệu quả hơn. Có thể chia thành ba nhóm chính:

### *1.3.1 Ngôn ngữ lập trình*

- **python:** hiện nay là ngôn ngữ phổ biến nhất trong khoa học dữ liệu nhờ hệ sinh thái phong phú với các thư viện như *numpy, pandas, matplotlib, seaborn, scikit-learn, tensorflow, pytorch*.
- **r:** được sử dụng rộng rãi trong thống kê (statistics) và trực quan hóa dữ liệu (data visualization), mạnh ở mảng phân tích học thuật.
- **sql (structured query language):** chuyên dùng để truy vấn và quản lý cơ sở dữ liệu quan hệ (relational databases).

### *1.3.2 Phần mềm và môi trường*

- **jupyter notebook:** một môi trường tương tác (interactive environment) cho phép viết mã (code), văn bản (markdown), và trực quan hóa (visualization) trong cùng một tài liệu.
- **google colab:** phiên bản trực tuyến của jupyter, tích hợp tài nguyên tính toán đám mây (cloud computing), thuận tiện cho việc học tập và cộng tác.

- **anaconda:** nền tảng phân phối python kèm theo nhiều thư viện khoa học dữ liệu, giúp cài đặt và quản lý môi trường dễ dàng.

### *1.3.3 Công cụ trực quan hóa và dashboard*

- **tableau** và **power bi:** công cụ thương mại mạnh mẽ cho việc xây dựng báo cáo và dashboard trực quan, hỗ trợ doanh nghiệp ra quyết định nhanh.
- **matplotlib** và **seaborn:** thư viện mã nguồn mở (open-source libraries) trong python để trực quan hóa dữ liệu.

Việc lựa chọn công cụ phụ thuộc vào mục tiêu (objective), quy mô dữ liệu (data scale), năng lực kỹ thuật (technical capacity) của nhóm, và cả yếu tố chi phí (cost). Trong giảng dạy và nghiên cứu, python và jupyter notebook thường được ưu tiên nhờ tính linh hoạt (flexibility), cộng đồng hỗ trợ mạnh (community support) và miễn phí (free).

## 1.4 Vai trò và ứng dụng

Khoa học dữ liệu giữ vai trò trung tâm (central role) trong việc biến dữ liệu thành tri thức phục vụ cho ra quyết định (decision-making). Ở mức độ căn bản, nó giúp trả lời các câu hỏi: “Điều gì đã xảy ra?”, “Tại sao lại xảy ra?”, “Điều gì có thể xảy ra tiếp theo?” và “Chúng ta nên làm gì?” [8], [9].

Một số vai trò chính:

- Hỗ trợ ra quyết định (decision support): dữ liệu được phân tích để cung cấp bằng chứng khách quan thay vì chỉ dựa vào trực giác.
- Dự báo xu hướng (trend forecasting): thông qua các mô hình dự báo (predictive models), tổ chức có thể chuẩn bị trước cho các tình huống trong tương lai.
- Tối ưu hóa hoạt động (process optimization): dữ liệu giúp phát hiện điểm yếu, cải thiện hiệu quả và giảm chi phí.

- Đổi mới sáng tạo (innovation): từ dữ liệu, doanh nghiệp có thể tìm thấy mô hình kinh doanh (business model) hoặc sản phẩm mới.

Ứng dụng trải rộng trong nhiều lĩnh vực:

- Kinh doanh và marketing: phân tích hành vi khách hàng (customer behavior analysis), dự báo nhu cầu (demand forecasting), đề xuất sản phẩm (recommendation systems).
- Y tế (healthcare): chẩn đoán hỗ trợ bằng dữ liệu (data-driven diagnosis), cá nhân hóa điều trị (personalized medicine), phân tích gen (genomics).
- Tài chính (finance): phát hiện gian lận (fraud detection), chấm điểm tín dụng (credit scoring), đầu tư dựa trên dữ liệu (data-driven investment).
- Khoa học và kỹ thuật: mô phỏng (simulation), xử lý tín hiệu (signal processing), nghiên cứu vật liệu (materials research).
- Xã hội và quản trị công (social governance): phân tích dữ liệu dân số (population data), quản lý đô thị (urban management), chính phủ điện tử (e-government).

Điểm nổi bật là khoa học dữ liệu có tính đa ngành (interdisciplinary): nó có thể kết hợp với bất kỳ lĩnh vực nào có dữ liệu. Chính sự linh hoạt này khiến nó trở thành công cụ không thể thiếu trong kỷ nguyên số (digital era).

## 1.5 Tóm tắt chương

Trong chương này, chúng ta đã tìm hiểu những khái niệm cơ bản của **khoa học dữ liệu (data science)**:

- Dữ liệu là nguồn tài nguyên quan trọng, cần được thu thập, lưu trữ và phân tích để tạo ra giá trị.
- Khoa học dữ liệu kết hợp toán học, thống kê, khoa học máy tính và hiểu biết chuyên ngành để khai thác thông tin từ dữ liệu.

- Quá trình khoa học dữ liệu gồm nhiều bước: thu thập, tiền xử lý, phân tích, mô hình hóa và trực quan hóa.
- Khoa học dữ liệu giữ vai trò trung tâm trong hỗ trợ ra quyết định, dự báo, tối ưu hóa và đổi mới sáng tạo, với ứng dụng rộng rãi trong kinh doanh, y tế, tài chính, khoa học, xã hội và quản trị công.

Tóm lại, khoa học dữ liệu là **nền tảng tri thức của kỷ nguyên số (digital era)**, giúp biến dữ liệu thành hành động và giá trị.

## 1.6 Câu hỏi và bài tập

### *1.6.1 Câu hỏi ôn tập*

*Câu hỏi 1-1 Khoa học dữ liệu là gì? Nó khác gì với thống kê truyền thống?*

*Câu hỏi 1-2 Chu trình phân tích dữ liệu gồm những bước chính nào?*

*Câu hỏi 1-3 Hãy nêu ít nhất ba công cụ phổ biến trong phân tích dữ liệu.*

*Câu hỏi 1-4 Vai trò của dữ liệu trong việc ra quyết định ngày nay quan trọng ra sao?*

*Câu hỏi 1-5 Cho ví dụ một ứng dụng của khoa học dữ liệu trong đời sống hàng ngày.*

### *1.6.2 Hướng dẫn thực hành*

*Thực hành 1-1* ***Làm quen với Python và Jupyter Notebook***

**Mục tiêu:**

- Cài đặt và khởi động môi trường Python.
- Thực hiện một số lệnh Python cơ bản để làm quen với cú pháp.
- Hiểu cách tổ chức mã lệnh và kết quả trong Jupyter Notebook.

***Bước 1. Cài đặt và mở Jupyter Notebook***

- Cài đặt **Anaconda** (khuyên dùng cho sinh viên vì tích hợp sẵn các thư viện).
- Mở **Anaconda Navigator** → chọn **Jupyter Notebook**.
- Tạo một notebook mới (New → Python 3).

***Bước 2. Viết lệnh Python cơ bản***

Trong ô lệnh đầu tiên, gõ:

```
print("Xin chào Khoa học dữ liệu!")
```

**Kết quả mong đợi:** Notebook in ra dòng chữ *"Xin chào Khoa học dữ liệu!"*.

***Bước 3. Biến số và phép toán đơn giản***

```
a = 10
b = 3
tong = a + b
thuong = a / b
print("Tổng =", tong)
print("Thương =", thuong)
```

**Kết quả mong đợi:** Hiển thị kết quả phép cộng và phép chia.

***Bước 4. Cấu trúc dữ liệu cơ bản***

Thử tạo **list** và **dictionary** để lưu trữ dữ liệu:

```
# Danh sách điểm số
diem = [7.5, 8.0, 6.5, 9.0, 7.0]
print("Danh sách điểm:", diem)

# Từ điển thông tin sinh viên
sinhvien = {"ten": "An", "tuoi": 20, "diem": 8.5}
print("Thông tin sinh viên:", sinhvien)
```

***Bước 5. Thống kê nhanh với dữ liệu số***

```
print("Điểm cao nhất:", max(diem))
print("Điểm thấp nhất:", min(diem))
print("Điểm trung bình:", sum(diem)/len(diem))
```

**Kết quả mong đợi:** Sinh viên thấy Python có thể thao tác nhanh trên dữ liệu số.

## *Thực hành 1-2* ***Làm quen với Pandas DataFrame***

**Mục tiêu:**

- Làm quen với thư viện **Pandas** trong Python.
- Biết cách tạo một bảng dữ liệu (DataFrame) đơn giản.
- Thực hiện một số thao tác cơ bản: xem dữ liệu, thống kê nhanh.

***Bước 1. Import thư viện Pandas***

```
import pandas as pd
```

***Bước 2. Tạo DataFrame từ dữ liệu Python***

```
# Tạo dữ liệu mẫu
data = {
    "Ten": ["An", "Bình", "Chi", "Dung"],
    "Tuoi": [20, 21, 19, 22],
    "Diem": [8.0, 7.5, 9.0, 6.5]
}

# Chuyển thành DataFrame
df = pd.DataFrame(data)

# Xem bảng dữ liệu
print(df)
```

**Kết quả mong đợi:** Một bảng dữ liệu 3 cột (Tên, Tuổi, Điểm) được in ra như bảng Excel nhỏ.

***Bước 3. Quan sát dữ liệu***

```
# Xem 2 dòng đầu tiên
print(df.head(2))

# Thông tin về số dòng, số cột, kiểu dữ liệu
print(df.info())

# Thống kê mô tả
print(df.describe())
```

**Kết quả mong đợi:**

- head() hiển thị vài dòng đầu tiên.
- info() cho biết số dòng/cột và kiểu dữ liệu.
- describe() cho biết trung bình, min, max, độ lệch chuẩn của các cột số.

***Bước 4. Truy xuất dữ liệu theo cột hoặc dòng***

```
# Truy xuất cột
print(df["Ten"])      # cột Tên
print(df[["Ten","Diem"]])   # nhiều cột

# Truy xuất dòng
print(df.iloc[0])     # dòng đầu tiên
print(df.iloc[1:3])   # từ dòng 2 đến 3
```

***Bước 5. Tính toán trên cột dữ liệu***

```
print("Điểm trung bình:", df["Diem"].mean())
print("Điểm cao nhất:", df["Diem"].max())
print("Điểm thấp nhất:", df["Diem"].min())
```

## *Thực hành 1-3* ***Thống kê mô tả cơ bản***

**Mục tiêu:**

- Làm quen với các chỉ số thống kê thường dùng: mean, median, mode, min, max, std.
- Biết cách tính trực tiếp bằng Pandas.
- Hiểu ý nghĩa của các chỉ số trong mô tả dữ liệu.

***Bước 1. Chuẩn bị dữ liệu mẫu***

```
import pandas as pd

data = {
    "Ten": ["An", "Bình", "Chi", "Dung", "Hà"],
    "Tuoi": [20, 21, 19, 22, 20],
    "Diem": [8.0, 7.5, 9.0, 6.5, 7.0]
}

df = pd.DataFrame(data)
print(df)
```

***Bước 2. Tính các chỉ số cơ bản***

```
print("Điểm trung bình:", df["Diem"].mean())       # Mean
print("Trung vị:", df["Diem"].median())            # Median
print("Giá trị xuất hiện nhiều nhất:", df["Diem"].mode()[0])
# Mode
print("Điểm cao nhất:", df["Diem"].max())
print("Điểm thấp nhất:", df["Diem"].min())
print("Độ lệch chuẩn:", df["Diem"].std())
```

**Kết quả mong đợi:**

- Sinh viên thấy được giá trị trung bình, trung vị, mode và độ phân tán của dữ liệu.

***Bước 3. Tóm tắt nhanh bằng describe()***

```
print(df["Diem"].describe())
```

**Giải thích:**

- count: số lượng phần tử.
- mean: trung bình.
- std: độ lệch chuẩn.
- min, 25%, 50%, 75%, max: giá trị nhỏ nhất, phân vị và lớn nhất.

***Bước 4. Phân tích ý nghĩa***

Ví dụ từ dữ liệu mẫu:

- Điểm trung bình ≈ 7.6 → lớp có mức khá.
- Độ lệch chuẩn nhỏ (< 1) → điểm các bạn không chênh lệch nhiều.
- Trung vị (7.5) gần với trung bình → dữ liệu phân bố khá đều.

## *Thực hành 1-4* ***Trực quan hóa dữ liệu cơ bản***

**Mục tiêu:**

- Làm quen với thư viện **Matplotlib**.

- Vẽ các biểu đồ cơ bản: **bar chart, line chart, histogram**.
- Hiểu cách trực quan hóa giúp diễn giải dữ liệu dễ hơn.

***Bước 1. Import thư viện***

```
import matplotlib.pyplot as plt
import pandas as pd

# Tạo dữ liệu mẫu
data = {
    "Ten": ["An", "Bình", "Chi", "Dung", "Hà"],
    "Diem": [8.0, 7.5, 9.0, 6.5, 7.0]
}
df = pd.DataFrame(data)
```

***Bước 2. Biểu đồ cột (Bar chart)***

Dùng để so sánh điểm số giữa các sinh viên.

```
plt.bar(df["Ten"], df["Diem"], color="skyblue",
edgecolor="black")
plt.xlabel("Sinh viên")
plt.ylabel("Điểm")
plt.title("Điểm số theo từng sinh viên")
plt.show()
```

***Bước 3. Biểu đồ đường (Line chart)***

Dùng để thể hiện xu hướng (ví dụ theo thứ tự danh sách).

```
plt.plot(df["Ten"], df["Diem"], marker="o", linestyle="-",
color="green")
plt.xlabel("Sinh viên")
plt.ylabel("Điểm")
plt.title("Xu hướng điểm số")
plt.show()
```

***Bước 4. Biểu đồ phân bố (Histogram)***

Dùng để xem phân phối điểm số trong lớp.

```
plt.hist(df["Diem"], bins=5, color="orange",
edgecolor="black")
plt.xlabel("Điểm")
```

```
plt.ylabel("Số lượng sinh viên")
plt.title("Phân bố điểm số")
plt.show()
```

***Bước 5. Giải thích kết quả***

- **Bar chart:** dễ so sánh ai cao điểm, ai thấp điểm.
- **Line chart:** thấy được xu hướng tăng/giảm.
- **Histogram:** cho biết phần lớn sinh viên tập trung ở mức điểm nào.

### *1.6.3 Bài tập mở rộng*

*Bài tập 1-1 Khám phá dữ liệu cá nhân*

- Thu thập một bộ dữ liệu nhỏ (vd: điểm số học tập, chi tiêu trong tuần, số bước đi mỗi ngày).
- Tính các chỉ số cơ bản: min, max, trung bình, độ lệch chuẩn.
- Vẽ biểu đồ để minh họa dữ liệu.
- Viết một đoạn ngắn (5–7 câu) diễn giải ý nghĩa dữ liệu.

*Bài tập 1-2 Tìm ứng dụng khoa học dữ liệu*

- Chọn một lĩnh vực (y tế, giáo dục, giao thông, tài chính, thương mại điện tử).
- Tìm 1–2 ví dụ thực tế về ứng dụng khoa học dữ liệu.
    - Viết một đoạn trình bày (khoảng ½ trang).

# CHƯƠNG 2 THU THẬP VÀ TIỀN XỬ LÝ DỮ LIỆU

Chương này tập trung vào thu thập dữ liệu (data collection) và tiền xử lý dữ liệu (data preprocessing), hai bước nền tảng trong khoa học dữ liệu. Dữ liệu thô từ các nguồn khác nhau thường chưa đồng nhất, thiếu hoặc chứa lỗi, do đó cần được làm sạch và chuẩn hóa trước khi tiến hành phân tích hoặc mô hình hóa.

Trong chương này, bạn sẽ được làm quen với:

- Các nguồn dữ liệu phổ biến, từ dữ liệu có cấu trúc (bảng, CSV, Excel, SQL) đến dữ liệu phi cấu trúc (văn bản, hình ảnh, video).
- Các định dạng dữ liệu thông dụng và cách lựa chọn định dạng phù hợp cho phân tích.
- Các bước tiền xử lý dữ liệu như xử lý missing values, chuẩn hóa dữ liệu, mã hóa dữ liệu danh mục và loại bỏ outliers.
- Một số công cụ Python hỗ trợ thu thập và tiền xử lý dữ liệu.
- Cách tích hợp dữ liệu từ nhiều nguồn để tạo dataset duy nhất cho phân tích.

Sau khi học xong chương này, sinh viên sẽ có khả năng:

1. Xác định các nguồn dữ liệu phù hợp cho mục tiêu phân tích.
2. Nhận biết và xử lý các vấn đề dữ liệu thô, bao gồm missing values, dữ liệu lỗi, outliers.
3. Chuẩn hóa và mã hóa dữ liệu để phục vụ phân tích và mô hình hóa.
4. Sử dụng các thư viện Python cơ bản (pandas, numpy, matplotlib, scikit-learn) cho thu thập và tiền xử lý dữ liệu.
5. Kết hợp dữ liệu từ nhiều nguồn thành dataset đồng nhất để phân tích thực tế.

## 2.1 Nguồn dữ liệu

Dữ liệu trong khoa học dữ liệu có thể đến từ nhiều nguồn khác nhau (data sources), và mỗi nguồn có đặc điểm, ưu nhược điểm riêng. Việc hiểu rõ nguồn dữ liệu giúp lựa chọn phương pháp thu thập và xử lý phù hợp [8].

Một số loại nguồn dữ liệu phổ biến:

1. Dữ liệu có cấu trúc (structured data):

   Đây là dữ liệu được tổ chức theo dạng bảng, với các hàng và cột rõ ràng, ví dụ dữ liệu trong cơ sở dữ liệu (databases), CSV hoặc Excel. Dữ liệu có cấu trúc dễ phân tích bằng các công cụ thống kê và Python (pandas, numpy).

2. Dữ liệu bán cấu trúc (semi-structured data):

   Dữ liệu bán cấu trúc không hoàn toàn theo bảng, nhưng vẫn có một số định dạng cố định để nhận dạng thông tin. Ví dụ JSON, XML hoặc log files. Dữ liệu bán cấu trúc thường xuất hiện khi thu thập từ API hoặc các hệ thống trực tuyến.

3. Dữ liệu phi cấu trúc (unstructured data):

   Đây là dữ liệu không theo định dạng bảng, ví dụ văn bản, hình ảnh, video, âm thanh. Phân tích dữ liệu phi cấu trúc thường đòi hỏi các kỹ thuật đặc thù như xử lý ngôn ngữ tự nhiên (Natural Language Processing) hoặc thị giác máy tính (Computer Vision).

4. Nguồn dữ liệu trực tuyến (online sources):

   Nhiều dữ liệu có thể truy cập trực tuyến thông qua API, web scraping, hoặc các kho dữ liệu mở (open data portals). Các công cụ Python như requests, BeautifulSoup hoặc Scrapy hỗ trợ việc thu thập dữ liệu từ web.

Việc lựa chọn nguồn dữ liệu phù hợp phụ thuộc vào mục tiêu phân tích (analysis objectives) và khả năng xử lý (computational capacity). Trong thực tế, một dự án thường kết hợp dữ liệu từ nhiều nguồn để tăng tính đầy đủ và chất lượng của dataset.

## 2.2 Định dạng dữ liệu phổ biến

Trong khoa học dữ liệu, dữ liệu được lưu trữ và trao đổi dưới nhiều định dạng khác nhau (data formats). Hiểu rõ định dạng giúp bạn chọn công cụ đọc, xử lý và phân tích phù hợp.

1. CSV (Comma-Separated Values):

   CSV là định dạng phổ biến nhất cho dữ liệu bảng. Mỗi dòng là một bản ghi (record), các giá trị được phân tách bằng dấu phẩy hoặc ký tự khác. CSV dễ đọc bằng Excel, Python (pandas) và nhiều phần mềm khác.

2. Excel (.xlsx, .xls):

   Excel là định dạng được sử dụng rộng rãi trong doanh nghiệp, hỗ trợ nhiều sheet, công thức và định dạng dữ liệu khác nhau. Python có thể thao tác Excel qua thư viện pandas hoặc openpyxl/xlrd.

3. JSON (JavaScript Object Notation):

   JSON là định dạng phổ biến cho dữ liệu bán cấu trúc, thường dùng khi làm việc với API hoặc web services. JSON biểu diễn dữ liệu dưới dạng key–value, dễ đọc và dễ parse bằng Python (json module).

4. SQL Database:

   Dữ liệu có cấu trúc được lưu trong cơ sở dữ liệu quan hệ (relational databases). SQL (Structured Query Language) cho phép truy vấn, lọc, nhóm và kết hợp dữ liệu từ nhiều bảng. Python có thể kết nối với SQL thông qua SQLAlchemy hoặc pandas.read_sql().

5. Các định dạng khác:

   - Parquet / Feather: định dạng tối ưu cho dữ liệu lớn (big data), truy cập nhanh, hỗ trợ compression.
   - HDF5: lưu trữ dữ liệu khoa học lớn, có thể đọc/ghi nhanh.
   - Text / log files: dữ liệu thô dạng văn bản, thường cần xử lý tiền xử lý trước khi phân tích.

Hiểu được các định dạng này giúp bạn lựa chọn phương pháp thu thập, tiền xử lý và tích hợp dữ liệu (data preprocessing and integration) phù hợp.

## 2.3 Tiền xử lý dữ liệu

Dữ liệu thô từ các nguồn khác nhau thường chưa hoàn thiện: có giá trị thiếu, lỗi nhập, định dạng không đồng nhất, hoặc chứa các outlier. Do đó, tiền xử lý dữ liệu (data preprocessing) là bước quan trọng để chuẩn bị dữ liệu sạch và đồng bộ cho phân tích hoặc mô hình hóa [7].

Các bước chính trong tiền xử lý dữ liệu

1. Xử lý dữ liệu thiếu (handling missing values):
    - Loại bỏ các dòng hoặc cột có giá trị thiếu nếu dữ liệu còn đủ.
    - Điền giá trị thay thế bằng mean, median, mode hoặc dự đoán từ các biến liên quan.
    - Python hỗ trợ qua pandas: df.fillna(), df.dropna().
2. Chuẩn hóa dữ liệu (data normalization / standardization):
    - Biến đổi dữ liệu về cùng thang đo để thuận tiện phân tích hoặc huấn luyện mô hình.
    - Normalization: đưa dữ liệu vào khoảng [0, 1] (MinMaxScaler trong scikit-learn).
    - Standardization: chuẩn hóa về trung bình 0, độ lệch chuẩn 1 (StandardScaler).
3. Mã hóa dữ liệu danh mục (encoding categorical data):
    - Chuyển đổi dữ liệu dạng chữ (nominal/ordinal) sang số.
    - Ví dụ: One-Hot Encoding (pd.get_dummies) hoặc Label Encoding (LabelEncoder).
4. Loại bỏ dữ liệu ngoại lai (outlier detection / removal):
    - Xác định các giá trị cực đoan có thể ảnh hưởng đến kết quả.
    - Phương pháp phổ biến: boxplot, z-score, IQR (Interquartile Range).
5. Tích hợp dữ liệu (data integration):
    - Khi dữ liệu từ nhiều nguồn, cần kết hợp thành dataset duy nhất.

- Chú ý đồng nhất các định dạng, loại bỏ trùng lặp, và đồng bộ hóa các cột thông tin.
- Python hỗ trợ qua pandas.merge(), concat().

Lưu ý thực tế

- Tiền xử lý dữ liệu không phải lúc nào cũng theo quy trình cố định, mà tùy thuộc vào mục tiêu phân tích và loại dữ liệu.
- Một dataset chất lượng tốt sẽ giúp phân tích chính xác hơn, giảm sai số trong mô hình hóa và đưa ra quyết định hiệu quả.

## 2.4 Tích hợp và biến đổi dữ liệu

Trong thực tế, dữ liệu thường đến từ nhiều nguồn khác nhau (data sources) với định dạng và cấu trúc không đồng nhất. Để phục vụ phân tích hoặc mô hình hóa, cần thực hiện tích hợp (integration) và biến đổi dữ liệu (transformation).

### *2.4.1 Tích hợp dữ liệu*

- Kết hợp dữ liệu từ nhiều nguồn: Dữ liệu từ CSV, Excel, SQL, API… thường phải gộp lại thành dataset duy nhất.
- Đồng bộ hóa thông tin: Chuẩn hóa tên cột, kiểu dữ liệu, đơn vị đo lường (ví dụ: USD và VND, hoặc cm và m).
- Loại bỏ trùng lặp: Khi ghép nhiều bảng, một số bản ghi có thể trùng nhau, cần xử lý để tránh sai lệch phân tích.
- Kỹ thuật phổ biến:
  - pandas.merge() để nối các bảng theo key chung.
  - pandas.concat() để ghép các bảng theo hàng hoặc cột.
  - Xử lý các bản ghi trùng bằng drop_duplicates().

### *2.4.2 Biến đổi dữ liệu*

Chuyển đổi định dạng dữ liệu: Ví dụ chuyển cột ngày tháng từ dạng text sang datetime (pd.to_datetime).

Tạo biến mới (feature engineering):

- Ghép các cột để tạo thông tin mới (ví dụ: total_price = quantity * unit_price).
- Rút trích thông tin từ văn bản, ngày tháng, hoặc địa chỉ.

Biến đổi thang đo dữ liệu: Chuẩn hóa (normalization) hoặc chuẩn hóa theo z-score (standardization) để dữ liệu phù hợp với thuật toán phân tích hoặc mô hình học máy.

Mã hóa dữ liệu danh mục (categorical encoding): Chuyển đổi các giá trị chữ sang số để mô hình xử lý, ví dụ One-Hot Encoding hoặc Label Encoding.

### *2.4.3 Lưu ý thực hành*

- Khi tích hợp dữ liệu từ nhiều nguồn, luôn kiểm tra số lượng bản ghi, giá trị thiếu và các outlier trước và sau khi kết hợp.
- Biến đổi dữ liệu cần ghi chú và lưu lại các bước (data pipeline), để có thể lặp lại phân tích hoặc cập nhật dữ liệu mới dễ dàng.
- Python cung cấp đầy đủ công cụ: pandas, numpy, scikit-learn, giúp tích hợp và biến đổi dữ liệu hiệu quả.

## 2.5 Công cụ hỗ trợ trong Python

Python là ngôn ngữ phổ biến trong khoa học dữ liệu nhờ thư viện phong phú và cú pháp dễ đọc [9], [10], [11]. Trong việc thu thập, tiền xử lý, tích hợp và biến đổi dữ liệu, một số thư viện chính bao gồm:

### *2.5.1 pandas*

- Hỗ trợ đọc/ghi dữ liệu từ CSV, Excel, SQL, JSON.
- Thao tác dữ liệu bảng: chọn cột/ dòng, lọc dữ liệu, nhóm (groupby), kết hợp bảng (merge, concat), xử lý missing values (fillna, dropna).
- Ví dụ:

```
import pandas as pd

# đọc dữ liệu CSV
df = pd.read_csv('data.csv')

# kiểm tra thông tin cơ bản
print(df.info())
print(df.describe())
```

```
# điền giá trị thiếu bằng trung bình
df['age'].fillna(df['age'].mean(), inplace=True)
```

### *2.5.2 numpy*

- Thư viện tính toán số học, mảng đa chiều (arrays).
- Hỗ trợ các phép toán vector hóa nhanh hơn vòng lặp Python thuần.
- Ví dụ tính trung bình, chuẩn hóa dữ liệu:

```
import numpy as np

arr = np.array([7.5, 8.0, 6.5, 9.0, 7.0])
mean_val = np.mean(arr)
std_val = np.std(arr)
normalized = (arr - mean_val)/std_val
```

### *2.5.3 Scikit-learn*

- Thư viện hỗ trợ chuẩn hóa dữ liệu, mã hóa biến, split dữ liệu thành train/test.
- Các công cụ preprocessing: StandardScaler, MinMaxScaler, LabelEncoder, OneHotEncoder.
- Ví dụ chuẩn hóa dữ liệu:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df[['income', 'age']])
```

### *2.5.4 Thư viện hỗ trợ thu thập dữ liệu trực tuyến*

- **requests:** gửi request HTTP, lấy dữ liệu từ API.
- **BeautifulSoup:** phân tích HTML, thu thập dữ liệu web.
- **Scrapy:** framework mạnh mẽ để crawling web.
- Ví dụ lấy dữ liệu từ web:

```
import requests
```

```
from bs4 import BeautifulSoup

url = "https://example.com"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
titles = [t.text for t in soup.find_all('h2')]
```

Python cung cấp các công cụ này giúp bạn **thu thập, xử lý, tích hợp và biến đổi dữ liệu một cách linh hoạt và hiệu quả**, chuẩn bị cho bước phân tích và mô hình hóa tiếp theo.

## 2.6 Tóm tắt chương

Trong chương này, chúng ta đã tìm hiểu các bước cơ bản để chuẩn bị dữ liệu cho phân tích:

1. **Nguồn dữ liệu:** Dữ liệu có thể đến từ nhiều nguồn khác nhau, bao gồm dữ liệu có cấu trúc (structured), bán cấu trúc (semi-structured) và phi cấu trúc (unstructured). Việc hiểu rõ nguồn dữ liệu giúp lựa chọn phương pháp thu thập và xử lý phù hợp.
2. **Định dạng dữ liệu phổ biến:** Các định dạng thường gặp gồm CSV, Excel, JSON, cơ sở dữ liệu SQL và các định dạng tối ưu cho big data như Parquet, HDF5. Chọn đúng định dạng giúp thao tác và phân tích dễ dàng hơn.
3. **Tiền xử lý dữ liệu (data preprocessing):** Đây là bước quan trọng để xử lý dữ liệu thiếu, loại bỏ outlier, chuẩn hóa dữ liệu, và mã hóa dữ liệu danh mục, nhằm đảm bảo dataset sạch, đồng nhất và sẵn sàng cho phân tích.
4. **Tích hợp và biến đổi dữ liệu (data integration and transformation):** Kết hợp dữ liệu từ nhiều nguồn, chuẩn hóa thông tin, tạo biến mới và biến đổi thang đo dữ liệu là tiền đề quan trọng để thực hiện phân tích thống kê và mô hình hóa.
5. **Công cụ hỗ trợ trong Python:** Python cung cấp các thư viện mạnh mẽ như pandas, numpy, scikit-learn, matplotlib, giúp thu thập, tiền xử lý, tích hợp và biến đổi dữ liệu hiệu quả.

Chương này đặt nền tảng cho các chương tiếp theo, đặc biệt là **khám phá dữ liệu, trực quan hóa, và xây dựng mô hình học máy**, bằng cách đảm bảo dữ liệu đầu vào sạch, đồng bộ và chất lượng cao.

## 2.7 Câu hỏi và bài tập

### *2.7.1 Câu hỏi ôn tập*

*Câu hỏi 2-1* *Liệt kê các loại nguồn dữ liệu thường gặp và nêu ví dụ cho từng loại.*

*Câu hỏi 2-2* *So sánh ưu nhược điểm của CSV, Excel và JSON khi lưu trữ dữ liệu.*

*Câu hỏi 2-3* *Nêu các bước chính trong tiền xử lý dữ liệu (data preprocessing) và giải thích lý do cần thực hiện.*

*Câu hỏi 2-4* *Giải thích sự khác nhau giữa normalization và standardization.*

*Câu hỏi 2-5* *Khi tích hợp dữ liệu từ nhiều nguồn, cần lưu ý những vấn đề gì?*

*Câu hỏi 2-6* *Nêu các thư viện Python phổ biến dùng cho tiền xử lý và tích hợp dữ liệu.*

### *2.7.2 Hướng dẫn thực hành*

*Thực hành 2-1* ***Đọc dữ liệu từ CSV và quan sát cấu trúc***

**Mục tiêu:**

- Làm quen với cách đọc dữ liệu từ file CSV bằng thư viện Pandas.
- Hiểu cấu trúc cơ bản của một DataFrame.
- Biết cách xem nhanh dữ liệu và các thông tin quan trọng (số dòng, số cột, kiểu dữ liệu).

**Dữ liệu sử dụng:** lesson01_data.csv (được cung cấp trong thư mục data).

***Bước 1. Import thư viện và đọc dữ liệu***

```
import pandas as pd

# Đọc dữ liệu từ file CSV
df = pd.read_csv('lesson01_data.csv', header=0, delimiter=',',
encoding='utf-8')
```

***Bước 2. Xem dữ liệu ban đầu***

```
# Hiển thị toàn bộ DataFrame (chỉ nên dùng với dữ liệu nhỏ)
df
```

```
# Hiển thị 10 dòng đầu tiên
df.head(10)

# Hiển thị 10 dòng cuối cùng
df.tail(10)
```

***Bước 3. Kiểm tra thông tin cơ bản của dữ liệu***

```
# Danh sách tên cột
df.columns

# Kích thước (số dòng, số cột)
df.shape

# Kiểu dữ liệu của từng cột
df.dtypes

# Chỉ số dòng
df.index

# Thông tin tổng quan (số lượng non-null, kiểu dữ liệu, bộ
nhớ)
df.info()
```

**Kết quả mong đợi:**

- Sinh viên biết cách đọc file CSV vào DataFrame bằng Pandas.
- Quan sát được các thành phần cơ bản: tên cột, số dòng/cột, kiểu dữ liệu.
- Bắt đầu hình thành kỹ năng **hiểu dữ liệu (data understanding)** trước khi xử lý.

## *Thực hành 2-2* ***Tiền xử lý cơ bản (đổi tên, kiểm tra kiểu, lọc dữ liệu)***

**Mục tiêu:**

- Làm quen với các thao tác tiền xử lý cơ bản trong Pandas.
- Đổi tên cột để dễ hiểu hơn.
- Chọn cột, chọn dòng, lọc dữ liệu theo điều kiện.

**Dữ liệu sử dụng:** lesson01_data.csv.

***Bước 1. Đổi tên cột***

```
# Đổi tên cột M1, M2 thành T, V
df.rename(columns={'M1': 'T', 'M2': 'V'}, inplace=True)

# Kiểm tra lại danh sách cột
df.columns
```

***Bước 2. Kiểm tra kiểu dữ liệu***

```
# Kiểu dữ liệu từng cột
df.dtypes

# Tổng số dòng và cột
df.shape

# Kiểm tra thông tin tổng quan
df.info()
```

***Bước 3. Chọn dữ liệu***

```
# Chọn 1 cột
df['T']

# Chọn nhiều cột
df[['T', 'V', 'KV', 'GT']]

# Chọn dòng theo vị trí
df[2:6]          # từ dòng 2 đến 5
df.iloc[2:6]     # tương tự, dùng chỉ số vị trí

# Chọn dòng theo nhãn
df.loc[2:6, ['T', 'KV']]
```

***Bước 4. Lọc dữ liệu theo điều kiện***

```
# Lấy các dòng có T >= 6
df[df['T'] >= 6]

# Kết hợp nhiều điều kiện
```

```
df[(df['T'] > 6) & (df['V'] > 5)]

# Lấy các dòng có GT = 'F'
df.loc[df['GT'] == 'F']
```

**Kết quả mong đợi:**

- Sinh viên thực hành được các thao tác tiền xử lý cơ bản.
- Biết đổi tên cột, chọn subset dữ liệu, và lọc dữ liệu theo điều kiện.
- Đây là kỹ năng nền tảng để xử lý dữ liệu trước khi phân tích hoặc xây dựng mô hình.

### *Thực hành 2-3 Xử lý giá trị thiếu (Missing values)*

**Mục tiêu:** Làm quen với cách phát hiện và xử lý dữ liệu bị thiếu.

***Bước 1. Kiểm tra giá trị thiếu***

```
# Kiểm tra số lượng giá trị thiếu trong mỗi cột
print(df.isnull().sum())
```

***Bước 2. Điền giá trị thiếu***

```
# Điền giá trị trung bình cho cột T
df['T'].fillna(df['T'].mean(), inplace=True)

# Điền giá trị phổ biến nhất cho cột KV
df['KV'].fillna(df['KV'].mode()[0], inplace=True)
```

**Kết quả mong đợi:** Sinh viên biết cách phát hiện giá trị thiếu và áp dụng các chiến lược đơn giản để xử lý.

### *Thực hành 2-4 Phát hiện và xử lý outlier*

**Mục tiêu:** Hiểu cách nhận diện và loại bỏ giá trị bất thường.

***Bước 1. Vẽ boxplot để phát hiện outlier***

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt

sns.boxplot(y=df['T'])
plt.title("Boxplot của điểm T")
plt.show()
```

***Bước 2. Loại bỏ outlier dựa trên IQR***

```
Q1 = df['T'].quantile(0.25)
Q3 = df['T'].quantile(0.75)
IQR = Q3 - Q1

df = df[(df['T'] >= Q1 - 1.5*IQR) & (df['T'] <= Q3 + 1.5*IQR)]
```

**Kết quả mong đợi:** Sinh viên biết phát hiện outlier bằng trực quan hóa và loại bỏ theo IQR.

*Thực hành 2-5* ***Tích hợp dữ liệu từ nhiều nguồn (Merge / Join)***

**Mục tiêu:** Làm quen với việc kết hợp dữ liệu.

**Ví dụ:** Giả sử có thêm bảng df_info chứa thông tin lớp học của học sinh.

```
df_info = pd.DataFrame({
    'ID': [1,2,3,4,5],
    'Lop': ['A','A','B','B','C']
})

# Merge với bảng chính dựa trên ID
df_merged = pd.merge(df, df_info, on='ID', how='left')
print(df_merged.head())
```

**Kết quả mong đợi:** Sinh viên biết sử dụng merge() để kết hợp bảng dữ liệu.

*Thực hành 2-6* ***Trực quan hóa dữ liệu cơ bản***

**Mục tiêu:** Hiển thị phân bố dữ liệu sau khi tiền xử lý.

**Ví dụ: Histogram phân bố điểm**

```

```

```
plt.hist(df['T'], bins=10, color='skyblue', edgecolor='black')
plt.xlabel("Điểm T")
plt.ylabel("Số lượng")
plt.title("Phân bố điểm T")
plt.show()
```

**Kết quả mong đợi:** Sinh viên thấy được sự thay đổi của phân bố dữ liệu sau khi làm sạch.

### *Thực hành 2-7 Tóm tắt và phân tích dữ liệu*

- Mục tiêu:
  - Tổng hợp, thống kê mô tả dữ liệu.
  - Thực hành groupby để phân tích theo nhóm.
  - Tạo pivot table để trình bày dữ liệu tổng hợp.
- Nội dung:

```
# Thống kê mô tả cơ bản
df.aggregate({'T': ['min', 'max'], 'V': ['mean', 'sum']})

# Nhóm dữ liệu theo GT, thống kê số lượng, max, sum
df.groupby('GT')['T'].agg(['count', 'max', 'sum'])

# Nhóm theo GT và KV, thống kê min, max của T và V
df.groupby(['GT', 'KV'])[['T', 'V']].agg(['min', 'max'])

# Pivot table theo GT
pd.pivot_table(df, values=['T', 'V'], columns='GT',
aggfunc=['min', 'mean', 'max'])

# Pivot table nâng cao theo KV và KT
pd.pivot_table(df, values=['M1', 'M2'], columns=['KV', 'KT'],
aggfunc=['min', 'mean', 'max'])
```

- Kết quả:
  - Hiểu cách tổng hợp và phân tích dữ liệu bằng Pandas.
  - Có thể khai thác insight qua groupby và pivot table.
  - Chuẩn bị cho bước phân tích nâng cao hoặc trực quan hóa.

## *2.7.3 Case Study: Phân tích dữ liệu điểm số học tập*

Bối cảnh: Chúng ta có hai bảng dữ liệu điểm số của hai lớp học khác nhau:

- class1.csv: tên, tuổi, điểm toán, điểm lý
- class2.csv: tên, tuổi, điểm toán, điểm hóa

Mục tiêu: tích hợp dữ liệu, xử lý missing values, tạo biến mới và trực quan hóa phân bố điểm.

Bước 1: Đọc dữ liệu

```
import pandas as pd
df1 = pd.read_csv('class1.csv')
df2 = pd.read_csv('class2.csv')

print(df1.head())
print(df2.head())
```

Bước 2: Xử lý missing values

```
df1['toan'].fillna(df1['toan'].mean(), inplace=True)
df2['toan'].fillna(df2['toan'].mean(), inplace=True)
```

Bước 3: Tích hợp dữ liệu

```
df_merged = pd.merge(df1, df2, on=['ten', 'tuoi'],
how='outer')
```

Bước 4: Biến đổi dữ liệu

```
df_merged['tong_diem'] =
df_merged[['toan_x','ly','hoa','toan_y']].sum(axis=1,
skipna=True)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_merged['tong_diem_scaled'] =
scaler.fit_transform(df_merged[['tong_diem']])
```

Bước 5: Trực quan hóa dữ liệu

```
import matplotlib.pyplot as plt

plt.hist(df_merged['tong_diem'], bins=10, color='skyblue')
plt.xlabel('Tổng điểm')
plt.ylabel('Số học sinh')
```

```
plt.title('Phân bố tổng điểm học sinh')
plt.show()
```

### *2.7.4 Bài tập mở rộng*

*Bài tập 2-1* *Tính điểm trung bình cho mỗi học sinh (mean_score) và thêm cột vào dataset.*

*Bài tập 2-2* *Vẽ biểu đồ cột so sánh điểm trung bình theo từng lớp.*

*Bài tập 2-3* *Xác định và loại bỏ các outlier dựa trên tổng điểm, sau đó vẽ lại histogram.*

*Bài tập 2-4* *Thử tạo một biến mới từ các cột hiện có (ví dụ: trung_binh_toan_hoa) và trực quan hóa.*

*Bài tập 2-5* *Tìm một dataset cá nhân (ví dụ chi tiêu, bước đi, hay điểm học tập) và thực hiện đầy đủ các bước tiền xử lý, tích hợp (nếu cần) và trực quan hóa.*

# CHƯƠNG 3 KỸ THUẬT PHÂN TÍCH DỮ LIỆU

Chương này tập trung vào các kỹ thuật phân tích dữ liệu (data analysis techniques) – các phương pháp, quy trình và công cụ giúp biến dữ liệu thô thành thông tin có giá trị, từ đó rút ra các insight hỗ trợ quyết định trong kinh doanh, khoa học, kỹ thuật và nhiều lĩnh vực khác.

Phân tích dữ liệu không chỉ dừng lại ở thống kê cơ bản mà còn bao gồm phân tích mô tả, phân tích chẩn đoán, phân tích dự báo và phân tích đề xuất. Việc hiểu rõ từng loại phân tích, cách áp dụng kỹ thuật phù hợp và sử dụng các công cụ hỗ trợ là bước quan trọng để thực hiện các dự án khoa học dữ liệu hiệu quả.

Mục tiêu của chương:

1. Giải thích được các dạng phân tích dữ liệu và vai trò của từng loại.
2. Thực hiện được các bước cơ bản trong quy trình phân tích dữ liệu.
3. Sử dụng thành thạo các công cụ Python cơ bản để phân tích và trực quan hóa dữ liệu.
4. Rèn luyện khả năng diễn giải kết quả và rút ra insight từ dữ liệu.
5. Thực hành kỹ thuật phân tích dữ liệu trên dữ liệu mẫu thông qua case study.

## 3.1 Vai trò của phân tích dữ liệu

Phân tích dữ liệu (data analysis) đóng vai trò trung tâm trong việc biến dữ liệu thô (raw data) thành thông tin có giá trị, giúp **hỗ trợ quyết định, cải thiện hiệu quả và dự báo tương lai**. Trong thế giới hiện đại, lượng dữ liệu sinh ra hàng ngày là rất lớn, từ giao dịch tài chính, hành vi người dùng trên mạng, đến dữ liệu cảm biến trong sản xuất hay y tế. Nếu không được phân tích đúng cách, dữ liệu chỉ là các con số rời rạc, khó khai thác.

Vai trò chính của phân tích dữ liệu bao gồm:

1. **Hiểu và mô tả hiện trạng** (descriptive analytics): giúp tổ chức nắm bắt bức tranh tổng quan về tình hình hiện tại, ví dụ như doanh số bán hàng theo tháng, phân bố điểm số học sinh hay mức tiêu thụ năng lượng.

2. **Phát hiện nguyên nhân và mối quan hệ** (diagnostic analytics): phân tích các yếu tố ảnh hưởng và tìm hiểu nguyên nhân của các hiện tượng, ví dụ như nguyên nhân giảm doanh số, lý do học sinh điểm thấp môn toán.
3. **Dự báo xu hướng tương lai** (predictive analytics): sử dụng dữ liệu quá khứ để dự đoán kết quả tương lai, ví dụ dự báo nhu cầu sản phẩm, rủi ro tín dụng hay tiến triển bệnh lý.
4. **Đề xuất hành động tối ưu** (prescriptive analytics): đưa ra các khuyến nghị dựa trên mô hình phân tích để ra quyết định tốt hơn, ví dụ lựa chọn chiến lược marketing, phân bổ nguồn lực hay tối ưu hóa lịch trình sản xuất.

Phân tích dữ liệu không chỉ dành cho chuyên gia IT hay nhà khoa học dữ liệu; trong nhiều ngành nghề, khả năng **hiểu dữ liệu và diễn giải kết quả** trở thành kỹ năng quan trọng. Kết hợp với các công cụ Python, việc phân tích dữ liệu trở nên **dễ tiếp cận, linh hoạt và hiệu quả**, giúp tổ chức và cá nhân đưa ra quyết định dựa trên bằng chứng thay vì cảm tính.

## 3.2 Các dạng phân tích dữ liệu

Trong phân tích dữ liệu (data analysis), tùy vào mục tiêu và tính chất của dữ liệu, người ta thường phân loại thành bốn dạng chính: **phân tích mô tả (descriptive analytics), phân tích chẩn đoán (diagnostic analytics), phân tích dự báo (predictive analytics) và phân tích đề xuất (prescriptive analytics)**. Mỗi dạng phục vụ một mục đích khác nhau, nhưng đều giúp biến dữ liệu thô thành thông tin giá trị [6].

### *3.2.1 Phân tích mô tả*

Phân tích mô tả (descriptive analytics) nhằm **tóm tắt và trình bày dữ liệu hiện có** để hiểu bức tranh tổng quan về hiện trạng.

- Ví dụ: thống kê doanh số theo tháng, phân bố tuổi của khách hàng, điểm trung bình của học sinh.
- Công cụ: bảng tổng hợp (pivot table), biểu đồ, thống kê cơ bản (mean, median, mode, standard deviation).

Mục tiêu chính là giúp người dùng **nắm rõ dữ liệu** trước khi thực hiện các phân tích sâu hơn.

### *3.2.2 Phân tích chẩn đoán*

Phân tích chẩn đoán (diagnostic analytics) nhằm **tìm nguyên nhân của các hiện tượng đã xảy ra**.

- Ví dụ: xác định lý do doanh số giảm, nguyên nhân học sinh điểm thấp môn toán, hoặc yếu tố ảnh hưởng đến hiệu suất sản xuất.
- Công cụ: phân tích tương quan (correlation), hồi quy tuyến tính (linear regression), kiểm định thống kê.

Dạng phân tích này giúp người ra quyết định hiểu **tại sao sự kiện xảy ra**, từ đó đưa ra hành động điều chỉnh phù hợp.

### *3.2.3 Phân tích dự báo*

Phân tích dự báo (predictive analytics) sử dụng dữ liệu lịch sử để **dự đoán kết quả tương lai**.

- Ví dụ: dự báo nhu cầu sản phẩm, rủi ro tín dụng, dự đoán tiến triển bệnh lý của bệnh nhân.
- Công cụ: mô hình học máy (machine learning) cơ bản như hồi quy tuyến tính, hồi quy logistic, cây quyết định (decision tree), hoặc các thuật toán nâng cao.

Dự báo giúp tổ chức **lường trước các xu hướng và rủi ro**, từ đó chuẩn bị kế hoạch phù hợp.

### *3.2.4 Phân tích đề xuất*

Phân tích đề xuất (prescriptive analytics) hướng tới **đưa ra khuyến nghị hành động tối ưu** dựa trên kết quả phân tích mô tả, chẩn đoán và dự báo.

- Ví dụ: tối ưu lịch trình sản xuất, phân bổ nguồn lực, đề xuất chiến lược marketing.
- Công cụ: mô phỏng (simulation), tối ưu hóa (optimization), thuật toán học máy nâng cao.

Dạng phân tích này giúp quyết định **không chỉ hiểu vấn đề mà còn biết cách hành động hiệu quả**.

Mỗi dạng phân tích có mối quan hệ tuần tự: thường bắt đầu với **mô tả**, tiếp đến **chẩn đoán**, rồi **dự báo** và cuối cùng là **đề xuất**. Hiểu rõ từng loại phân tích giúp lựa chọn phương pháp và công cụ phù hợp với dữ liệu và mục tiêu thực tế.

## 3.3 Quy trình phân tích dữ liệu

Quy trình phân tích dữ liệu (data analysis process) giúp đảm bảo mọi bước từ **xác định mục tiêu đến diễn giải kết quả** được thực hiện có hệ thống, giảm rủi ro lỗi và tăng chất lượng insight. Quy trình điển hình bao gồm các bước sau:

### *3.3.1 Xác định mục tiêu phân tích*

Trước khi xử lý dữ liệu, cần **xác định rõ vấn đề cần giải quyết** và mục tiêu của phân tích.

- Ví dụ: dự báo doanh số tháng tới, tìm yếu tố ảnh hưởng đến điểm học sinh, hay tối ưu lịch trình sản xuất.
- Mục tiêu rõ ràng giúp chọn **dữ liệu phù hợp, phương pháp phân tích thích hợp và tiêu chí đánh giá kết quả**.

### *3.3.2 Thu thập dữ liệu từ nhiều nguồn*

Dữ liệu có thể đến từ nhiều nguồn: cơ sở dữ liệu, CSV, API, web scraping, cảm biến IoT,…

- Yêu cầu: đảm bảo **đầy đủ, chính xác và hợp pháp**.
- Công cụ hỗ trợ: pandas, SQL, requests, BeautifulSoup,…

### *3.3.3 Làm sạch và chuẩn hóa dữ liệu*

Dữ liệu thô thường **không hoàn hảo**: thiếu giá trị (missing values), lỗi nhập liệu, không đồng nhất.

- Các bước:
    - Xử lý dữ liệu thiếu (fill, drop, interpolate)
    - Loại bỏ outlier (giá trị bất thường)
    - Chuẩn hóa định dạng (standardization, normalization)
    - Mã hóa dữ liệu danh mục (categorical encoding)

### *3.3.4 Phân tích thống kê và trực quan hóa*

- Phân tích thống kê cơ bản: mean, median, mode, min, max, standard deviation.
- Trực quan hóa dữ liệu (visualization): biểu đồ cột, histogram, scatter, boxplot, heatmap,…
- Mục tiêu: **hiểu dữ liệu, phát hiện xu hướng, mối quan hệ và bất thường**.

### *3.3.5 Diễn giải kết quả, rút ra insight*

- Sau khi phân tích, cần **diễn giải kết quả** bằng ngôn ngữ dễ hiểu, kết hợp trực quan hóa.
- Insight phải **gắn với mục tiêu ban đầu** và có thể hỗ trợ quyết định hoặc đề xuất hành động.

### *3.3.6 Chuẩn bị cho mô hình hóa*

- Nếu mục tiêu là **dự báo hoặc tối ưu**, dữ liệu phải được chuẩn bị để xây dựng mô hình học máy hoặc mô hình tối ưu hóa.
- Bước này bao gồm **chia dữ liệu training/test, feature selection, scaling và encoding**.

## 3.4 Kỹ thuật và công cụ chính

Trong phân tích dữ liệu hiện đại, Python là ngôn ngữ phổ biến nhờ thư viện phong phú, cú pháp dễ đọc và khả năng tích hợp cao. Chương này sẽ giới thiệu các kỹ thuật và công cụ chính giúp sinh viên thực hành hiệu quả [2].

### *3.4.1 Python và các thư viện cơ bản*

1. **pandas**:
    - Hỗ trợ thao tác dữ liệu có cấu trúc (dataframe, series).
    - Các chức năng chính: đọc/ghi dữ liệu (CSV, Excel, SQL), xử lý missing values, lọc, nhóm, tổng hợp, merge/join dữ liệu.
2. **numpy**:
    - Thư viện xử lý mảng và tính toán số học hiệu quả.
    - Hữu ích cho các phép toán ma trận, chuẩn hóa, tính toán vector.
3. **matplotlib**:

- Thư viện trực quan hóa cơ bản, vẽ biểu đồ 2D (line, bar, scatter, histogram).
- Dễ kết hợp với pandas để vẽ trực tiếp từ dataframe.

4. **seaborn**:
   - Nâng cao khả năng trực quan hóa so với matplotlib.
   - Hỗ trợ biểu đồ phức tạp như heatmap, violin plot, pairplot, thể hiện mối quan hệ giữa nhiều biến.
5. **scikit-learn** (cho bước mô hình hóa, nếu cần):
   - Hỗ trợ học máy cơ bản: regression, classification, clustering.
   - Bao gồm các công cụ tiền xử lý dữ liệu, feature selection, splitting dữ liệu, đánh giá mô hình.

### *3.4.2 Kỹ thuật thường dùng trong phân tích dữ liệu*

- **Lọc và chọn dữ liệu**: sử dụng pandas để lọc các dòng, cột theo điều kiện, chọn subset dữ liệu.
- **Nhóm và tổng hợp (groupby, pivot table)**: tổng hợp dữ liệu theo nhóm, tính tổng, trung bình, đếm,…
- **Xử lý dữ liệu thiếu và outlier**: điền giá trị thiếu, loại bỏ hoặc thay thế outlier.
- **Chuẩn hóa và biến đổi dữ liệu**: normalization, standardization, encoding dữ liệu danh mục.
- **Trực quan hóa dữ liệu**: từ biểu đồ đơn giản đến biểu đồ phức hợp, heatmap, scatter matrix.
- **Tích hợp dữ liệu**: merge/join nhiều bảng dữ liệu, tạo biến mới từ dữ liệu hiện có.

### *3.4.3 Ứng dụng trong thực hành*

Các kỹ thuật này sẽ được **áp dụng trong các case study nhỏ**, giúp sinh viên:

1. Làm quen với dữ liệu thực tế.
2. Thực hiện quy trình phân tích từ thu thập, làm sạch, biến đổi đến trực quan hóa.

3. Chuẩn bị dữ liệu cho các bước học máy trong các chương sau.

## 3.5 Tóm tắt chương

Trong chương này, sinh viên đã học về **kỹ thuật phân tích dữ liệu** bao gồm:

1. **Vai trò của phân tích dữ liệu:** biến dữ liệu thô thành thông tin có giá trị, hỗ trợ ra quyết định, dự báo và đề xuất hành động.
2. **Các dạng phân tích dữ liệu:**
    - **Mô tả (descriptive analytics):** tóm tắt hiện trạng.
    - **Chẩn đoán (diagnostic analytics):** tìm nguyên nhân.
    - **Dự báo (predictive analytics):** dự đoán tương lai.
    - **Đề xuất (prescriptive analytics):** đưa ra khuyến nghị tối ưu.
3. **Quy trình phân tích dữ liệu:** từ xác định mục tiêu → thu thập → làm sạch → phân tích thống kê & trực quan hóa → diễn giải → chuẩn bị mô hình nếu cần.
4. **Kỹ thuật và công cụ chính:** sử dụng Python và các thư viện như pandas, numpy, matplotlib, seaborn, scikit-learn để xử lý, biến đổi và trực quan hóa dữ liệu.

**Insight:** Hiểu rõ các bước và kỹ thuật phân tích dữ liệu giúp sinh viên áp dụng thực tế, rút ra kết luận chính xác và chuẩn bị dữ liệu cho các bước mô hình hóa và học máy trong các chương sau.

## 3.6 Câu hỏi và Bài tập

### *3.6.1 Câu hỏi ôn tập*

*Câu hỏi 3-1 Phân tích dữ liệu khác gì so với thống kê truyền thống?*

*Câu hỏi 3-2 Nêu bốn dạng phân tích dữ liệu chính và ví dụ minh họa cho mỗi dạng.*

*Câu hỏi 3-3 Quy trình phân tích dữ liệu bao gồm những bước nào?*

*Câu hỏi 3-4 Kể tên ít nhất ba thư viện Python phổ biến dùng trong phân tích dữ liệu và chức năng chính của chúng.*

*Câu hỏi 3-5 Vai trò của trực quan hóa dữ liệu trong phân tích là gì?*

### *3.6.2 Hướng dẫn thực hành*

*Thực hành 3-1* ***Chuẩn bị dữ liệu và khám phá nhanh***

**Mục tiêu:**

- Làm quen với tập dữ liệu bán hàng.
- Kiểm tra cấu trúc, kiểu dữ liệu và thông tin thống kê cơ bản.
- Tạo thói quen khám phá dữ liệu trước khi phân tích sâu.

***Bước 1. Import thư viện và đọc dữ liệu***

```
import pandas as pd

# Đọc file CSV
df = pd.read_csv("sales_data.csv")

# Hiển thị 5 dòng đầu tiên
print(df.head())
```

**Kết quả mong đợi:** 5 dòng dữ liệu đầu tiên với các cột Date, Product, Quantity, Revenue.

***Bước 2. Kiểm tra kích thước và kiểu dữ liệu***

```
# Số dòng và số cột
print("Kích thước dữ liệu:", df.shape)

# Thông tin về kiểu dữ liệu và giá trị null
print(df.info())
```

**Kết quả mong đợi:**

- Thông tin số lượng bản ghi (rows), số cột (columns).
- Kiểu dữ liệu: Date là object (sẽ chuyển về datetime ở bước sau), Product là object, Quantity và Revenue là số.

***Bước 3. Chuyển cột Date sang kiểu thời gian***

```
# Chuyển sang kiểu datetime
df["Date"] = pd.to_datetime(df["Date"])
```

```
# Kiểm tra lại kiểu dữ liệu
print(df.dtypes)
```

***Bước 4. Thống kê mô tả nhanh***

```
# Thống kê mô tả cho cột số
print(df.describe())

# Kiểm tra số lượng từng sản phẩm
print(df["Product"].value_counts())
```

**Kết quả mong đợi:**

- Trung bình, min, max của Quantity và Revenue.
- Số lượng bản ghi của từng sản phẩm.

***Bước 5. Kiểm tra dữ liệu thiếu***

```
print("Số giá trị thiếu trong mỗi cột:")
print(df.isnull().sum())
```

**Kết quả mong đợi:** Thấy toàn bộ số lượng giá trị thiếu (nếu có), chuẩn bị xử lý ở bước tiếp theo.

Như vậy, sinh viên có thể đọc và xem nhanh dữ liệu, nắm được kích thước, kiểu dữ liệu, làm quen với các thao tác kiểm tra dữ liệu thiếu và chuẩn bị cho các bước tiền xử lý dữ liệu tiếp theo.

### *Thực hành 3-2* ***Làm sạch và chuẩn hóa dữ liệu***

**Mục tiêu:**

- Xử lý giá trị thiếu (nếu có).
- Phát hiện và loại bỏ outlier (giá trị bất thường).
- Chuẩn hóa dữ liệu để sẵn sàng cho phân tích.

***Bước 1. Kiểm tra và xử lý giá trị thiếu***

```
# Kiểm tra lại số lượng giá trị thiếu
print(df.isnull().sum())
```

```
# Nếu có giá trị thiếu ở Quantity hoặc Revenue -> điền trung
bình
df["Quantity"].fillna(df["Quantity"].mean(), inplace=True)
df["Revenue"].fillna(df["Revenue"].mean(), inplace=True)

# Nếu có cột Product bị thiếu -> điền giá trị phổ biến nhất
df["Product"].fillna(df["Product"].mode()[0], inplace=True)
```

**Kết quả mong đợi:** Không còn giá trị NaN trong dữ liệu.

***Bước 2. Phát hiện và loại bỏ outlier***

```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot để kiểm tra outlier của Quantity
sns.boxplot(y=df["Quantity"])
plt.title("Boxplot - Quantity")
plt.show()

# Loại bỏ outlier dựa trên IQR
Q1 = df["Quantity"].quantile(0.25)
Q3 = df["Quantity"].quantile(0.75)
IQR = Q3 - Q1

df = df[(df["Quantity"] >= Q1 - 1.5*IQR) & (df["Quantity"] <=
Q3 + 1.5*IQR)]
print("Số dòng sau khi loại outlier:", len(df))
```

**Kết quả mong đợi:** Dữ liệu gọn hơn, loại bỏ các dòng có số lượng bán quá bất thường.

***Bước 3. Chuẩn hóa cột thời gian***

```
# Tạo cột Month (tháng) để tiện phân tích
df["Month"] = df["Date"].dt.to_period("M")

# Kiểm tra kết quả
print(df[["Date", "Month"]].head())
```

***Bước 4. Mã hóa biến phân loại (nếu cần cho mô hình sau này)***

```
# Mã hóa Product thành biến giả (one-hot encoding)
```

```
df_encoded = pd.get_dummies(df, columns=["Product"],
prefix="Prod")

# Xem 5 dòng đầu
print(df_encoded.head())
```

Như vậy, dữ liệu đã sạch, không còn missing values, outlier, đã có cột Month để phân tích theo tháng và sẵn sàng cho phân tích mô tả, groupby, pivot table tiếp theo.

## *Thực hành 3-3 Phân tích mô tả (Descriptive Analytics)*

**Mục tiêu:**

- Tóm tắt dữ liệu theo nhóm, theo thời gian.
- Tạo bảng tổng hợp (pivot table).
- Trực quan hóa kết quả bằng biểu đồ để thấy xu hướng.

### *Bước 1. Tổng hợp doanh thu theo sản phẩm*

```
# Tổng doanh thu và số lượng bán theo từng sản phẩm
product_summary = df.groupby("Product").agg(
    Total_Quantity=("Quantity", "sum"),
    Total_Revenue=("Revenue", "sum"),
    Avg_Quantity=("Quantity", "mean")
).reset_index()

print(product_summary)
```

**Kết quả mong đợi:** Bảng tổng hợp cho thấy sản phẩm nào bán nhiều nhất, doanh thu cao nhất.

### *Bước 2. Tổng hợp doanh thu theo tháng*

```
# Tổng doanh thu theo tháng
monthly_revenue = df.groupby("Month")["Revenue"].sum()
print(monthly_revenue)
```

### *Bước 3. Tạo bảng tổng hợp (pivot table)*

```
pivot_table = df.pivot_table(
    values="Revenue",
    index="Month",
```

```
    columns="Product",
    aggfunc="sum"
)

print(pivot_table)
```

**Giải thích:**

- Mỗi hàng là một tháng.
- Mỗi cột là một sản phẩm.
- Giá trị trong ô là tổng doanh thu.

***Bước 4. Vẽ biểu đồ so sánh doanh thu theo sản phẩm***

```
import matplotlib.pyplot as plt

product_summary.plot(
    x="Product",
    y="Total_Revenue",
    kind="bar",
    legend=False,
    color="skyblue"
)
plt.title("Tổng doanh thu theo sản phẩm")
plt.ylabel("Doanh thu")
plt.show()
```

***Bước 5. Vẽ biểu đồ doanh thu theo tháng***

```
monthly_revenue.plot(kind="line", marker="o")
plt.title("Doanh thu theo tháng")
plt.ylabel("Doanh thu")
plt.show()
```

***Bước 6. Biểu đồ heatmap cho pivot table***

```
import seaborn as sns

sns.heatmap(pivot_table, annot=True, fmt=".0f", cmap="YlGnBu")
plt.title("Doanh thu theo tháng và sản phẩm")
plt.ylabel("Tháng")
plt.xlabel("Sản phẩm")
```

```
plt.show()
```

Vậy, sinh viên đã biết dùng groupby, agg, pivot_table để tóm tắt dữ liệu; thực hành trực quan hóa dữ liệu với bar chart, line chart, heatmap, và nắm rõ cách trả lời câu hỏi: "Sản phẩm nào bán chạy?", "Tháng nào doanh thu cao nhất?".

## *Thực hành 3-4 Phân tích chẩn đoán (Diagnostic Analytics)*

**Mục tiêu:**

- Phát hiện mối tương quan giữa các biến số.
- Hiểu yếu tố nào ảnh hưởng nhiều đến doanh thu.
- Làm quen với hồi quy tuyến tính đơn giản để kiểm tra mối quan hệ nhân quả.

### *Bước 1. Kiểm tra tương quan giữa các biến số*

```
# Ma trận tương quan
corr_matrix = df[["Quantity", "Revenue"]].corr()
print(corr_matrix)
```

### *Bước 2. Trực quan hóa tương quan bằng heatmap*

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(corr_matrix, annot=True, cmap="coolwarm",
center=0)
plt.title("Ma trận tương quan giữa Quantity và Revenue")
plt.show()
```

**Giải thích:**

- Hệ số tương quan gần 1 → hai biến có mối quan hệ tuyến tính chặt chẽ.
- Gần 0 → ít hoặc không có tương quan.

### *Bước 3. Phân tích mối quan hệ Quantity – Revenue bằng scatter plot*

```
plt.scatter(df["Quantity"], df["Revenue"], alpha=0.5)
plt.xlabel("Quantity")
plt.ylabel("Revenue")
plt.title("Mối quan hệ giữa Quantity và Revenue")
```

```
plt.show()
```

***Bước 4. Xây dựng mô hình hồi quy tuyến tính đơn giản***

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = df[["Quantity"]]  # Biến độc lập
y = df["Revenue"]     # Biến phụ thuộc

model = LinearRegression()
model.fit(X, y)

print("Hệ số góc (slope):", model.coef_[0])
print("Hệ số chặn (intercept):", model.intercept_)
print("R^2 score:", model.score(X, y))
```

**Giải thích:**

- **Slope**: cho biết trung bình doanh thu tăng bao nhiêu khi bán thêm 1 đơn vị sản phẩm.
- **Intercept**: doanh thu dự đoán khi số lượng = 0 (thường gần 0).
- **$R^2$ score**: mức độ giải thích biến thiên doanh thu bởi số lượng (gần 1 là mô hình tốt).

***Bước 5. Vẽ đường hồi quy trên scatter plot***

```
y_pred = model.predict(X)
plt.scatter(df["Quantity"], df["Revenue"], alpha=0.5,
label="Dữ liệu thực tế")
plt.plot(df["Quantity"], y_pred, color="red", linewidth=2,
label="Đường hồi quy")
plt.xlabel("Quantity")
plt.ylabel("Revenue")
plt.title("Hồi quy tuyến tính: Quantity vs Revenue")
plt.legend()
plt.show()
```

Đến đây, sinh viên biết cách tìm mối tương quan giữa biến số, làm quen với hồi quy tuyến tính đơn giản và có thể giải thích mối quan hệ nguyên nhân – kết quả ở mức cơ bản.

## *Thực hành 3-5 Phân tích dự báo (Predictive Analytics)*

**Mục tiêu:**

- Sử dụng dữ liệu quá khứ để dự đoán doanh thu tương lai.
- Làm quen với mô hình hồi quy tuyến tính theo thời gian.
- Minh họa cách đánh giá độ chính xác dự báo.

***Bước 1. Chuẩn bị dữ liệu theo tháng***

```
# Tổng doanh thu theo tháng
monthly_df =
df.groupby("Month")["Revenue"].sum().reset_index()

# Đổi Month về dạng datetime (giá trị giữa tháng)
monthly_df["Month"] = monthly_df["Month"].dt.to_timestamp()

print(monthly_df)
```

***Bước 2. Tạo biến số thời gian để làm đầu vào mô hình***

```
# Chuyển thời gian thành số thứ tự (0,1,2,...)
monthly_df["t"] = range(len(monthly_df))
```

***Bước 3. Chia dữ liệu Train/Test***

```
from sklearn.model_selection import train_test_split

X = monthly_df[["t"]]
y = monthly_df["Revenue"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=1/3, shuffle=False)
```

***Bước 4. Huấn luyện mô hình hồi quy tuyến tính***

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
print("Hệ số góc:", model.coef_[0])
print("Hệ số chặn:", model.intercept_)
```

***Bước 5. Dự báo và đánh giá***

```
import matplotlib.pyplot as plt

y_pred = model.predict(X_test)

plt.plot(monthly_df["Month"], monthly_df["Revenue"],
label="Doanh thu thực tế", marker="o")
plt.plot(X_test.index, y_pred, label="Dự báo", marker="x",
linestyle="--")
plt.title("Dự báo doanh thu theo tháng")
plt.ylabel("Doanh thu")
plt.legend()
plt.show()

from sklearn.metrics import mean_squared_error, r2_score
print("MSE:", mean_squared_error(y_test, y_pred))
print("R²:", r2_score(y_test, y_pred))
```

**Giải thích:**

- **MSE** càng nhỏ càng tốt.
- **R²** càng gần 1 càng tốt (mô hình giải thích tốt dữ liệu).

***Bước 6. Dự báo tháng tiếp theo***

```
next_t = np.array([[monthly_df["t"].max() + 1]])
next_month_prediction = model.predict(next_t)
print("Dự báo doanh thu tháng tiếp theo:",
next_month_prediction[0])
```

Sinh viên đã biết cách chuyển dữ liệu thành chuỗi thời gian, huấn luyện mô hình dự báo đơn giản, có thể trực quan so sánh giá trị thực tế và giá trị dự báo và thấy cách mô hình giúp ước lượng doanh thu tương lai.

### *Thực hành 3-6* ***Phân tích đề xuất (Prescriptive Analytics)***

**Mục tiêu:**

- Dựa trên kết quả phân tích mô tả, chẩn đoán, dự báo để đưa ra đề xuất hành động.
- Minh họa cách tối ưu hóa quyết định bằng công cụ Python.

***Bước 1. Xác định vấn đề cần tối ưu***

Ví dụ: *Doanh nghiệp có ngân sách marketing giới hạn, muốn phân bổ sao cho tổng doanh thu dự kiến cao nhất.*

Giả định:

- Ngân sách marketing = **100 đơn vị**.
- Nếu tăng ngân sách cho sản phẩm nào, số lượng bán dự kiến tăng tỉ lệ thuận theo **hệ số lợi nhuận biên** (marginal revenue).

***Bước 2. Tính hệ số lợi nhuận biên cho từng sản phẩm***

```
# Tính doanh thu trung bình mỗi đơn vị bán
product_summary["Revenue_per_unit"] = (
    product_summary["Total_Revenue"] /
product_summary["Total_Quantity"]
)
print(product_summary[["Product", "Revenue_per_unit"]])
```

***Bước 3. Thiết lập mô hình tối ưu hóa***

Dùng thư viện pulp để giải bài toán phân bổ ngân sách:

```
!pip install pulp  # chạy 1 lần nếu chưa cài

import pulp

# Biến quyết định: ngân sách phân bổ cho từng sản phẩm (0–100)
products = product_summary["Product"].tolist()
budget = 100

# Khởi tạo bài toán tối ưu
prob = pulp.LpProblem("Marketing_Optimization",
pulp.LpMaximize)

# Biến quyết định
alloc = pulp.LpVariable.dicts("Budget", products, lowBound=0)
```

```
# Hàm mục tiêu: tối đa hóa doanh thu dự kiến
prob += pulp.lpSum([alloc[p] *
product_summary.loc[product_summary["Product"]==p,
"Revenue_per_unit"].values[0] for p in products])

# Ràng buộc: tổng ngân sách không vượt quá 100
prob += pulp.lpSum([alloc[p] for p in products]) <= budget

# Giải bài toán
prob.solve()
for p in products:
    print(f"Phân bổ cho {p}: {alloc[p].value()}")
```

***Bước 4. Diễn giải kết quả***

- Sinh viên so sánh kết quả tối ưu với phân bổ hiện tại.
- Thảo luận vì sao nên dồn ngân sách cho sản phẩm có doanh thu biên cao.

***Bước 5. Đề xuất hành động***

- Ưu tiên sản phẩm có doanh thu biên cao để tăng lợi nhuận.
- Điều chỉnh chiến lược marketing cho sản phẩm có doanh thu thấp nhưng tiềm năng tăng trưởng (theo kết quả dự báo ở 3.5).
- Theo dõi kết quả sau khi áp dụng và lặp lại phân tích định kỳ.

Sinh viên đã biết cách kết hợp kết quả phân tích, dự báo, tối ưu hóa để ra quyết định; làm quen với bài toán tối ưu đơn giản bằng Python, và hiểu tư duy prescriptive analytics: không chỉ trả lời "Điều gì sẽ xảy ra?" mà còn "Nên làm gì?".

### *3.6.3 Case Study – Phân tích dữ liệu bán hàng*

**Bối cảnh:** Một cửa hàng bán lẻ muốn phân tích dữ liệu bán hàng trong ba tháng gần đây để đánh giá hiệu quả kinh doanh và lên kế hoạch cải thiện doanh số. Ban quản lý kỳ vọng việc phân tích sẽ giúp họ hiểu rõ tình hình bán hàng theo từng sản phẩm, nhận diện xu hướng theo thời gian và xác định các cơ hội tăng trưởng.

Dữ liệu được lưu trong tệp sales_data.csv, bao gồm các cột:

- **Date**: ngày bán hàng.
- **Product**: mã sản phẩm (A, B, C, D).
- **Quantity**: số lượng bán ra trong ngày.

- **Revenue**: doanh thu thu được từ sản phẩm (Quantity × đơn giá).

Phân tích dữ liệu này sẽ giúp cửa hàng đưa ra quyết định marketing và quản lý tồn kho hiệu quả hơn, từ đó tối ưu hóa lợi nhuận trong các kỳ kinh doanh tiếp theo.

```
# Bước 1: Đọc dữ liệu và quan sát sơ bộ
import pandas as pd

df = pd.read_csv('sales_data.csv')
df['Date'] = pd.to_datetime(df['Date'])  # Chuyển sang
datetime

print(df.head())
print(df.info())
print(df.describe())

# Bước 2: Xử lý dữ liệu thiếu và outlier
df['Revenue'].fillna(df['Revenue'].mean(), inplace=True)
mean = df['Revenue'].mean()
std = df['Revenue'].std()
df = df[df['Revenue'] < mean + 3*std]  # loại bỏ giá trị quá
lớn

# Bước 3: Biến đổi dữ liệu
# Tổng doanh thu theo sản phẩm
df_grouped =
df.groupby('Product')['Revenue'].mean().reset_index()
df_grouped.rename(columns={'Revenue': 'Average_Revenue'},
inplace=True)
print(df_grouped)

# Bổ sung doanh thu theo tháng để tìm mùa vụ
df['Month'] = df['Date'].dt.to_period('M')
monthly_revenue =
df.groupby('Month')['Revenue'].sum().reset_index()
print(monthly_revenue)

# Bước 4: Trực quan hóa dữ liệu
import matplotlib.pyplot as plt
import seaborn as sns

# Bar chart doanh thu trung bình theo sản phẩm
sns.barplot(x='Product', y='Average_Revenue', data=df_grouped)
plt.title('Doanh thu trung bình theo sản phẩm')
plt.show()

# Line chart doanh thu theo tháng
```

```
sns.lineplot(x='Month', y='Revenue', data=monthly_revenue,
marker='o')
plt.title('Doanh thu theo tháng')
plt.show()

# Bước 5: Diễn giải kết quả
# (viết tự luận)
# - Sản phẩm nào có doanh thu trung bình cao nhất?
# - Tháng nào doanh thu cao nhất/thấp nhất?
# - Đề xuất tăng sản lượng hoặc marketing cho sản phẩm chủ
lực.
```

### *3.6.4 Bài tập mở rộng*

*Bài tập 3-1* *Tính tổng doanh thu theo tháng, vẽ biểu đồ line chart thể hiện xu hướng doanh thu theo thời gian.*

- Gợi ý: dùng groupby("Month")["Revenue"].sum() và plot(kind="line").
- Nhận xét: tháng nào doanh thu cao nhất, thấp nhất, xu hướng tăng hay giảm?

*Bài tập 3-2* *Xác định sản phẩm bán chạy nhất (theo tổng số lượng bán) và sản phẩm kém hiệu quả nhất.*

- Viết báo cáo ngắn (5–7 dòng) giải thích kết quả và gợi ý hành động kinh doanh.
- Gợi ý: dùng groupby("Product")["Quantity"].sum().

*Bài tập 3-3* *Tạo biến mới:* ***Doanh thu trung bình mỗi đơn bán*** *= Revenue / Quantity.*

- Vẽ biểu đồ boxplot để so sánh giữa các sản phẩm.
- Thảo luận: sản phẩm nào có đơn giá cao nhất, sản phẩm nào có biên lợi nhuận thấp nhất?

*Bài tập 3-4* *Thực hiện phân tích tương tự trên một dataset khác mà bạn tự tìm kiếm (open dataset từ Kaggle, Google Dataset Search hoặc dữ liệu của chính bạn).*

- Yêu cầu: phải thực hiện ít nhất 3 bước gồm: làm sạch dữ liệu, phân tích mô tả, trực quan hóa.
- Chuẩn bị báo cáo ngắn (1–2 trang) tóm tắt insight rút ra được.

# CHƯƠNG 4 KHÁM PHÁ VÀ TRỰC QUAN HÓA DỮ LIỆU

Chương này tập trung vào **khám phá dữ liệu (exploratory data analysis – EDA)**, một bước quan trọng trong phân tích dữ liệu nhằm **hiểu rõ đặc điểm, phân phối và mối quan hệ giữa các biến**. EDA giúp phát hiện dữ liệu bất thường, xu hướng tiềm ẩn và tiền đề cho các bước mô hình hóa hoặc dự báo.

Ngoài ra, chương cũng giới thiệu **trực quan hóa dữ liệu (data visualization)**, sử dụng các biểu đồ để truyền tải thông tin một cách trực quan, dễ hiểu và hỗ trợ ra quyết định. Các thư viện Python như **matplotlib** và **seaborn** sẽ được áp dụng để sinh viên thực hành trực tiếp trên dữ liệu mẫu.

Sau khi học xong chương này, sinh viên sẽ có khả năng:

1. Hiểu rõ vai trò và ý nghĩa của khám phá dữ liệu (EDA) trong phân tích dữ liệu.
2. Sử dụng các **thống kê mô tả** để tóm tắt dữ liệu cơ bản.
3. Trực quan hóa dữ liệu bằng **matplotlib** và **seaborn**, từ biểu đồ cơ bản đến nâng cao.
4. Phát hiện mối quan hệ giữa các biến và nhận diện dữ liệu bất thường.
5. Chuẩn bị dữ liệu hiệu quả cho các bước mô hình hóa và học máy sau này.

## 4.1 Khái niệm và vai trò của EDA

Khám phá dữ liệu (exploratory data analysis – EDA) là bước **phân tích ban đầu nhằm hiểu rõ dữ liệu trước khi áp dụng các mô hình phức tạp**. Thông qua EDA, người phân tích có thể nắm bắt **cấu trúc, phân phối, xu hướng, và mối quan hệ giữa các biến**, đồng thời phát hiện các **giá trị bất thường (outlier)** hoặc dữ liệu thiếu (missing values).

Vai trò của EDA trong quy trình phân tích dữ liệu rất quan trọng:

- Giúp **định hướng phân tích** bằng cách chỉ ra các biến quan trọng và các vấn đề tiềm ẩn.

- Hỗ trợ **quyết định kỹ thuật tiền xử lý dữ liệu** như loại bỏ outlier, điền giá trị thiếu, chuẩn hóa hay biến đổi dữ liệu.
- Là cơ sở để **chọn mô hình phân tích hoặc học máy phù hợp**, tránh áp dụng mô hình không thích hợp với đặc điểm dữ liệu.
- Cung cấp cái nhìn **trực quan về dữ liệu**, giúp người dùng, quản lý hay khách hàng dễ dàng hiểu thông tin và ra quyết định.

EDA thường kết hợp **thống kê mô tả (descriptive statistics)** và **trực quan hóa dữ liệu (data visualization)**, tạo nền tảng vững chắc cho mọi phân tích sâu hơn.

## 4.2 Thống kê mô tả

Thống kê mô tả (descriptive statistics) là **công cụ cơ bản trong EDA** dùng để **tóm tắt và mô tả các đặc điểm chính của dữ liệu** mà không đưa ra suy luận phức tạp. Thống kê mô tả giúp người phân tích nhanh chóng hiểu **phân phối, xu hướng trung tâm và sự biến thiên** của dữ liệu.

Các chỉ số phổ biến bao gồm:

1. **Trung bình (mean):** giá trị trung bình của tập dữ liệu.
2. **Trung vị (median):** giá trị nằm ở giữa khi dữ liệu được sắp xếp.
3. **Mode:** giá trị xuất hiện nhiều nhất trong tập dữ liệu.
4. **Giá trị lớn nhất và nhỏ nhất (max, min):** xác định phạm vi dữ liệu.
5. **Khoảng biến thiên (range):** chênh lệch giữa giá trị lớn nhất và nhỏ nhất.
6. **Độ lệch chuẩn (standard deviation – std):** đo lường sự phân tán quanh trung bình.
7. **Phân vị (percentiles/quartiles):** chia dữ liệu thành các phần bằng nhau để đánh giá phân bố.

**Ví dụ thực hành với Python**

```
import pandas as pd

# Tạo dataframe mẫu
data = {'san_pham': ['A','B','C','D','E'],
        'doanh_thu': [1000, 1500, 800, 2000, 1200],
        'so_luong': [10, 15, 8, 20, 12]}
```

```
df = pd.DataFrame(data)

# Thống kê cơ bản
print(df.describe())

# Tính trung bình doanh thu
print("Trung bình doanh thu:", df['doanh_thu'].mean())

# Tính trung vị số lượng
print("Trung vị số lượng:", df['so_luong'].median())
```

**Giải thích:**

- df.describe() trả về tổng quan: count, mean, std, min, 25%, 50%, 75%, max cho mỗi cột số.
- Hàm mean() và median() giúp tính nhanh các chỉ số trung tâm.

## 4.3 Trực quan hóa dữ liệu với Matplotlib

Trực quan hóa dữ liệu (data visualization) là **cách biểu diễn dữ liệu bằng hình ảnh**, giúp người phân tích dễ dàng **nhận diện xu hướng, mối quan hệ giữa các biến và các giá trị bất thường**. Trong Python, **Matplotlib** là thư viện cơ bản và mạnh mẽ, hỗ trợ tạo nhiều loại biểu đồ khác nhau từ đơn giản đến nâng cao [14].

**Các loại biểu đồ phổ biến**

1. **Line chart (biểu đồ đường):** dùng để thể hiện xu hướng thay đổi theo thời gian.
2. **Bar chart (biểu đồ cột):** so sánh giá trị giữa các nhóm.
3. **Histogram (biểu đồ tần suất):** phân phối dữ liệu.
4. **Scatter plot (biểu đồ điểm):** phân tích mối quan hệ giữa hai biến.
5. **Pie chart (biểu đồ tròn):** tỷ lệ phần trăm trong tổng thể.

**Ví dụ thực hành với Matplotlib**

```
import matplotlib.pyplot as plt
```

```
# Dữ liệu mẫu
san_pham = ['A','B','C','D','E']
doanh_thu = [1000, 1500, 800, 2000, 1200]

# Biểu đồ cột
plt.bar(san_pham, doanh_thu, color='skyblue')
plt.xlabel('Sản phẩm')
plt.ylabel('Doanh thu')
plt.title('Doanh thu theo sản phẩm')
plt.show()

# Biểu đồ đường
thang = [1,2,3,4,5]
doanh_thu_thang = [1000, 1200, 1500, 1300, 1600]

plt.plot(thang, doanh_thu_thang, marker='o', linestyle='-',
color='green')
plt.xlabel('Tháng')
plt.ylabel('Doanh thu')
plt.title('Doanh thu theo tháng')
plt.show()
```

**Giải thích:**

- plt.bar() tạo biểu đồ cột, trực quan so sánh giá trị từng sản phẩm.
- plt.plot() tạo biểu đồ đường, thể hiện xu hướng theo thời gian.
- Các tham số như color, marker, linestyle giúp tùy chỉnh màu sắc và kiểu biểu diễn.

## 4.4 Trực quan hóa nâng cao với Seaborn

Seaborn là thư viện trực quan hóa dữ liệu dựa trên Matplotlib, cung cấp **giao diện cao cấp, dễ sử dụng và đẹp mắt hơn**, đặc biệt khi phân tích mối quan hệ giữa nhiều biến. Seaborn hỗ trợ trực tiếp các **biểu đồ thống kê**, giúp khám phá dữ liệu nhanh và hiệu quả.

**Các loại biểu đồ phổ biến trong Seaborn**

1. **Bar plot:** so sánh giá trị trung bình theo nhóm, kèm khoảng tin cậy (confidence interval).
2. **Box plot (biểu đồ hộp):** hiển thị phân bố, trung vị, quartiles và outlier.

3. **Violin plot:** kết hợp box plot và density plot, cho thấy phân phối dữ liệu.
4. **Scatter plot với hue:** phân tích mối quan hệ giữa hai biến, phân loại theo nhóm.
5. **Heatmap:** biểu diễn ma trận dữ liệu (ví dụ ma trận tương quan) bằng màu sắc.

**Ví dụ thực hành với Seaborn**

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Dữ liệu mẫu
data = {'san_pham': ['A','B','C','D','E'],
        'doanh_thu': [1000, 1500, 800, 2000, 1200],
        'so_luong': [10, 15, 8, 20, 12]}

df = pd.DataFrame(data)

# Bar plot
sns.barplot(x='san_pham', y='doanh_thu', data=df,
palette='Blues_d')
plt.title('Doanh thu theo sản phẩm')
plt.show()

# Box plot
sns.boxplot(y='doanh_thu', data=df)
plt.title('Phân bố doanh thu')
plt.show()

# Heatmap ma trận tương quan
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Ma trận tương quan')
plt.show()
```

**Giải thích:**

- sns.barplot() cho phép so sánh giá trị trung bình theo nhóm với màu sắc đẹp mắt.
- sns.boxplot() giúp phát hiện outlier và hiểu phân phối dữ liệu.
- sns.heatmap() hiển thị mối quan hệ giữa các biến số, thuận tiện để chọn biến quan trọng trong phân tích.

## 4.5 Phân tích quan hệ giữa các biến

Trong EDA, phân tích mối quan hệ giữa các biến (variable relationships) là bước quan trọng để hiểu tương quan, ảnh hưởng lẫn nhau và phát hiện xu hướng tiềm ẩn. Phân tích này giúp chọn biến đầu vào phù hợp cho mô hình học máy hoặc các phân tích nâng cao [3].

**Các loại quan hệ thường gặp**

1. **Quan hệ tuyến tính (linear relationship):** hai biến thay đổi theo tỷ lệ tương ứng.
2. **Quan hệ phi tuyến (non-linear relationship):** biến thay đổi không theo tỷ lệ cố định.
3. **Biến phân loại – biến số (categorical vs numerical):** dùng box plot, violin plot để so sánh phân phối theo nhóm.
4. **Biến số – biến số (numerical vs numerical):** dùng scatter plot, heatmap tương quan.
5. **Biến phân loại – biến phân loại (categorical vs categorical):** dùng cross-tab, stacked bar chart.

**Ví dụ thực hành với Python và Seaborn**

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Dữ liệu mẫu
data = {'san_pham': ['A','B','C','D','E'],
        'doanh_thu': [1000, 1500, 800, 2000, 1200],
        'so_luong': [10, 15, 8, 20, 12],
        'loai_san_pham': ['X','Y','X','Y','X']}

df = pd.DataFrame(data)

# Scatter plot: doanh thu vs số lượng
sns.scatterplot(x='so_luong', y='doanh_thu', data=df,
hue='loai_san_pham', s=100)
plt.title('Doanh thu theo số lượng và loại sản phẩm')
plt.show()

# Box plot: doanh thu theo loại sản phẩm
```

```
sns.boxplot(x='loai_san_pham', y='doanh_thu', data=df)
plt.title('Phân phối doanh thu theo loại sản phẩm')
plt.show()

# Heatmap tương quan giữa các biến số
corr = df[['doanh_thu', 'so_luong']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Ma trận tương quan doanh thu và số lượng')
plt.show()
```

**Giải thích:**

- Scatter plot với hue giúp quan sát mối quan hệ giữa hai biến số, đồng thời phân loại theo nhóm.
- Box plot minh họa phân bố theo nhóm, giúp phát hiện outlier và xu hướng khác nhau giữa các loại sản phẩm.
- Heatmap trực quan hóa ma trận tương quan, nhận diện biến nào có ảnh hưởng mạnh tới biến mục tiêu.

## 4.6 Tóm tắt chương

Chương này tập trung vào **khám phá và trực quan hóa dữ liệu (EDA – exploratory data analysis)**, là bước quan trọng giúp người phân tích:

1. **Hiểu rõ đặc điểm dữ liệu:** thông qua thống kê mô tả (mean, median, std, min, max, percentiles) để nắm phân phối, xu hướng trung tâm và sự biến thiên của dữ liệu.
2. **Nhận diện dữ liệu bất thường:** phát hiện giá trị thiếu (missing values) và outlier, từ đó quyết định các bước tiền xử lý phù hợp.
3. **Trực quan hóa dữ liệu:** sử dụng Matplotlib và Seaborn để biểu diễn dữ liệu bằng biểu đồ đường, cột, scatter, box, violin và heatmap, giúp dễ dàng phát hiện xu hướng và mối quan hệ giữa các biến.
4. **Phân tích quan hệ giữa các biến:** xác định tương quan, ảnh hưởng lẫn nhau giữa các biến số và biến phân loại, hỗ trợ việc chọn biến và mô hình hóa sau này.

Tóm lại, EDA và trực quan hóa là **nền tảng không thể thiếu trong phân tích dữ liệu**, giúp chuyển dữ liệu thô thành thông tin có giá trị, đồng thời chuẩn bị cho các bước phân tích nâng cao hoặc mô hình học máy.

## 4.7 Câu hỏi và Bài tập

### *4.7.1 Câu hỏi ôn tập*

*Câu hỏi 4-1 EDA là gì và vai trò của nó trong phân tích dữ liệu ra sao?*

*Câu hỏi 4-2 Nêu các chỉ số thống kê mô tả cơ bản và ý nghĩa của từng chỉ số.*

*Câu hỏi 4-3 Khác biệt giữa scatter plot, box plot và heatmap là gì?*

*Câu hỏi 4-4 Khi nào bạn nên sử dụng violin plot thay vì box plot?*

*Câu hỏi 4-5 Tại sao phân tích mối quan hệ giữa các biến lại quan trọng trước khi xây dựng mô hình học máy?*

### *4.7.2 Hướng dẫn thực hành*

*Thực hành 4-1* ***Khám phá dữ liệu khách hàng***

**Mục tiêu:**

- Làm quen với tập dữ liệu khách hàng và đơn hàng.
- Kiểm tra cấu trúc, kiểu dữ liệu, số lượng bản ghi.
- Tóm tắt thống kê mô tả để hiểu đặc điểm của khách hàng và hành vi chi tiêu.

***Bước 1. Đọc dữ liệu và xem trước***

```
import pandas as pd

df = pd.read_csv("customer_orders.csv")

# Hiển thị 5 dòng đầu
print(df.head())
```

Sinh viên quan sát cấu trúc dataset: CustomerID, Age, Gender, OrderDate, TotalSpent, Region.

***Bước 2. Kiểm tra thông tin tổng quát***

```
print("Kích thước dữ liệu:", df.shape)
print(df.info())
```

Sinh viên biết số lượng bản ghi (rows) và cột (columns), kiểu dữ liệu từng cột.

***Bước 3. Chuyển đổi kiểu dữ liệu ngày tháng***

```
df["OrderDate"] = pd.to_datetime(df["OrderDate"])
print(df.dtypes)
```

Mục đích: Đảm bảo cột OrderDate ở dạng datetime để phân tích theo tháng, quý.

***Bước 4. Thống kê mô tả cho biến số***

```
print(df.describe())
```

Sinh viên quan sát min, max, mean, std của Age và TotalSpent, phát hiện độ trải rộng dữ liệu.

***Bước 5. Kiểm tra phân phối biến phân loại***

```
print("Phân bố giới tính:")
print(df["Gender"].value_counts())

print("\nPhân bố vùng miền:")
print(df["Region"].value_counts())
```

Mục đích: Giúp hiểu dữ liệu có cân bằng hay không, có vùng miền nào áp đảo.

Qua bài thực hành này, sinh viên đã đọc và kiểm tra được dữ liệu khách hàng, biết được phân bố tuổi, giới tính, vùng miền, chi tiêu trung bình và sẵn sàng bước sang xử lý dữ liệu thiếu, phát hiện outlier.

## *Thực hành 4-2* ***Kiểm tra dữ liệu thiếu và phát hiện outlier***

**Mục tiêu:**

- Kiểm tra dữ liệu thiếu (missing values).
- Xử lý giá trị thiếu bằng phương pháp đơn giản.
- Phát hiện và loại bỏ các giá trị bất thường (outlier) trong TotalSpent.

***Bước 1. Kiểm tra dữ liệu thiếu***

```
# Đếm số lượng giá trị thiếu trong từng cột
print(df.isnull().sum())
```

Nếu dataset được tạo đủ sạch, kết quả thường là 0.

***Bước 2. Điền giá trị thiếu (nếu có)***

```
# Điền giá trị thiếu trong Age bằng giá trị trung bình
df["Age"].fillna(df["Age"].mean(), inplace=True)

# Điền giá trị thiếu trong TotalSpent bằng giá trị trung vị
df["TotalSpent"].fillna(df["TotalSpent"].median(),
inplace=True)

# Điền giá trị thiếu trong Region bằng giá trị phổ biến nhất
df["Region"].fillna(df["Region"].mode()[0], inplace=True)
```

Kỹ thuật này giúp dữ liệu đủ đầy để phân tích tiếp theo.

***Bước 3. Phát hiện outlier bằng boxplot***

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(y=df["TotalSpent"])
plt.title("Boxplot - Tổng chi tiêu (TotalSpent)")
plt.show()
```

Quan sát xem có điểm nào nằm ngoài whisker → đó là outlier.

***Bước 4. Loại bỏ outlier bằng IQR***

```
Q1 = df["TotalSpent"].quantile(0.25)
Q3 = df["TotalSpent"].quantile(0.75)
IQR = Q3 - Q1

df = df[(df["TotalSpent"] >= Q1 - 1.5*IQR) & (df["TotalSpent"]
<= Q3 + 1.5*IQR)]
print("Số bản ghi sau khi loại bỏ outlier:", len(df))
```

Sau bước này, dữ liệu sẽ "sạch" hơn, ít bị méo khi tính trung bình.

***Bước 5. Kiểm tra lại thống kê mô tả***

```
print(df["TotalSpent"].describe())
```

Quan sát sự thay đổi của mean, max → xem việc loại bỏ outlier có hợp lý.

Qua bài thực hành này, sinh viên đã biết cách kiểm tra dữ liệu thiếu và xử lý bằng các phương pháp đơn giản, đã phát hiện và loại bỏ outlier bằng boxplot và IQR và dữ liệu đã sẵn sàng cho bước trực quan hóa và phân tích mối quan hệ ở các bài sau.

## *Thực hành 4-3 Trực quan hóa dữ liệu cơ bản*

**Mục tiêu:**

- Vẽ biểu đồ cột để so sánh tổng chi tiêu theo vùng.
- Vẽ histogram để quan sát phân phối chi tiêu.
- Vẽ line chart để theo dõi xu hướng chi tiêu theo thời gian.

***Bước 1. Bar chart – Tổng chi tiêu theo vùng***

```
import matplotlib.pyplot as plt
import seaborn as sns

# Tổng chi tiêu theo Region
region_summary =
df.groupby("Region")["TotalSpent"].sum().reset_index()

sns.barplot(x="Region", y="TotalSpent", data=region_summary,
palette="Blues_d")
plt.title("Tổng chi tiêu theo vùng")
plt.ylabel("Tổng chi tiêu")
plt.show()
```

Dùng để so sánh vùng nào mang lại doanh thu cao nhất.

***Bước 2. Histogram – Phân phối chi tiêu***

```
plt.hist(df["TotalSpent"], bins=20, color="skyblue",
edgecolor="black")
plt.title("Phân phối tổng chi tiêu")
plt.xlabel("Tổng chi tiêu")
plt.ylabel("Số khách hàng")
plt.show()
```

Quan sát xem phân phối lệch phải hay cân đối, có nhiều khách hàng chi tiêu cao hay không.

***Bước 3. Line chart – Xu hướng chi tiêu theo thời gian***

```
# Tính tổng chi tiêu theo ngày
daily_spending =
df.groupby("OrderDate")["TotalSpent"].sum().reset_index()

plt.plot(daily_spending["OrderDate"],
daily_spending["TotalSpent"], marker="o")
plt.title("Xu hướng chi tiêu theo ngày")
plt.xlabel("Ngày")
plt.ylabel("Tổng chi tiêu")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Giúp phát hiện xu hướng tăng/giảm theo thời gian, mùa vụ hay đột biến.

***Bước 4. Biểu đồ cột so sánh theo giới tính***

```
sns.barplot(x="Gender", y="TotalSpent", data=df,
estimator=sum, ci=None, palette="pastel")
plt.title("Tổng chi tiêu theo giới tính")
plt.ylabel("Tổng chi tiêu")
plt.show()
```

Cho thấy nhóm nào chi tiêu nhiều hơn, giúp gợi ý chiến lược marketing phù hợp.

Đến đây, sinh viên đã biết vẽ bar chart, histogram, line chart với dữ liệu thật, thấy rõ sự khác biệt theo vùng, theo giới tính, theo thời gian, đồng thời đây là tiền đề để phân tích sâu hơn mối quan hệ giữa các biến ở bài thực hành tiếp theo.

## *Thực hành 4-4* ***Phân tích quan hệ giữa các biến***

**Mục tiêu:**

- Phân tích mối quan hệ giữa số lượng sản phẩm, tổng chi tiêu và giới tính.
- Phát hiện nhóm khách hàng chi tiêu cao/thấp.
- Tìm mối tương quan giữa các biến để chuẩn bị cho mô hình hóa.

***Bước 1. Scatter Plot – Tổng chi tiêu theo số lượng sản phẩm***

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.scatterplot(
    x="Quantity", y="TotalSpent", data=df, hue="Gender",
palette="Set1", s=80
)
plt.title("Tổng chi tiêu theo số lượng sản phẩm")
plt.xlabel("Số lượng sản phẩm")
plt.ylabel("Tổng chi tiêu")
plt.show()
```

Quan sát xem khách mua nhiều sản phẩm có xu hướng chi tiêu cao hơn không, có sự khác biệt giữa giới tính không.

***Bước 2. Box Plot – So sánh chi tiêu theo vùng***

```
sns.boxplot(x="Region", y="TotalSpent", data=df,
palette="pastel")
plt.title("Phân phối chi tiêu theo vùng")
plt.ylabel("Tổng chi tiêu")
plt.show()
```

Box plot giúp thấy phân bố, trung vị và các outlier – có vùng nào có nhiều khách hàng chi tiêu đột biến không?

***Bước 3. Heatmap – Ma trận tương quan giữa các biến số***

```
corr = df[["Quantity", "Price", "TotalSpent"]].corr()

sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Ma trận tương quan giữa các biến số")
plt.show()
```

Xác định biến nào ảnh hưởng mạnh đến **TotalSpent** (ví dụ Quantity hay Price).

***Bước 4. Pair Plot – Quan sát mối quan hệ nhiều biến cùng lúc***

```
sns.pairplot(df[["Quantity", "Price", "TotalSpent"]])
```

```
plt.suptitle("Mối quan hệ giữa Quantity, Price, TotalSpent",
y=1.02)
plt.show()
```

Pair plot hữu ích để phát hiện quan hệ tuyến tính/phi tuyến, phân bố từng biến, và tương quan trực quan hơn.

Qua bài tập này sinh viên xác định được mối quan hệ chính giữa các biến (ví dụ TotalSpent tăng khi Quantity tăng), phát hiện vùng hoặc nhóm khách hàng có phân phối chi tiêu khác biệt cũng như nắm được biến nào nên dùng làm đặc trưng (feature) quan trọng trong mô hình dự báo.

### *4.7.3 Case Study - Phân tích dữ liệu bán hàng*

Một cửa hàng bán lẻ online đã thu thập dữ liệu bán hàng trong 3 tháng gần đây. Dữ liệu bao gồm thông tin về:

- Ngày mua hàng (OrderDate)
- Sản phẩm (Product)
- Số lượng (Quantity)
- Giá bán (Price)
- Tổng chi tiêu (TotalSpent)
- Vùng (Region)
- Giới tính khách hàng (Gender)

Cửa hàng muốn phân tích dữ liệu để:

- Hiểu được xu hướng doanh thu theo thời gian.
- Xác định sản phẩm bán chạy nhất, sản phẩm bán chậm.
- So sánh hành vi mua sắm theo giới tính và khu vực.
- Phát hiện những mối quan hệ giữa các biến số (ví dụ Quantity – TotalSpent).
- Đưa ra đề xuất chiến lược marketing và tối ưu tồn kho.

***Bước 1: Chuẩn bị dữ liệu***

```
import pandas as pd

df = pd.read_csv("customer_orders.csv",
parse_dates=["OrderDate"])
```

```
print(df.head())
print(df.info())
print(df.describe())
```

Kiểm tra dữ liệu, các kiểu cột, dữ liệu thiếu. Làm sạch nếu cần: điền giá trị thiếu, loại bỏ dòng sai định dạng.

***Bước 2: Thống kê mô tả***

- Tính mean, median, std cho Quantity và TotalSpent.
- Đếm số giao dịch, số sản phẩm khác nhau.
- Tổng doanh thu theo từng Region và Gender.

***Bước 3: Trực quan hóa dữ liệu***

- Bar chart: Tổng chi tiêu theo sản phẩm hoặc vùng.
- Line chart: Xu hướng tổng doanh thu theo ngày.
- Box plot: So sánh chi tiêu giữa các vùng.
- Scatter plot: Tổng chi tiêu theo số lượng, phân nhóm theo giới tính.
- Heatmap: Ma trận tương quan giữa Quantity, Price, TotalSpent.

***Bước 4: Diễn giải kết quả***

- Xác định sản phẩm bán chạy nhất, sản phẩm doanh thu thấp.
- Nhận diện vùng có doanh thu cao nhất/thấp nhất.
- Phát hiện mối tương quan chính (Quantity – TotalSpent, Price – TotalSpent).

***Bước 5 – Đề xuất hành động***

- Gợi ý chiến lược marketing: ví dụ chạy khuyến mãi cho vùng doanh thu thấp.
- Đề xuất tối ưu tồn kho cho sản phẩm bán chạy.
- Đề xuất khuyến khích nhóm khách hàng ít chi tiêu để tăng doanh số.

Yêu cầu: Sinh viên viết một báo cáo ngắn (1–2 trang) trình bày:

- Tóm tắt dữ liệu (số dòng, số sản phẩm, số khách hàng, tổng doanh thu).

- Biểu đồ chính (ít nhất 3 loại biểu đồ).
- Insight chính (sản phẩm nào bán chạy, vùng nào mạnh/yếu).
- Đề xuất hành động dựa trên kết quả phân tích.

### *4.7.4 Bài tập mở rộng*

#### *Bài tập 4-1 Phân tích theo tháng*

- Tạo cột Month từ OrderDate.
- Tính tổng doanh thu theo từng tháng.
- Vẽ biểu đồ line chart để quan sát xu hướng doanh thu theo thời gian.
- Diễn giải: Có tháng nào cao đột biến? Nguyên nhân có thể là gì?

#### *Bài tập 4-2 Doanh thu trên mỗi sản phẩm*

- Tính doanh thu trung bình trên mỗi sản phẩm (TotalSpent / Quantity).
- Vẽ bar chart so sánh giá trị này giữa các sản phẩm.
- Diễn giải: Sản phẩm nào có giá bán trung bình cao nhất/thấp nhất?

#### *Bài tập 4-3 Phân tích theo khách hàng*

- Tính tổng doanh thu cho từng khách hàng (nếu có cột CustomerID).
- Vẽ histogram để xem phân phối chi tiêu của khách hàng.
- Chia khách hàng thành 3 nhóm: Low spenders, Medium spenders, High spenders (dựa trên phân vị 33% và 66%).
- Diễn giải: nhóm nào chiếm tỷ trọng doanh thu cao nhất?

#### *Bài tập 4-4 Phân tích theo vùng & giới tính*

- Tạo bảng tổng hợp (pivot table) cho tổng doanh thu theo Region và Gender.
- Vẽ stacked bar chart để trực quan hóa sự khác biệt giữa nam và nữ theo từng vùng.
- Đề xuất chiến lược marketing riêng cho từng vùng dựa trên kết quả.

#### *Bài tập 4-5 Mở rộng với dataset khác*

- Tải một open dataset từ Kaggle hoặc Google Dataset Search (ví dụ về bán hàng, e-commerce hoặc dịch vụ).
- Thực hiện lại quy trình phân tích: thống kê mô tả, trực quan hóa, phân tích quan hệ.

- Viết báo cáo so sánh kết quả với dataset customer_orders.csv.

# CHƯƠNG 5 NHẬP MÔN MACHINE LEARNING

Machine learning (học máy) là lĩnh vực cốt lõi trong trí tuệ nhân tạo, giúp máy tính học từ dữ liệu và tự đưa ra dự đoán hoặc quyết định mà không cần lập trình tường minh từng bước. Trong bối cảnh khoa học dữ liệu, machine learning là công cụ mạnh mẽ để khai thác thông tin, dự báo xu hướng và tối ưu hóa quyết định dựa trên dữ liệu thực tế.

Chương này giới thiệu các khái niệm cơ bản, quy trình xây dựng mô hình, công cụ và thuật toán phổ biến, đồng thời hướng dẫn sinh viên thực hành xây dựng mô hình đơn giản bằng Python.

Sau khi học xong chương này, sinh viên có thể:

1. Hiểu được **khái niệm cơ bản và phân loại machine learning** (học có giám sát, học không giám sát).
2. Nắm được **quy trình xây dựng một mô hình machine learning**, từ chuẩn bị dữ liệu đến đánh giá mô hình.
3. Sử dụng **công cụ Python cơ bản** như scikit-learn để triển khai các mô hình.
4. Thực hành xây dựng **một số thuật toán cơ bản**, áp dụng cho dữ liệu mẫu.
5. Phân tích và **diễn giải kết quả mô hình** để rút ra insight hữu ích.

## 5.1 Khái niệm cơ bản

Machine learning (học máy) là lĩnh vực nghiên cứu và phát triển các thuật toán giúp máy tính tự học từ dữ liệu và cải thiện hiệu suất theo thời gian mà không cần lập trình tường minh từng bước. Nói cách khác, máy học từ dữ liệu lịch sử (historical data) để dự đoán, phân loại hoặc đưa ra quyết định cho dữ liệu mới [4].

**Phân loại machine learning**

1. **Học có giám sát (supervised learning):**

- Máy học từ dữ liệu đã có nhãn (labeled data) để dự đoán nhãn cho dữ liệu mới.
- Ví dụ: dự đoán giá nhà dựa trên diện tích, số phòng; phân loại email là spam hay không spam.

2. **Học không giám sát (unsupervised learning):**
   - Máy học từ dữ liệu **không có nhãn**, tìm ra cấu trúc, nhóm (cluster) hoặc mô hình tiềm ẩn.
   - Ví dụ: phân nhóm khách hàng theo hành vi mua sắm; phát hiện bất thường trong giao dịch tài chính.
3. **Học bán giám sát (semi-supervised learning):**
   - Kết hợp dữ liệu có nhãn và không có nhãn để cải thiện dự đoán.
   - Dùng khi nhãn dữ liệu khó thu thập hoặc tốn kém.
4. **Học tăng cường (reinforcement learning):**
   - Máy học thông qua **thử và sai**, nhận thưởng hoặc phạt dựa trên hành vi để tối ưu mục tiêu.
   - Ví dụ: huấn luyện robot đi, chơi cờ, quản lý kho hàng thông minh.

**Ứng dụng cơ bản**

- Dự báo (forecasting) doanh số, nhu cầu, giá cả.
- Phân loại hình ảnh, văn bản, email.
- Phát hiện gian lận, bất thường trong giao dịch.
- Khuyến nghị sản phẩm, tối ưu hóa marketing.

**Ví dụ minh họa Python (học có giám sát đơn giản)**

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Dữ liệu mẫu: diện tích (m2) và giá nhà (triệu VND)
X = np.array([[50], [60], [70], [80], [90]])
y = np.array([500, 600, 650, 700, 750])

# Tạo mô hình hồi quy tuyến tính
model = LinearRegression()
```

```
model.fit(X, y)

# Dự đoán giá nhà cho diện tích 100 m2
gia_du_doan = model.predict([[100]])
print("Giá dự đoán cho 100 m2:", gia_du_doan[0])
```

**Giải thích:**

- LinearRegression() xây dựng mô hình hồi quy tuyến tính.
- fit() huấn luyện mô hình với dữ liệu.
- predict() dự đoán giá trị mới dựa trên dữ liệu đầu vào.

## 5.2 Quy trình xây dựng mô hình ML

Xây dựng một mô hình machine learning không chỉ là viết code mà còn bao gồm **một quy trình khoa học để đảm bảo mô hình học chính xác, đáng tin cậy và có khả năng áp dụng vào thực tế**.

**Các bước cơ bản trong quy trình**

1. **Xác định vấn đề và mục tiêu**
    - Xác định bài toán muốn giải quyết: dự đoán, phân loại, phân nhóm, phát hiện bất thường…
    - Xác định biến mục tiêu (target variable) và biến đầu vào (features).
    - Ví dụ: dự đoán giá nhà, phân loại email spam, phân nhóm khách hàng.
2. **Thu thập dữ liệu (data collection)**
    - Từ cơ sở dữ liệu, file CSV, API, web scraping hoặc dữ liệu công khai.
    - Đảm bảo dữ liệu **đa dạng, đầy đủ và đại diện cho vấn đề**.
3. **Tiền xử lý dữ liệu (data preprocessing)**
    - Xử lý giá trị thiếu (missing values), loại bỏ outlier nếu cần.
    - Chuyển đổi dữ liệu dạng categorical sang numerical (one-hot encoding).
    - Chuẩn hóa dữ liệu (normalization/standardization) nếu thuật toán yêu cầu.
4. **Chia dữ liệu (train-test split)**

- Chia dữ liệu thành tập huấn luyện (training set) và tập kiểm thử (test set).
- Thường tỷ lệ 70–80% cho huấn luyện, 20–30% cho kiểm thử.

5. **Lựa chọn và huấn luyện mô hình (model selection & training)**
   - Chọn thuật toán phù hợp: hồi quy, cây quyết định, kNN, SVM…
   - Huấn luyện mô hình trên tập dữ liệu huấn luyện.
6. **Đánh giá mô hình (model evaluation)**
   - Sử dụng tập test để kiểm tra hiệu quả.
   - Các chỉ số phổ biến:
     - Regression: MAE, MSE, RMSE, $R^2$
     - Classification: Accuracy, Precision, Recall, F1-score, ROC-AUC
   - Điều chỉnh tham số nếu cần (hyperparameter tuning).
7. **Triển khai và áp dụng mô hình (deployment)**
   - Áp dụng mô hình vào dữ liệu thực tế, tích hợp vào hệ thống.
   - Giám sát hiệu quả và cập nhật mô hình theo thời gian.

**Ví dụ minh họa quy trình đơn giản với scikit-learn**

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Dữ liệu mẫu
X = [[50],[60],[70],[80],[90]]
y = [500,600,650,700,750]

# Chia dữ liệu
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán và đánh giá
```

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

**Giải thích:**

- train_test_split() chia dữ liệu thành tập huấn luyện và tập kiểm thử.
- fit() huấn luyện mô hình.
- predict() dự đoán trên dữ liệu test.
- mean_squared_error() đánh giá độ chính xác của mô hình.

## 5.3 Công cụ cơ bản: Scikit-Learn

Scikit-Learn là **thư viện Python phổ biến nhất cho machine learning**, cung cấp **các thuật toán, công cụ tiền xử lý, đánh giá và tối ưu mô hình**. Nó đặc biệt thích hợp cho sinh viên và người mới học nhờ **giao diện đơn giản và nhất quán**.

**Các tính năng chính của Scikit-Learn**

1. **Thuật toán học máy (Machine Learning Algorithms):**
    - Học có giám sát: Linear Regression, Logistic Regression, Decision Tree, Random Forest, k-Nearest Neighbors (kNN), Support Vector Machines (SVM)…
    - Học không giám sát: K-Means, DBSCAN, PCA, Hierarchical Clustering…
2. **Tiền xử lý dữ liệu (Preprocessing):**
    - Chuẩn hóa dữ liệu (StandardScaler)
    - Biến đổi dữ liệu categorical (OneHotEncoder, LabelEncoder)
    - Loại bỏ giá trị thiếu (SimpleImputer)
3. **Chia dữ liệu (Train-test split):**
    - train_test_split() giúp tách dữ liệu thành tập huấn luyện và kiểm thử.
4. **Đánh giá mô hình (Model Evaluation):**
    - Regression: mean squared error, $R^2$ score
    - Classification: accuracy, precision, recall, F1-score

5. **Tối ưu tham số (Hyperparameter Tuning):**
    - GridSearchCV, RandomizedSearchCV giúp tìm tham số tốt nhất cho mô hình.

**Ví dụ cơ bản: hồi quy tuyến tính với Scikit-Learn**

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Dữ liệu mẫu: diện tích (m2) và giá nhà (triệu VND)
X = [[50], [60], [70], [80], [90]]
y = [500, 600, 650, 700, 750]

# Chia dữ liệu
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Tạo mô hình và huấn luyện
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán và đánh giá
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Giá dự đoán:", y_pred)
```

**Giải thích:**

- Tạo mô hình LinearRegression và huấn luyện trên tập huấn luyện (fit).
- Dự đoán giá trị trên tập test (predict) và đánh giá bằng mean_squared_error.
- Quy trình này là cơ bản nhưng **giúp sinh viên hiểu vòng đời mô hình trong Scikit-Learn**.

## 5.4 Một số thuật toán cơ bản

Trong machine learning, có hai nhóm thuật toán phổ biến: học có giám sát (supervised learning) và học không giám sát (unsupervised learning). Chúng ta sẽ xem xét các thuật toán cơ bản và ví dụ minh họa Python [2], [6].

### *5.4.1 Học có giám sát*

**a) Hồi quy tuyến tính (Linear Regression)**

- Dự đoán giá trị liên tục dựa trên quan hệ tuyến tính giữa biến đầu vào và biến mục tiêu.
- Ví dụ: dự đoán giá nhà dựa trên diện tích, số phòng.

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = np.array([[50], [60], [70], [80], [90]])
y = np.array([500, 600, 650, 700, 750])

model = LinearRegression()
model.fit(X, y)
print("Dự đoán giá nhà cho 100 m2:", model.predict([[100]]))
```

**b) Phân loại (Classification) – Logistic Regression**

- Dự đoán nhãn rời rạc, ví dụ spam/không spam, bệnh/norm.

```
from sklearn.linear_model import LogisticRegression

X = [[0],[1],[2],[3],[4],[5]]
y = [0,0,0,1,1,1]  # 0: không spam, 1: spam

model = LogisticRegression()
model.fit(X, y)
print("Dự đoán nhãn cho 2.5:", model.predict([[2.5]]))
```

**c) Cây quyết định (Decision Tree)**

- Dễ hiểu, trực quan hóa, dùng cho phân loại hoặc hồi quy.

```
from sklearn.tree import DecisionTreeClassifier

X = [[0], [1], [2], [3], [4], [5]]
y = [0,0,0,1,1,1]

model = DecisionTreeClassifier()
model.fit(X, y)
print("Dự đoán nhãn cho 2.5:", model.predict([[2.5]]))
```

### *5.4.2 Học không giám sát*

**a) Phân nhóm K-Means (Clustering)**

- Nhóm dữ liệu thành K cụm dựa trên khoảng cách.

```
from sklearn.cluster import KMeans
import numpy as np

X = np.array([[1,2],[1,4],[1,0],[4,2],[4,4],[4,0]])
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)
print("Nhãn cụm:", kmeans.labels_)
```

**b) Phân tích thành phần chính (PCA – Principal Component Analysis)**

- Giảm số chiều dữ liệu, giữ thông tin quan trọng, hỗ trợ trực quan hóa và giảm tải mô hình.

```
from sklearn.decomposition import PCA

X = np.array([[2,3,4],[3,4,5],[4,5,6]])
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
print("Dữ liệu giảm chiều:", X_reduced)
```

**Ghi chú:**

- Học có giám sát cần **dữ liệu có nhãn**, học không giám sát dùng **dữ liệu không nhãn**.
- Sinh viên nên thử trực tiếp với **dữ liệu mẫu và các thuật toán đơn giản** để cảm nhận cách hoạt động.

## 5.5 Đánh giá mô hình

Đánh giá mô hình là bước quan trọng để kiểm tra hiệu quả, độ chính xác và khả năng áp dụng thực tế của mô hình machine learning. Tùy vào loại bài toán, các chỉ số đánh giá sẽ khác nhau [7].

### *5.5.1 Đánh giá mô hình hồi quy*

Đánh giá mô hình hồi quy (Regression) gồm các yếu tố sau:

- **Mean Absolute Error (MAE):** trung bình sai số tuyệt đối giữa giá trị thực và dự đoán.
- **Mean Squared Error (MSE):** trung bình bình phương sai số; nhấn mạnh sai số lớn hơn.
- **Root Mean Squared Error (RMSE):** căn bậc hai của MSE, đơn vị giống với biến mục tiêu.
- **$R^2$ score (Coefficient of Determination):** tỷ lệ phương sai dữ liệu được mô hình giải thích (0–1, càng gần 1 càng tốt).

**Ví dụ Python:**

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
import numpy as np

y_true = [500, 600, 650, 700, 750]
y_pred = [510, 590, 640, 710, 740]

mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_true, y_pred)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R²:", r2)
```

### *5.5.2 Đánh giá mô hình phân loại*

Đánh giá mô hình phân loại (Classification) gồm các yếu tố sau:

- **Accuracy (Độ chính xác):** tỷ lệ dự đoán đúng trên tổng số mẫu.
- **Precision (Độ chính xác của nhãn dương):** trong số dự đoán dương, bao nhiêu là đúng.
- **Recall (Độ nhạy):** trong số mẫu thực sự dương, mô hình dự đoán đúng bao nhiêu.
- **F1-score:** trung bình điều hòa giữa precision và recall.
- **Confusion Matrix:** ma trận thể hiện dự đoán đúng/sai theo từng lớp.

**Ví dụ Python:**

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix

y_true = [0, 0, 1, 1, 1]
y_pred = [0, 0, 1, 0, 1]

print("Accuracy:", accuracy_score(y_true, y_pred))
print("Precision:", precision_score(y_true, y_pred))
print("Recall:", recall_score(y_true, y_pred))
print("F1-score:", f1_score(y_true, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
```

Các lưu ý khi đánh giá mô hình:

1. Luôn chia dữ liệu **train-test** để đánh giá mô hình trên dữ liệu chưa thấy.
2. Với dữ liệu ít, có thể dùng **cross-validation** để đánh giá ổn định hơn.
3. Chọn chỉ số phù hợp với bài toán:
    - Hồi quy: MAE, RMSE, $R^2$
    - Phân loại: Accuracy, F1-score, ROC-AUC

### *5.5.3 Kỹ thuật đánh giá nâng cao*

#### *5.5.3.1 Cross-Validation (CV)*

Cross-Validation là kỹ thuật đánh giá mô hình bằng cách chia dữ liệu thành nhiều tập con, lần lượt huấn luyện trên một số tập và kiểm tra trên tập còn lại.

- **K-Fold CV:** Chia dữ liệu thành K phần bằng nhau. Lặp K lần: mỗi lần chọn một phần làm tập kiểm tra, phần còn lại làm tập huấn luyện. Tính điểm trung bình để đánh giá mô hình.
- **Stratified K-Fold:** Dùng cho bài toán phân loại, đảm bảo tỉ lệ các lớp trong mỗi fold giống như toàn bộ dữ liệu.
- **Ưu điểm:** Giảm phụ thuộc vào một lần chia dữ liệu, đánh giá ổn định hơn.
- **Ví dụ:**

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
```

```
scores = cross_val_score(DecisionTreeClassifier(), X, y, cv=5)
print(scores.mean())
```

#### *5.5.3.2 Bias-Variance Tradeoff*

Mối quan hệ giữa độ phức tạp mô hình và hiệu quả dự đoán:

- **Bias cao:** Mô hình quá đơn giản → underfitting.
- **Variance cao:** Mô hình quá phức tạp → overfitting.
- **Giải pháp:** Điều chỉnh độ phức tạp (max_depth, regularization) và sử dụng cross-validation để chọn mô hình tối ưu.
- **Ví dụ minh họa**

Sử dụng Decision Tree với các giá trị max_depth khác nhau để quan sát lỗi train/test.

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

# Giả sử đã có X, y
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

train_errors, test_errors, depths = [], [], range(1, 16)

for d in depths:
model = DecisionTreeRegressor(max_depth=d, random_state=0)
model.fit(X_train, y_train)
train_errors.append(mean_squared_error(y_train,
model.predict(X_train)))
test_errors.append(mean_squared_error(y_test,
model.predict(X_test)))

plt.plot(depths, train_errors, label="Training Error")
plt.plot(depths, test_errors, label="Test Error")
plt.xlabel("Max Depth")
plt.ylabel("MSE")
plt.legend()
plt.title("Bias-Variance Tradeoff")
plt.show()
```

**Diễn giải:**

- Training error giảm dần khi max_depth tăng → mô hình học chi tiết hơn.
- Test error giảm đến một mức tối ưu, sau đó tăng trở lại → overfitting.
- Điểm test error nhỏ nhất chính là mức phức tạp tối ưu (cân bằng giữa bias và variance).

### *5.5.4 Tối ưu hóa mô hình và tự động hóa quy trình*

#### *5.5.4.1 Hyperparameter Tuning*

Hyperparameter là các tham số do người dùng cấu hình, không được mô hình tự học. Tối ưu chúng giúp cải thiện độ chính xác.

- **Grid Search:** Thử tất cả tổ hợp tham số, chọn kết quả tốt nhất.
- **Random Search:** Chọn ngẫu nhiên một số tổ hợp tham số, nhanh hơn khi không gian tìm kiếm lớn.
- **Bayesian Optimization:** (Giới thiệu ở mức khái niệm) – chọn thông minh dựa trên kết quả trước đó, ít lần thử hơn.

Ví dụ:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'max_depth':[3,5,7],
'min_samples_split':[2,5,10]}
grid = GridSearchCV(DecisionTreeClassifier(), param_grid,
cv=5)
grid.fit(X, y)
print(grid.best_params_)
```

#### *5.5.4.2 Pipeline và ColumnTransformer*

Pipeline cho phép đóng gói toàn bộ quy trình tiền xử lý + mô hình thành một đối tượng duy nhất, đảm bảo tính nhất quán và tránh rò rỉ dữ liệu.

- **Pipeline:** Chuỗi các bước [(tên_bước, transformer/mô hình)].
- **ColumnTransformer:** Cho phép xử lý riêng biệt các cột số và cột phân loại.

Ví dụ:

```
from sklearn.pipeline import Pipeline
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
pipe = Pipeline([
('scaler', StandardScaler()),
('clf', LogisticRegression())
])
pipe.fit(X_train, y_train)
```

## 5.6 Tóm tắt chương

Trong chương này, sinh viên đã học về các kiến thức và kỹ thuật nền tảng của Machine Learning:

1. **Khái niệm cơ bản:** ML là phương pháp giúp máy tính học từ dữ liệu thay vì lập trình theo quy tắc cứng.
2. **Quy trình xây dựng mô hình:** Xác định vấn đề → chuẩn bị dữ liệu → chọn mô hình → huấn luyện → đánh giá → triển khai.
3. **Thuật toán cơ bản:** Học có giám sát (Linear/Logistic Regression, Decision Tree, KNN), học không giám sát (K-Means, PCA).
4. **Công cụ chính:** scikit-learn, pandas, numpy, matplotlib.
5. **Đánh giá và tối ưu mô hình:** Bao gồm train/test split, cross-validation (K-Fold, Stratified K-Fold), các thước đo đánh giá (accuracy, MSE, confusion matrix), cùng kỹ thuật tối ưu hyperparameter bằng Grid Search, Random Search và sử dụng Pipeline để tự động hóa quy trình.

Nhờ nắm vững các nội dung này, sinh viên có thể xây dựng và đánh giá mô hình ML một cách có hệ thống, hạn chế underfitting/overfitting thông qua việc cân bằng bias-variance và tối ưu hóa tham số.

## 5.7 Câu hỏi và Bài tập.

### *5.7.1 Câu hỏi ôn tập*

*Câu hỏi 5. 1.* Khái niệm cơ bản:

- Machine learning là gì? Nó khác gì với lập trình truyền thống?
- Nêu sự khác nhau giữa học có giám sát và học không giám sát.

*Câu hỏi 5. 2.* Quy trình xây dựng mô hình:

- Liệt kê các bước cơ bản trong quy trình xây dựng mô hình machine learning.
- Tại sao cần chia dữ liệu thành tập huấn luyện và tập kiểm thử?

*Câu hỏi 5. 3.* Thuật toán cơ bản:

- Nêu ví dụ ít nhất hai thuật toán học có giám sát và hai thuật toán học không giám sát.
- Trong trường hợp dự đoán giá nhà, bạn nên dùng loại thuật toán nào? Tại sao?

*Câu hỏi 5. 4.* Đánh giá mô hình:

- Nêu các chỉ số đánh giá cho bài toán hồi quy.
- Nêu các chỉ số đánh giá cho bài toán phân loại.
- Tại sao cần sử dụng cross-validation trong một số trường hợp?

*Câu hỏi 5. 5.* Công cụ Scikit-Learn:

- Scikit-Learn hỗ trợ những tính năng chính nào cho machine learning?
- Làm thế nào để chuẩn hóa dữ liệu và mã hóa biến categorical trong Scikit-Learn?

### *5.7.2 Hướng dẫn thực hành*

*Thực hành 5-1* ***Chuẩn bị dữ liệu & Hồi quy tuyến tính đơn giản***

**Mục tiêu:**

- Làm quen với scikit-learn, chia dữ liệu train/test.
- Huấn luyện mô hình Linear Regression và đánh giá bằng MSE, $R^2$.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Dataset ví dụ: boston.csv (cột RM: số phòng trung bình,
MEDV: giá nhà)
df = pd.read_csv("boston.csv")

X = df[["RM"]]   # số phòng
y = df["MEDV"]   # giá nhà
```

```
# Chia dữ liệu train/test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Huấn luyện mô hình
model = LinearRegression()
model.fit(X_train, y_train)

# Dự đoán và đánh giá
y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("R²:", r2_score(y_test, y_pred))
```

### *Thực hành 5-2* ***Hồi quy đa biến***

**Mục tiêu:**

- Dùng nhiều biến đầu vào để cải thiện dự đoán.
- So sánh hiệu quả với hồi quy một biến.

```
# Chọn nhiều biến đầu vào
X = df[["RM", "LSTAT", "PTRATIO"]]  # số phòng, % dân nghèo,
tỷ lệ GV/HS
y = df["MEDV"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("R²:", r2_score(y_test, y_pred))
```

### *Thực hành 5-3* ***Phân loại với Logistic Regression***

**Mục tiêu:**

- Thực hành bài toán phân loại nhị phân.
- Đánh giá mô hình bằng accuracy, precision, recall, F1.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
```

```
# Dataset: drug200.csv (predict Drug dựa trên Age, Sex, BP,
Cholesterol, Na_to_K)
df = pd.read_csv("drug200.csv")

X = df[["Age", "Na_to_K"]]  # chọn 2 đặc trưng để đơn giản
y = (df["Drug"] == "drugY").astype(int)  # 1: dùng drugY, 0:
khác

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1-score:", f1_score(y_test, y_pred))
```

*Thực hành 5-4* ***Phân nhóm với K-Means***

**Mục tiêu:**

- Làm quen với bài toán không giám sát.
- Quan sát cách KMeans phân nhóm dữ liệu.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

X = df[["Age", "Na_to_K"]]

# Tìm số cụm tối ưu bằng Elbow Method
inertia = []
k_values = range(2, 8)
for k in k_values:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X)
    inertia.append(km.inertia_)

plt.plot(k_values, inertia, marker="o")
plt.xlabel("Số cụm k")
plt.ylabel("Inertia")
plt.title("Elbow Method")
plt.show()
```

```
# Huấn luyện với số cụm tối ưu (ví dụ k=3)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)

print("Silhouette Score:", silhouette_score(X, clusters))

plt.scatter(X["Age"], X["Na_to_K"], c=clusters,
cmap="viridis")
plt.xlabel("Age")
plt.ylabel("Na_to_K")
plt.title("KMeans với k=3")
plt.show()
```

*Thực hành 5-5* ***Đánh giá mô hình nâng cao***

**Mục tiêu:**

- Hiểu và áp dụng kỹ thuật Cross-Validation (K-Fold) để đánh giá mô hình ổn định hơn.
- Quan sát sự thay đổi kết quả mô hình khi thay đổi train/test split.

***Bước 1: Chuẩn bị dữ liệu***

- Dùng dataset Boston Housing (hoặc California Housing nếu boston.csv không còn).
- Chọn một số biến đầu vào (ví dụ: RM, LSTAT, PTRATIO) để dự đoán MEDV.

***Bước 2: Huấn luyện và đánh giá với Train/Test Split nhiều lần***

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Đọc dữ liệu
# df = pd.read_csv("boston.csv") # Nếu có sẵn
from sklearn.datasets import fetch_california_housing
cal = fetch_california_housing(as_frame=True)
df = cal.frame

X = df[["MedInc", "AveRooms", "AveOccup"]]
y = df["MedHouseVal"]

for random_state in [0, 10, 42, 99, 123]:
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=random_state)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(f"Random state {random_state}: MSE =
{mean_squared_error(y_test, y_pred):.4f}")
```

- Nhận xét: MSE thay đổi khi chia dữ liệu khác nhau.

***Bước 3: Đánh giá bằng K-Fold Cross-Validation***

```
from sklearn.model_selection import cross_val_score
import numpy as np

model = LinearRegression()
scores = cross_val_score(model, X, y, cv=5,
scoring="neg_mean_squared_error")
print("MSE trung bình (5-Fold CV):", np.mean(-scores))
```

- Nhận xét: Kết quả ổn định hơn, ít phụ thuộc vào một lần chia dữ liệu.

**Câu hỏi thảo luận:**

- Vì sao kết quả MSE thay đổi khi thay random_state?
- Cross-Validation giúp gì trong việc lựa chọn mô hình tối ưu?
- Nếu tăng số Fold (K), kết quả thay đổi như thế nào?

**Kết quả mong đợi:**

- Sinh viên thấy được sự dao động của MSE khi chia dữ liệu khác nhau.
- Hiểu rằng Cross-Validation giúp đánh giá mô hình đáng tin cậy hơn, giảm overfitting/underfitting.

### *Thực hành 5-6 Tối ưu mô hình Decision Tree bằng Grid Search*

**Mục tiêu:**

- Hiểu cách tối ưu siêu tham số (hyperparameters) của mô hình bằng Grid Search.
- Thực hành Decision Tree Classifier với các giá trị tham số khác nhau.
- Đánh giá và chọn mô hình tốt nhất dựa trên Cross-Validation.

***Bước 1: Chuẩn bị dữ liệu***

- Dùng dataset Iris (có sẵn trong scikit-learn).

***Bước 2: Xây dựng mô hình Decision Tree và Grid Search***

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

# Tải dữ liệu
iris = load_iris()
X, y = iris.data, iris.target

# Định nghĩa lưới tham số để tìm kiếm
param_grid = {
    'max_depth': [2, 3, 4, 5, None],
    'min_samples_split': [2, 4, 6],
    'criterion': ['gini', 'entropy']
}

# Tạo mô hình và GridSearchCV
dt = DecisionTreeClassifier(random_state=42)
grid = GridSearchCV(dt, param_grid, cv=5, scoring='accuracy')
grid.fit(X, y)

print("Best Parameters:", grid.best_params_)
print("Best CV Accuracy:", grid.best_score_)
```

***Bước 3: Đánh giá mô hình tốt nhất***

```
best_model = grid.best_estimator_
print("Accuracy trên toàn bộ dữ liệu:", best_model.score(X,
y))
```

***Bước 4: Trực quan hóa cây quyết định (Tùy chọn)***

```
from sklearn import tree
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
tree.plot_tree(best_model, filled=True,
feature_names=iris.feature_names,
class_names=iris.target_names)
```

```
plt.show()
```

**Câu hỏi thảo luận:**

- Vì sao cần dùng Grid Search thay vì thử tham số thủ công?
- Nếu thêm nhiều giá trị vào param_grid, thời gian chạy thay đổi thế nào?
- Cross-Validation giúp đảm bảo điều gì khi chọn tham số tối ưu?

**Kết quả mong đợi:**

- Sinh viên thấy được cách tìm ra tham số tối ưu.
- Hiểu vai trò của Cross-Validation trong việc đánh giá công bằng.
- Quan sát trực quan cây quyết định tối ưu (nếu vẽ).

### *Thực hành 5-7 Random Search & So sánh với Grid Search*

**Mục tiêu:**

- Làm quen với RandomizedSearchCV và so sánh kết quả với GridSearchCV.
- Quan sát sự khác biệt về thời gian chạy và độ chính xác.

***Bước 1: Chuẩn bị dữ liệu***

- Tiếp tục sử dụng dataset Iris.

***Bước 2: Xây dựng mô hình với Random Search***

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
import time

# Định nghĩa không gian tìm kiếm ngẫu nhiên
param_dist = {
'max_depth': [2, 3, 4, 5, None],
'min_samples_split': randint(2, 10),
'criterion': ['gini', 'entropy']
}

start = time.time()
random_search =
RandomizedSearchCV(DecisionTreeClassifier(random_state=42),
param_distributions=param_dist,
n_iter=10, cv=5, scoring='accuracy', random_state=42)
random_search.fit(X, y)
```

```
end = time.time()

print("Best Parameters (Random Search):",
random_search.best_params_)
print("Best CV Accuracy:", random_search.best_score_)
print(f"Thời gian chạy Random Search: {end - start:.4f} giây")
```

***Bước 3: So sánh với Grid Search***

```
from sklearn.model_selection import GridSearchCV

param_grid = {
'max_depth': [2, 3, 4, 5, None],
'min_samples_split': [2, 4, 6, 8],
'criterion': ['gini', 'entropy']
}

start = time.time()
grid = GridSearchCV(DecisionTreeClassifier(random_state=42),
param_grid, cv=5, scoring='accuracy')
grid.fit(X, y)
end = time.time()

print("Best Parameters (Grid Search):", grid.best_params_)
print("Best CV Accuracy:", grid.best_score_)
print(f"Thời gian chạy Grid Search: {end - start:.4f} giây")
```

**Câu hỏi thảo luận:**

- Khi không gian tham số rất lớn, Grid Search và Random Search khác nhau như thế nào về hiệu quả?
- Random Search có thể tìm được kết quả gần tối ưu không? Khi nào nên ưu tiên Random Search?
- Nếu tăng n_iter, kết quả có thay đổi đáng kể không?

**Kết quả mong đợi:**

- Sinh viên nhận thấy Random Search có thể nhanh hơn nhưng vẫn đạt kết quả gần tối ưu.
- Hiểu tình huống nào nên dùng Random Search để tiết kiệm tài nguyên tính toán.

## *Thực hành 5-8* ***Pipeline + ColumnTransformer***

**Mục tiêu:**

- Làm quen với Pipeline và ColumnTransformer trong scikit-learn.
- Thực hành xử lý dữ liệu hỗn hợp (numeric + categorical) trong một quy trình thống nhất.
- Tự động hóa toàn bộ quá trình tiền xử lý → huấn luyện → dự đoán.

### *Bước 1: Chuẩn bị dữ liệu*

- Dùng dataset titanic.csv hoặc bất kỳ dataset có cả cột số và cột phân loại.

### *Bước 2: Tách cột số và cột phân loại*

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Ví dụ: Titanic dataset
df = pd.read_csv("titanic.csv")

X = df[["Pclass", "Sex", "Age", "Fare"]]
y = df["Survived"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

### *Bước 3: Tạo Pipeline tiền xử lý*

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder,
StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression

# Danh sách cột
numeric_features = ["Age", "Fare"]
categorical_features = ["Pclass", "Sex"]

# Pipeline cho cột số
numeric_transformer = Pipeline(steps=[
('imputer', SimpleImputer(strategy='median')),
('scaler', StandardScaler())
```

```
])

# Pipeline cho cột phân loại
categorical_transformer = Pipeline(steps=[
('imputer', SimpleImputer(strategy='most_frequent')),
('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Gộp vào ColumnTransformer
preprocessor = ColumnTransformer(
transformers=[
('num', numeric_transformer, numeric_features),
('cat', categorical_transformer, categorical_features)
])

# Pipeline tổng
model = Pipeline(steps=[
('preprocessor', preprocessor),
('classifier', LogisticRegression(max_iter=1000))
])

# Huấn luyện
model.fit(X_train, y_train)
print("Accuracy trên tập test:", model.score(X_test, y_test))
Dự đoán thử với dữ liệu mới
sample = pd.DataFrame({
"Pclass": [3], "Sex": ["male"], "Age": [22], "Fare": [7.25]
})
print("Dự đoán sống sót:", model.predict(sample))
```

**Câu hỏi thảo luận:**

- Ưu điểm của việc dùng Pipeline thay vì xử lý thủ công từng bước?
- ColumnTransformer giúp ích thế nào khi dữ liệu có nhiều loại cột?
- Nếu thêm bước chọn đặc trưng hoặc tuning mô hình, Pipeline hỗ trợ ra sao?

**Kết quả mong đợi:**

- Sinh viên biết cách xây dựng Pipeline chuẩn trong scikit-learn.
- Hiểu cách xử lý dữ liệu số và phân loại cùng lúc.
- Thấy được lợi ích của tự động hóa quy trình tiền xử lý và huấn luyện.

### *5.7.3 Case Study*

**Case Study A: Dự đoán giá nhà đơn giản**

Bạn đang làm việc cho một công ty bất động sản. Công ty muốn xây dựng một mô hình dự đoán giá nhà dựa trên các đặc trưng như số phòng trung bình, tỷ lệ dân nghèo, tỷ lệ giáo viên/học sinh,…

Dữ liệu đã được thu thập trong file boston.csv, gồm các cột như:

- RM: số phòng trung bình
- LSTAT: % dân có thu nhập thấp
- PTRATIO: tỷ lệ giáo viên / học sinh
- MEDV: giá nhà trung vị (biến mục tiêu)

  và một số biến khác.

Mục tiêu:

- Xây dựng mô hình dự đoán giá nhà dựa trên các đặc trưng đầu vào.
- Đánh giá mô hình bằng $R^2$ và RMSE.
- Diễn giải ý nghĩa của các đặc trưng chính.

***Bước 1 – Chuẩn bị dữ liệu***

```
import pandas as pd

df = pd.read_csv("boston.csv")
print(df.head())
print(df.info())
print(df.describe())
```

- Kiểm tra dữ liệu, xem có giá trị thiếu không.
- Nếu cần, chuẩn hóa dữ liệu (StandardScaler) trước khi huấn luyện.

***Bước 2 – Chia dữ liệu train/test***

```
from sklearn.model_selection import train_test_split
```

```
X = df[["RM", "LSTAT", "PTRATIO"]]  # có thể thêm biến khác
y = df["MEDV"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

***Bước 3 – Huấn luyện mô hình***

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

***Bước 4 – Đánh giá mô hình***

```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("RMSE:", rmse)
print("R²:", r2)
```

***Bước 5 – Phân tích hệ số***

```
coef_df = pd.DataFrame({"Feature": X.columns, "Coefficient":
model.coef_})
print(coef_df)
```

- Giải thích biến nào ảnh hưởng tăng/giảm giá nhà.
- Ví dụ: hệ số RM dương → số phòng tăng → giá nhà tăng.

***Bước 6 – Diễn giải & đề xuất***

- Viết báo cáo 1 trang:
    - Mô hình dự đoán tốt đến mức nào ($R^2$).

- Yếu tố nào ảnh hưởng mạnh nhất đến giá nhà.
- Gợi ý: công ty nên ưu tiên đầu tư cải thiện yếu tố nào (VD: giảm PTRATIO, tăng số phòng trung bình).

**Yêu cầu nộp:**

- Notebook hoặc file Python chạy được.
- Kết quả đánh giá mô hình (RMSE, $R^2$).
- Bảng hệ số mô hình.
- Nhận xét về 3 biến quan trọng nhất.

**Case Study B: Phân loại loại thuốc cho bệnh nhân**

Bột bác sĩ muốn dự đoán loại thuốc thích hợp cho bệnh nhân dựa trên các đặc điểm lâm sàng:

- Tuổi (Age)
- Giới tính (Sex)
- Huyết áp (BP)
- Cholesterol
- Tỷ lệ Na/K (Na_to_K)

Dữ liệu được lưu trong file drug200.csv, mỗi dòng là một bệnh nhân với nhãn Drug (A, B, C, X, Y).

Mục tiêu: xây dựng mô hình phân loại để dự đoán loại thuốc cho bệnh nhân mới.

***Bước 1 – Chuẩn bị dữ liệu***

```
import pandas as pd

df = pd.read_csv("drug200.csv")
print(df.head())
print(df.info())
```

- Kiểm tra dữ liệu, xem phân phối của Drug.

- Mã hóa biến phân loại (Sex, BP, Cholesterol) bằng LabelEncoder hoặc OneHotEncoder.

***Bước 2 – Chia dữ liệu train/test***

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Mã hóa categorical
le_sex = LabelEncoder()
df["Sex"] = le_sex.fit_transform(df["Sex"])

le_bp = LabelEncoder()
df["BP"] = le_bp.fit_transform(df["BP"])

le_chol = LabelEncoder()
df["Cholesterol"] = le_chol.fit_transform(df["Cholesterol"])

X = df[["Age", "Sex", "BP", "Cholesterol", "Na_to_K"]]
y = df["Drug"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

***Bước 3 – Huấn luyện mô hình***

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

***Bước 4 – Đánh giá mô hình***

```
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Vẽ confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
```

```
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

***Bước 5 – Diễn giải kết quả***

- Xem chỉ số Precision, Recall, F1-score của từng loại thuốc.
- Nhận xét loại thuốc nào dễ phân loại, loại nào mô hình hay nhầm lẫn.
- Đề xuất cải thiện (ví dụ: thử RandomForestClassifier, tune max_depth của cây quyết định).

**Yêu cầu nộp:**

- Notebook/Python script chạy được.
- Accuracy tổng thể và bảng classification report.
- Ma trận nhầm lẫn trực quan.
- Nhận xét kết quả, đề xuất cải thiện mô hình.

### *5.7.4 Bài tập mở rộng*

#### *Bài tập 5-1 Hồi quy đa biến nâng cao*

- Dùng dữ liệu boston.csv.
- Thêm các biến đầu vào khác như CRIM, DIS, TAX để huấn luyện mô hình Linear Regression.
- So sánh $R^2$ và RMSE khi:
    1. Dùng một biến đầu vào (RM)
    2. Dùng nhiều biến đầu vào
- Nhận xét: mô hình đa biến có cải thiện độ chính xác không?

#### *Bài tập 5-2 Hồi quy Ridge và Lasso*

- Thử huấn luyện mô hình Ridge Regression và Lasso Regression trên cùng dữ liệu.
- Quan sát sự thay đổi hệ số khi tăng alpha (regularization).

- Giải thích ý nghĩa của việc làm trơn hệ số (regularization giúp gì cho mô hình).

### *Bài tập 5-3* ***Phân loại với nhiều mô hình***

- Dùng dữ liệu drug200.csv.
- Huấn luyện và so sánh các mô hình:
    1. Decision Tree
    2. k-Nearest Neighbors (k=3, 5, 7)
    3. Random Forest
- So sánh Accuracy, Precision, Recall, F1-score.
- Vẽ confusion matrix cho mô hình tốt nhất.

---

### *Bài tập 5-4* ***Feature Engineering***

- Tạo biến mới từ dữ liệu Boston:
    - TAX_RM = TAX / RM
    - LSTAT_RM = LSTAT * RM
- Thêm các biến này vào mô hình và quan sát sự thay đổi $R^2$.
- Thảo luận: có nên giữ các biến mới này không?

### *Bài tập 5-5* ***Cross-Validation***

- Thực hiện k-fold cross-validation (k=5) với mô hình Linear Regression.
- So sánh điểm $R^2$ trung bình với kết quả train-test split thông thường.
- Giải thích tại sao cross-validation cho kết quả ổn định hơn.

### *Bài tập 5-6* ***Phân nhóm khách hàng (Clustering)***

- Dùng dữ liệu drug200.csv, chọn cột Age và Na_to_K.
- Áp dụng K-Means clustering với k=3.
- Trực quan hóa kết quả phân cụm bằng scatter plot (màu theo cluster).
- So sánh phân cụm với nhãn thật Drug: có mối tương quan nào không?

## *Bài tập 5-7* ***Dự án Mini***

- Sinh viên tự chọn một open dataset (Kaggle hoặc UCI ML Repository).
- Thực hiện quy trình ML hoàn chỉnh:
    1. Chuẩn bị và làm sạch dữ liệu
    2. Huấn luyện ít nhất 2 mô hình (hồi quy hoặc phân loại)
    3. Đánh giá và so sánh mô hình
    4. Trình bày insight và đề xuất giải pháp
- Nộp báo cáo và notebook kèm theo.

# CHƯƠNG 6 TRÍ TUỆ NHÂN TẠO NHẬP MÔN

Trong nhiều thập kỷ qua, trí tuệ nhân tạo (artificial intelligence – AI) đã chuyển từ một khái niệm trong các phòng thí nghiệm nghiên cứu trở thành công nghệ cốt lõi trong hầu hết lĩnh vực của đời sống. Từ trợ lý ảo, xe tự lái, đến hệ thống chẩn đoán y khoa, AI đang thay đổi cách con người làm việc, học tập và tương tác với thế giới. Chương này cung cấp cái nhìn tổng quan về khái niệm, ứng dụng, các lĩnh vực liên quan cũng như những vấn đề đạo đức và xã hội đi kèm với AI. Người học sẽ tiếp cận các công cụ và framework phổ biến để có thể bắt đầu thử nghiệm các ứng dụng AI cơ bản.

Sau khi học xong chương này, sinh viên có thể:

- Hiểu rõ khái niệm trí tuệ nhân tạo và phân biệt với machine learning, deep learning.
- Nhận diện các ứng dụng tiêu biểu của AI trong công nghiệp và đời sống.
- Trình bày được các lĩnh vực chính trong nghiên cứu và ứng dụng AI.
- Phân tích một số vấn đề đạo đức và xã hội liên quan đến AI.
- Làm quen với các công cụ và framework phổ biến hỗ trợ phát triển AI.

## 6.1 Khái niệm cơ bản về Trí tuệ nhân tạo (AI)

Trí tuệ nhân tạo (*artificial intelligence – AI*) là một lĩnh vực trong khoa học máy tính tập trung vào việc tạo ra các hệ thống có khả năng thực hiện những nhiệm vụ thường đòi hỏi trí tuệ của con người, chẳng hạn như học hỏi (*learning*), suy luận (*reasoning*), nhận thức (*perception*), hiểu ngôn ngữ tự nhiên (*natural language understanding*) và ra quyết định (*decision making*) [5], [15].

Khái niệm AI không phải mới – từ thập niên 1950, Alan Turing đã đặt ra câu hỏi nổi tiếng *"Máy móc có thể suy nghĩ không?"* và đề xuất **Turing Test** để đánh giá khả năng thông minh của máy. Tuy nhiên, nhờ sự phát triển của dữ liệu lớn (*big data*),

sức mạnh tính toán và các thuật toán học máy (*machine learning algorithms*), AI mới thực sự bùng nổ trong thập niên gần đây.

Một số khái niệm thường được nhắc đến khi nói về AI:

- **Trí tuệ nhân tạo hẹp (narrow AI hay weak AI):** Hệ thống AI được thiết kế để thực hiện một nhiệm vụ cụ thể (ví dụ: nhận diện khuôn mặt, gợi ý phim trên Netflix). Đây là loại AI phổ biến hiện nay.
- **Trí tuệ nhân tạo tổng quát (general AI):** Hệ thống có khả năng hiểu, học hỏi và áp dụng kiến thức vào nhiều lĩnh vực khác nhau, tương tự trí tuệ con người. Đây là mục tiêu dài hạn, chưa đạt được.
- **Siêu trí tuệ nhân tạo (superintelligence):** Một giả thuyết về mức AI vượt xa trí tuệ con người trong mọi khía cạnh. Khái niệm này còn nhiều tranh cãi và mang tính dự báo.

Ví dụ gần gũi của AI trong đời sống hiện nay bao gồm: trợ lý ảo (Siri, Google Assistant), công cụ dịch ngôn ngữ tự động, hệ thống khuyến nghị sản phẩm trên các sàn thương mại điện tử, hay xe tự lái thử nghiệm của Tesla và Waymo.

## 6.2 Ứng dụng của AI trong đời sống và công nghiệp

Trí tuệ nhân tạo (AI) đã trở thành một phần không thể thiếu trong nhiều lĩnh vực của đời sống và sản xuất. Sự lan tỏa mạnh mẽ của AI đến từ khả năng xử lý dữ liệu khổng lồ, học hỏi từ kinh nghiệm và tối ưu hóa các quyết định [16]. Có thể phân loại ứng dụng AI thành một số nhóm chính sau.

### *6.2.1 Ứng dụng trong đời sống hằng ngày*

- **Trợ lý ảo (*virtual assistants*)**: Các công cụ như Siri, Google Assistant hay Alexa có thể nhận diện giọng nói, trả lời câu hỏi, điều khiển thiết bị thông minh trong gia đình.
- **Gợi ý nội dung (*recommendation systems*)**: AI đứng sau hệ thống đề xuất phim của Netflix, video của YouTube hay sản phẩm trên Shopee, Lazada.
- **Dịch ngôn ngữ tự động (*machine translation*)**: Google Translate hay DeepL hỗ trợ dịch nhanh hàng chục ngôn ngữ, nâng cao khả năng giao tiếp toàn cầu.
- **Ứng dụng trong y tế cá nhân**: Đồng hồ thông minh có thể theo dõi nhịp tim, giấc ngủ, gợi ý chế độ tập luyện dựa trên phân tích dữ liệu sức khỏe.

### *6.2.2 Ứng dụng trong công nghiệp và sản xuất*

- **Sản xuất thông minh (*smart manufacturing*)**: AI kết hợp với IoT để giám sát dây chuyền sản xuất, phát hiện lỗi và dự đoán sự cố bảo trì.
- **Xe tự lái (*autonomous vehicles*)**: Tesla, Waymo và nhiều hãng khác phát triển xe có khả năng tự điều khiển nhờ AI xử lý hình ảnh, tín hiệu giao thông và hành vi người lái.
- **Nông nghiệp thông minh (*smart agriculture*)**: AI hỗ trợ dự báo thời tiết, phát hiện sâu bệnh qua hình ảnh, tối ưu lượng phân bón và tưới tiêu.
- **Thương mại điện tử (*e-commerce*)**: Hệ thống chatbot chăm sóc khách hàng, phân tích hành vi người mua để tối ưu giá và chương trình khuyến mãi.
- **Tài chính (*fintech*)**: AI phát hiện giao dịch bất thường, hỗ trợ chấm điểm tín dụng, tự động tư vấn danh mục đầu tư.

### *6.2.3 Ứng dụng trong nghiên cứu và khoa học*

- **Phát hiện thuốc mới (*drug discovery*)**: AI giúp phân tích hàng triệu hợp chất hóa học, rút ngắn thời gian nghiên cứu thuốc.
- **Khám phá không gian (*space exploration*)**: NASA sử dụng AI để điều hướng robot thám hiểm sao Hỏa.
- **Khoa học dữ liệu (*data science*)**: AI tự động hóa phân tích dữ liệu, từ đó hỗ trợ nhà khoa học tập trung vào giải thích và ứng dụng kết quả.

Như vậy, AI không chỉ hiện diện trong các công nghệ cao mà còn đang thâm nhập sâu vào từng hoạt động thường nhật, từ việc mua sắm, giải trí đến y tế, giáo dục và sản xuất.

## 6.3 Các kỹ thuật và thuật toán nền tảng của trí tuệ nhân tạo

Trí tuệ nhân tạo (AI) bao gồm nhiều kỹ thuật khác nhau, mỗi kỹ thuật giải quyết một loại bài toán đặc thù [4],[5]. Một số nhóm kỹ thuật nền tảng thường được sử dụng trong thực tế gồm:

### *6.3.1 Học máy (machine learning)*

Học máy là phương pháp giúp máy tính tự cải thiện hiệu năng thông qua dữ liệu, mà không cần lập trình chi tiết từng bước. Thuật toán học máy phổ biến gồm: hồi quy

tuyến tính (linear regression), cây quyết định (decision tree), k-nearest neighbors (k-NN), và máy vector hỗ trợ (support vector machine – SVM).
Ví dụ: dự đoán giá nhà dựa trên dữ liệu diện tích, vị trí, số phòng.

### *6.3.2 Học sâu (deep learning)*

Học sâu là một nhánh của học máy, sử dụng mạng nơ-ron nhân tạo nhiều lớp (artificial neural networks). Các kiến trúc phổ biến gồm CNN (convolutional neural network) cho hình ảnh, RNN (recurrent neural network) cho dữ liệu chuỗi.
Ví dụ: nhận diện khuôn mặt trong ảnh, dịch tự động từ tiếng Anh sang tiếng Việt.

### *6.3.3 Xử lý ngôn ngữ tự nhiên (natural language processing – NLP)*

NLP giúp máy tính hiểu, phân tích và tạo ra ngôn ngữ của con người. Ứng dụng trải dài từ chatbot, dịch máy, phân tích cảm xúc, đến tóm tắt văn bản.
Ví dụ: chatbot hỗ trợ khách hàng, phân loại đánh giá sản phẩm thành tích cực/tiêu cực.

### *6.3.4 Thị giác máy tính (computer vision)*

Thị giác máy tính cho phép máy tính "nhìn" và hiểu được nội dung từ hình ảnh và video. Các bài toán điển hình gồm: nhận diện vật thể (object detection), phân loại hình ảnh (image classification), phân đoạn ảnh (image segmentation).
Ví dụ: xe tự lái phát hiện đèn giao thông và người đi bộ.

### *6.3.5 Học tăng cường (reinforcement learning – RL)*

RL là phương pháp trong đó tác nhân (agent) học cách hành động trong môi trường để tối đa hóa phần thưởng. Khác với học máy thông thường, RL dựa trên thử-sai (trial-and-error).
Ví dụ: AI chơi cờ vây (AlphaGo), robot học cách đi bộ.

Bảng so sánh các kỹ thuật AI chính

| **Kỹ thuật** | **Đặc điểm chính** | **Ví dụ ứng dụng** |
|---|---|---|
| Học máy (ML) | Học từ dữ liệu có gắn nhãn hoặc không gắn nhãn | Dự đoán giá, phân loại email spam |
| Học sâu (DL) | Mạng nơ-ron nhiều lớp, xử lý dữ liệu lớn | Nhận diện khuôn mặt, dịch máy |

| | | |
|---|---|---|
| NLP | Hiểu và sinh ngôn ngữ tự nhiên | Chatbot, phân tích cảm xúc |
| Thị giác máy tính | Xử lý ảnh và video | Xe tự lái, camera giám sát |
| RL | Học qua thử-sai, tối ưu phần thưởng | Game AI, robot |

## 6.4 Các lĩnh vực liên quan đến trí tuệ nhân tạo

Trí tuệ nhân tạo không tồn tại độc lập, mà phát triển nhờ sự kết hợp với nhiều lĩnh vực khoa học và công nghệ khác. Việc hiểu rõ các lĩnh vực liên quan giúp người học có cái nhìn toàn diện hơn về hệ sinh thái AI. Một số lĩnh vực tiêu biểu bao gồm:

- **Khoa học dữ liệu (Data Science)**: tập trung vào thu thập, xử lý và phân tích dữ liệu để rút ra tri thức. Đây là nền tảng quan trọng vì AI cần dữ liệu để học hỏi và đưa ra quyết định.
- **Máy học (Machine Learning – ML)**: là một nhánh cốt lõi của AI, trong đó các thuật toán cho phép máy tính tự cải thiện hiệu suất dựa trên dữ liệu thay vì lập trình thủ công.
- **Học sâu (Deep Learning – DL)**: mở rộng từ ML, sử dụng mạng nơ-ron nhiều lớp để xử lý dữ liệu phức tạp như hình ảnh, giọng nói, văn bản.
- **Xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP)**: tập trung vào việc giúp máy tính hiểu và sinh ngôn ngữ tự nhiên của con người, ứng dụng trong chatbot, dịch máy, phân tích cảm xúc.
- **Thị giác máy tính (Computer Vision – CV)**: nghiên cứu cách máy tính "nhìn" và phân tích hình ảnh, video, ứng dụng trong nhận diện khuôn mặt, xe tự lái, y học chẩn đoán hình ảnh.
- **Robot học (Robotics)**: kết hợp AI để xây dựng các hệ thống robot có khả năng nhận thức, di chuyển, và tương tác thông minh với môi trường.
- **Người–máy tương tác (Human–Computer Interaction – HCI)**: nghiên cứu cách thức con người giao tiếp với hệ thống AI một cách hiệu quả, trực quan và an toàn.

Có thể thấy, AI là một "chiếc ô công nghệ" rộng lớn, trong đó mỗi lĩnh vực liên quan vừa đóng vai trò hỗ trợ, vừa đồng thời hưởng lợi từ những tiến bộ của AI.

Hình ảnh minh họa

## 6.5 Vấn đề đạo đức và xã hội trong AI

Sự phát triển mạnh mẽ của trí tuệ nhân tạo không chỉ đem lại lợi ích kinh tế – xã hội mà còn đặt ra nhiều thách thức về đạo đức, pháp lý và trách nhiệm xã hội. Một số vấn đề quan trọng có thể kể đến:

### *6.5.1 Quyền riêng tư và bảo mật dữ liệu (Privacy & Data Security)*

- AI thường cần lượng dữ liệu lớn, trong đó có dữ liệu cá nhân.
- Việc thu thập, xử lý và lưu trữ dữ liệu cá nhân có thể dẫn đến vi phạm quyền riêng tư.
- Cần có cơ chế minh bạch và luật pháp để đảm bảo người dùng được bảo vệ.

### *6.5.2 Thiên lệch và công bằng (Bias & Fairness)*

- Thuật toán học từ dữ liệu, nếu dữ liệu chứa thiên lệch (bias) sẽ dẫn đến kết quả không công bằng.
- Ví dụ: AI tuyển dụng có thể vô tình phân biệt giới tính hoặc chủng tộc.
- Cần phát triển kỹ thuật phát hiện và giảm thiểu thiên lệch trong AI.

### *6.5.3 Trách nhiệm và tính minh bạch (Accountability & Transparency)*

- AI có thể đưa ra quyết định quan trọng (chẩn đoán bệnh, cấp tín dụng, ra quyết định pháp lý).
- Câu hỏi đặt ra: ai chịu trách nhiệm khi AI gây hậu quả sai lầm?
- Khái niệm "AI có thể giải thích" (Explainable AI – XAI) ngày càng được quan tâm.

### *6.5.4 Tác động đến việc làm và kinh tế (Employment & Economy Impact)*

- Tự động hóa bằng AI có thể thay thế nhiều công việc truyền thống.
- Điều này tạo ra lo ngại về thất nghiệp, đồng thời đòi hỏi con người phải học kỹ năng mới.
- AI cũng mở ra nhiều cơ hội việc làm trong lĩnh vực công nghệ cao.

### *6.5.5 An toàn và sử dụng sai mục đích (Safety & Misuse)*

- AI có thể bị lợi dụng trong giám sát hàng loạt, vũ khí tự động, tấn công mạng hoặc tạo thông tin giả (deepfake).

- Cần có chính sách và cơ chế quốc tế để quản lý việc phát triển và sử dụng AI.

### *6.5.6 Khía cạnh xã hội và triết học (Social & Philosophical Issues)*

- AI đặt ra câu hỏi về ranh giới giữa con người và máy móc.
- Liệu có nên coi AI có "quyền" như một thực thể nếu chúng đạt đến mức trí tuệ tương đương con người?
- Đây vẫn là vấn đề còn nhiều tranh luận.

AI không chỉ là vấn đề kỹ thuật mà còn gắn chặt với trách nhiệm xã hội, pháp luật và đạo đức. Người phát triển, nhà quản lý và cộng đồng cần hợp tác để xây dựng một nền tảng AI minh bạch, công bằng và có lợi cho toàn xã hội.

Case study tình huống về vấn đề đạo đức

## 6.6 Công cụ và Framework AI phổ biến

Để xây dựng và triển khai các ứng dụng trí tuệ nhân tạo, cộng đồng nghiên cứu và phát triển phần mềm đã xây dựng nhiều công cụ (tools) và bộ khung (frameworks) mạnh mẽ. Các công cụ này giúp rút ngắn thời gian phát triển, tối ưu hóa hiệu suất và hỗ trợ khả năng mở rộng [9], [10], [11]. Một số framework nổi bật:

### *6.6.1 TensorFlow*

- Được phát triển bởi Google, mã nguồn mở.
- Hỗ trợ học sâu (deep learning) và học máy (machine learning) với nhiều mô hình phức tạp.
- Tích hợp tốt với GPU/TPU, có khả năng triển khai trên nhiều nền tảng (máy chủ, di động, web).
- Thư viện con: **Keras** (API cấp cao) giúp xây dựng mô hình dễ dàng hơn.

### *6.6.2 PyTorch*

- Được phát triển bởi Facebook AI Research.
- Nổi bật với khả năng tính toán động (dynamic computation graph), dễ dàng thử nghiệm và nghiên cứu.
- Thân thiện với Python, được cộng đồng học thuật ưa chuộng.
- Được dùng rộng rãi trong các dự án nghiên cứu AI và sản phẩm công nghiệp.

### *6.6.3 Scikit-learn*

- Thư viện Python mạnh mẽ cho học máy truyền thống (machine learning).
- Cung cấp nhiều thuật toán: hồi quy, phân loại, gom cụm, giảm chiều dữ liệu.
- Dễ sử dụng, phù hợp cho người mới bắt đầu học machine learning.
- Tích hợp tốt với NumPy, Pandas và Matplotlib.

### *6.6.4 OpenCV (Open Source Computer Vision Library)*

- Thư viện mã nguồn mở cho thị giác máy tính (computer vision).
- Hỗ trợ xử lý ảnh, nhận diện khuôn mặt, theo dõi đối tượng, thị giác 3D.
- Được ứng dụng rộng rãi trong camera thông minh, robot, xe tự lái.

### *6.6.5 NLTK và spaCy*

- **NLTK (Natural Language Toolkit):** Bộ công cụ mạnh mẽ cho xử lý ngôn ngữ tự nhiên (NLP), hỗ trợ phân tích cú pháp, gán nhãn từ loại, dịch máy.
- **spaCy:** Hiệu năng cao, tập trung vào các ứng dụng NLP hiện đại như phân loại văn bản, trích xuất thực thể, chatbot.

### *6.6.6 Hugging Face Transformers*

- Thư viện chuyên về các mô hình ngôn ngữ hiện đại (transformer models) như BERT, GPT, T5.
- Dễ dàng tải và huấn luyện lại các mô hình ngôn ngữ mạnh mẽ cho các ứng dụng NLP.
- Cộng đồng hỗ trợ lớn, thường xuyên cập nhật.

### *6.6.7 Các nền tảng triển khai AI*

- **Google AI/Vertex AI, Microsoft Azure AI, Amazon AWS AI Services**: cung cấp dịch vụ AI trên đám mây, hỗ trợ huấn luyện, triển khai và giám sát mô hình.
- Phù hợp cho doanh nghiệp cần mở rộng và vận hành AI ở quy mô lớn.

Các công cụ và framework AI giúp rút ngắn quá trình nghiên cứu, phát triển và triển khai ứng dụng. Việc lựa chọn framework phụ thuộc vào mục tiêu dự án, loại dữ liệu và mức độ phức tạp của mô hình.

Bảng so sánh ngắn (ví dụ: TensorFlow vs PyTorch vs Scikit-learn)

## 6.7 Tóm tắt chương

Trong chương này, chúng ta đã tìm hiểu các khái niệm cơ bản của **trí tuệ nhân tạo (artificial intelligence – AI)**, cùng những đặc điểm giúp AI trở thành một lĩnh vực cốt lõi trong khoa học dữ liệu và công nghệ hiện đại. Chương đã trình bày:

- **Khái niệm và định nghĩa AI:** AI là khả năng của máy móc mô phỏng các hoạt động thông minh của con người, từ nhận thức, suy luận cho đến học tập và ra quyết định.
- **Ứng dụng thực tiễn:** AI hiện diện trong nhiều lĩnh vực, từ y tế, tài chính, giáo dục, giao thông cho đến thương mại điện tử và đời sống hằng ngày.
- **Các kỹ thuật và thuật toán nền tảng:** bao gồm học máy (machine learning), học sâu (deep learning), xử lý ngôn ngữ tự nhiên (natural language processing – NLP), và thị giác máy tính (computer vision).
- **Các lĩnh vực liên quan:** AI gắn bó chặt chẽ với khoa học dữ liệu, robot học, IoT và phân tích dữ liệu lớn.
- **Vấn đề đạo đức và xã hội:** sự minh bạch, công bằng, bảo mật dữ liệu và tác động đến việc làm là những thách thức cần được cân nhắc.
- **Công cụ và framework phổ biến:** TensorFlow, PyTorch, Scikit-learn, OpenCV, spaCy, Hugging Face cùng các nền tảng AI trên đám mây đã giúp việc nghiên cứu và ứng dụng AI trở nên dễ dàng hơn.

Nhìn chung, chương này cung cấp cho sinh viên cái nhìn nền tảng về AI, từ lý thuyết cơ bản đến công cụ thực hành, đồng thời nhấn mạnh các cơ hội và thách thức mà AI mang lại cho xã hội.

## 6.8 Câu hỏi và Bài tập

### *6.8.1 Câu hỏi ôn tập*

Câu hỏi 6.1. Trí tuệ nhân tạo (AI) là gì? Nó khác gì so với machine learning?

Câu hỏi 6.2. Hãy nêu ba ứng dụng tiêu biểu của AI trong đời sống.

Câu hỏi 6.3. Các kỹ thuật nền tảng của AI bao gồm những gì?

Câu hỏi 6.4. AI có mối quan hệ như thế nào với khoa học dữ liệu và big data?

Câu hỏi 6.5. Nêu một vấn đề đạo đức quan trọng mà AI đang đối mặt.

Câu hỏi 6.6. TensorFlow và PyTorch khác nhau chủ yếu ở điểm nào?

### *6.8.2 Hướng dẫn thực hành*

*Thực hành 6-1* ***Khám phá AI với Python***

**Mục tiêu**

- Giúp sinh viên hiểu sự khác biệt giữa chương trình điều kiện tĩnh (rule-based) và AI học từ dữ liệu.
- Làm quen với lập trình Python cơ bản thông qua một ví dụ vui nhộn.

***Bước 1: Tạo chatbot đơn giản***

```
# Thực hành 6.1 - Chatbot rule-based đơn giản

print("Chào bạn! Tôi là chatbot đơn giản.")
while True:
    user_input = input("Bạn: ").lower()
    if "chào" in user_input:
        print("Chatbot: Xin chào! Rất vui được gặp bạn.")
    elif "tên" in user_input:
        print("Chatbot: Tôi là chatbot rule-based, chưa thông
minh lắm.")
    elif "tạm biệt" in user_input:
        print("Chatbot: Hẹn gặp lại!")
        break
    else:
        print("Chatbot: Xin lỗi, tôi chưa hiểu ý bạn.")
```

Sinh viên chạy thử và thử nhập các câu khác nhau để thấy phản hồi.

***Bước 2: Phân tích & Thảo luận***

1. Đây có phải là AI không? (Gợi ý: đây chỉ là **if-else** do con người lập trình trước → không học từ dữ liệu.)
2. Điểm yếu của chatbot này là gì? (Không học thêm, không hiểu câu phức tạp.)
3. Nếu dùng AI/NLP thật sự, chatbot có thể học từ dữ liệu hội thoại để trả lời thông minh hơn.

***Bước 3: Mở rộng***

- Sinh viên thử bổ sung thêm 2–3 câu trả lời mới (ví dụ về thời tiết, môn học yêu thích).
- Thảo luận: nếu có **1000 câu trả lời**, liệu cách if-else này còn hiệu quả không? → Dẫn dắt sang chương Machine Learning & NLP.

## *Thực hành 6-2 Nâng cao: Chatbot AI đơn giản*

**Mục tiêu**

- Giúp sinh viên trải nghiệm một mô hình AI thực thụ, biết cách tải model có sẵn từ thư viện.
- So sánh chatbot AI với chatbot rule-based đã làm ở bước trước.

***Bước 1: Cài đặt thư viện cần thiết***

Trong Google Colab hoặc môi trường ảo, chạy:

```
pip install transformers torch
```

***Bước 2: Tạo chatbot AI với mô hình ngôn ngữ***

```
from transformers import pipeline

# Tải mô hình hội thoại (transformer model)
chatbot = pipeline("text-generation", model="gpt2")

while True:
    user_input = input("Bạn: ")
    if user_input.lower() in ["thoát", "tạm biệt"]:
        print("Chatbot: Hẹn gặp lại!")
        break

    response = chatbot(user_input, max_length=50,
do_sample=True, temperature=0.7)
    print("Chatbot:", response[0]['generated_text'])
```

**Điểm nhấn:** Mô hình sẽ sinh câu trả lời đa dạng, không phải chỉ là if-else cố định.

***Bước 3: So sánh và Thảo luận***

| Tiêu chí | Chatbot Rule-based | Chatbot AI (GPT-2) |
|---|---|---|
| **Cách hoạt động** | If-else do con người lập trình | Sinh câu dựa trên mô hình học từ dữ liệu lớn |
| **Khả năng mở rộng** | Khó mở rộng nếu nhiều kịch bản | Có thể trả lời linh hoạt, sáng tạo |
| **Nhược điểm** | Không học được, trả lời cứng nhắc | Đôi khi trả lời chưa chính xác, cần fine-tune |

***Bước 4: Thử mở rộng***

- Sinh viên thử hỏi các câu mở như:

  *"Bạn nghĩ gì về trí tuệ nhân tạo?"*

  *"Hãy viết một câu chuyện ngắn 1 câu."*

- Quan sát phản hồi để thấy tính sáng tạo.

## *Thực hành 6-3 Trải nghiệm Scikit-learn với phân loại cơ bản*

**Mục tiêu**

- Làm quen với quy trình xây dựng mô hình trong Scikit-learn.
- Thấy được cách mô hình học từ dữ liệu và dự đoán kết quả mới.

***Bước 1: Chuẩn bị dữ liệu mẫu***

```
import pandas as pd

# Tạo dataset mẫu (chiều cao, cân nặng, giới tính)
data = {
    'chieu_cao': [150, 160, 165, 170, 175, 180, 185],
    'can_nang': [45, 55, 60, 65, 70, 80, 85],
    'gioi_tinh': ['Nữ','Nữ','Nữ','Nam','Nam','Nam','Nam']
}

df = pd.DataFrame(data)
print(df)
```

***Bước 2: Tiền xử lý dữ liệu***

Chuyển biến phân loại (giới tính) thành nhãn số:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['gioi_tinh'] = le.fit_transform(df['gioi_tinh'])  # Nữ=0,
Nam=1
print(df)
```

***Bước 3: Chia dữ liệu train/test***

```
from sklearn.model_selection import train_test_split

X = df[['chieu_cao','can_nang']]
y = df['gioi_tinh']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

***Bước 4: Huấn luyện mô hình kNN***

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
```

***Bước 5: Dự đoán và đánh giá***

```
y_pred = model.predict(X_test)
from sklearn.metrics import accuracy_score

print("Dự đoán:", y_pred)
print("Thực tế:", y_test.values)
print("Độ chính xác:", accuracy_score(y_test, y_pred))
```

***Bước 6: Thử dự đoán cho người mới***

```
new_data = [[172, 68]]
result = model.predict(new_data)
print("Dự đoán giới tính cho (172 cm, 68 kg):",
le.inverse_transform(result)[0])
```

***Bước 7: Thảo luận***

- Sinh viên quan sát độ chính xác thay đổi khi điều chỉnh n_neighbors (k = 1, 3, 5).
- Thử thêm dữ liệu mới để xem mô hình dự đoán hợp lý không.
- Nhấn mạnh vai trò của train/test split và việc đánh giá mô hình.

## *Thực hành 6-4 Nâng cao: Phân loại hoa Iris với Scikit-learn*

**Mục tiêu**

- Làm quen với dataset kinh điển Iris (150 mẫu, 3 loại hoa).
- Thực hành toàn bộ quy trình ML: train/test split, huấn luyện, đánh giá, trực quan hóa.

***Bước 1: Nạp dữ liệu***

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target

print(df.head())
print(df['species'].value_counts())
```

***Bước 2: Chia train/test***

```
from sklearn.model_selection import train_test_split

X = df[iris.feature_names]  # 4 đặc trưng: sepal & petal
y = df['species']           # nhãn (0,1,2)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

***Bước 3: Huấn luyện mô hình kNN***

```
from sklearn.neighbors import KNeighborsClassifier
```

```
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
```

***Bước 4: Đánh giá mô hình***

```
from sklearn.metrics import accuracy_score,
classification_report

y_pred = model.predict(X_test)

print("Độ chính xác:", accuracy_score(y_test, y_pred))
print("Báo cáo phân loại:\n", classification_report(y_test,
y_pred, target_names=iris.target_names))
```

***Bước 5: Trực quan hóa kết quả***

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6,5))
sns.scatterplot(x='petal length (cm)', y='petal width (cm)',
                data=df, hue='species', palette='Set1')
plt.title('Phân bố loài hoa theo kích thước cánh hoa')
plt.show()
```

***Bước 6: Thử dự đoán mới***

```
sample = [[5.0, 3.4, 1.5, 0.2]]  # Một mẫu mới
prediction = model.predict(sample)
print("Dự đoán loài hoa:", iris.target_names[prediction][0])
```

***Bước 7: Thảo luận***

- Kiểm tra độ chính xác khi thay đổi n_neighbors (3, 7, 9).
- So sánh kết quả dự đoán giữa các loài (setosa, versicolor, virginica).
- Thảo luận ý nghĩa trực quan của scatter plot – nhận thấy các cụm tách biệt khá rõ theo petal length & width.

## *Thực hành 6-5* ***Ứng dụng Computer Vision cơ bản***

**Mục tiêu**

- Làm quen với thư viện OpenCV.
- Hiểu cách xử lý ảnh số: đọc ảnh, chuyển đổi màu, phát hiện cạnh.
- Quan sát sự khác biệt giữa ảnh gốc và ảnh đã xử lý.

***Bước 1: Cài đặt OpenCV***

Trong Google Colab hoặc môi trường Python:

```
pip install opencv-python matplotlib
```

***Bước 2: Đọc và hiển thị ảnh***

```
import cv2
import matplotlib.pyplot as plt

# Đọc ảnh (dùng ảnh có sẵn hoặc tải từ internet)
img = cv2.imread('sample_image.jpg')  # Đảm bảo ảnh nằm trong
thư mục làm việc

# Chuyển từ BGR (OpenCV) sang RGB (Matplotlib)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb)
plt.title('Ảnh gốc')
plt.axis('off')
plt.show()
```

***Bước 3: Chuyển ảnh sang thang xám***

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.imshow(gray, cmap='gray')
plt.title('Ảnh xám')
plt.axis('off')
plt.show()
```

***Bước 4: Phát hiện cạnh bằng Canny Edge Detection***

```
edges = cv2.Canny(gray, 100, 200)
```

```
plt.imshow(edges, cmap='gray')
plt.title('Phát hiện cạnh (Canny)')
plt.axis('off')
plt.show()
```

***Bước 5: Làm mờ ảnh (Gaussian Blur)***

```
blurred = cv2.GaussianBlur(img_rgb, (15, 15), 0)

plt.imshow(blurred)
plt.title('Ảnh sau khi làm mờ')
plt.axis('off')
plt.show()
```

***Bước 6: Thảo luận***

- Quan sát sự khác biệt giữa ảnh gốc, ảnh xám, ảnh phát hiện cạnh.
- Hỏi sinh viên: khi tăng/giảm ngưỡng trong cv2.Canny, ảnh thay đổi thế nào?
- Giải thích ứng dụng thực tế: phát hiện biên trong nhận diện vật thể, xử lý ảnh y tế, robot tự hành.

**Mở rộng (Tùy chọn)**

- Thử đọc ảnh từ webcam (nếu có):

```
cap = cv2.VideoCapture(0)
ret, frame = cap.read()
cap.release()

plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
plt.title('Ảnh chụp từ webcam')
plt.axis('off')
plt.show()
```

- Áp dụng **thresholding** để nhị phân hóa ảnh.
- Thử vẽ bounding box quanh các cạnh tìm được.

*Thực hành 6-6* ***Phát hiện vật thể với mô hình pre-trained***

**Mục tiêu**

- Giúp sinh viên thấy Computer Vision không chỉ dừng lại ở xử lý ảnh cơ bản mà còn có thể **hiểu nội dung ảnh**.
- Làm quen với khái niệm mô hình pre-trained (đã huấn luyện trước) trong deep learning.
- Thấy ngay kết quả thực tế mà không cần huấn luyện mô hình từ đầu.

***Bước 1: Cài đặt thư viện***

Chạy trong Colab:

```
pip install opencv-python opencv-python-headless matplotlib
torch torchvision
```

***Bước 2: Tải mô hình YOLOv5 (hoặc dùng torchvision pre-trained model)***

Ví dụ YOLOv5 từ Ultralytics:

```
import torch

# Tải mô hình YOLOv5 nhỏ (yolov5s)
model = torch.hub.load('ultralytics/yolov5', 'yolov5s',
pretrained=True)
```

***Bước 3: Chọn ảnh đầu vào***

```
img_path = 'sample_image.jpg'  # Đảm bảo có ảnh trong thư mục
results = model(img_path)
results.print()  # In nhãn các đối tượng tìm thấy
results.show()   # Hiển thị ảnh có bounding boxes
```

***Bước 4: Phân tích kết quả***

- YOLOv5 trả về ảnh có vẽ bounding box, tên đối tượng và xác suất dự đoán.
- Sinh viên sẽ thấy mô hình nhận diện được người, xe, chó mèo… (tùy ảnh).
- Có thể thử với ảnh khác hoặc ảnh do sinh viên tải lên.

***Bước 5: Mở rộng tùy chọn***

- Thử nhiều ảnh khác nhau (ảnh đường phố, lớp học, sản phẩm).

- Quan sát mô hình nhận diện tốt/không tốt với từng loại ảnh → thảo luận về độ chính xác và giới hạn của mô hình.
- Hỏi sinh viên: nếu muốn mô hình nhận diện sản phẩm trong cửa hàng, cần làm gì? (gợi ý về fine-tuning, thu thập dữ liệu riêng).

## *Thực hành 6-7 NLP đơn giản với Hugging Face*

**Mục tiêu**

- Làm quen với **transformer models** (BERT, DistilBERT, GPT, v.v.) trong xử lý ngôn ngữ tự nhiên.
- Trải nghiệm pipeline của Hugging Face để **phân tích cảm xúc (sentiment analysis)** hoặc **dịch văn bản**.
- Hiểu cách ứng dụng mô hình có sẵn (pre-trained models) vào bài toán thực tế mà không cần huấn luyện.

### *Bước 1: Cài đặt thư viện*

Chạy trong Google Colab hoặc Jupyter:

```
pip install transformers datasets
```

### *Bước 2: Tạo pipeline phân tích cảm xúc*

```
from transformers import pipeline

# Tạo pipeline phân tích cảm xúc
classifier = pipeline("sentiment-analysis")

# Văn bản mẫu
texts = [
    "I love this product! It's really amazing.",
    "This is the worst service I've ever experienced."
]

results = classifier(texts)
for text, result in zip(texts, results):
    print(f"Text: {text}\nSentiment: {result['label']}, Score:
{result['score']:.4f}\n")
```

**Kết quả**: Trả về nhãn POSITIVE hoặc NEGATIVE cùng độ tin cậy.

***Bước 3: Thử nghiệm dịch máy***

```
translator = pipeline("translation", model="Helsinki-NLP/opus-
mt-en-vi")

text = "Machine learning is transforming the world of
technology."
translation = translator(text, max_length=100)
print("Original:", text)
print("Translated:", translation[0]['translation_text'])
```

**Kết quả**: Chuỗi tiếng Anh được dịch sang tiếng Việt bằng mô hình dịch máy.

***Bước 4: Thực hành mở rộng***

- Thử với **danh sách 5–10 câu** tiếng Anh do sinh viên tự chọn.
- Thử với văn bản tiếng Việt để xem mô hình có hiểu không (và thảo luận tại sao).
- So sánh cảm xúc giữa các câu tích cực, tiêu cực, trung tính.

***Bước 5: Diễn giải và thảo luận***

- **Điểm mạnh**: Không cần huấn luyện, áp dụng được ngay.
- **Hạn chế**: Đôi khi mô hình có thiên lệch văn hóa hoặc kết quả dịch chưa chính xác.
- **Ứng dụng thực tế**: phân tích review sản phẩm, lọc bình luận tiêu cực, chatbot trả lời tự động.

***Bước 6: Gợi ý mở rộng bài tập mini***

1. Thu thập 10 review sản phẩm từ một trang TMĐT, chạy sentiment analysis và thống kê % đánh giá tích cực/tiêu cực.
2. Viết một đoạn code dịch tự động 5 câu từ tiếng Việt sang tiếng Anh (dùng mô hình khác như Helsinki-NLP/opus-mt-vi-en).
3. So sánh kết quả với Google Translate, thảo luận ưu/nhược điểm.

### *6.8.3 Case Study*

#### *6.8.3.1 Case Study A – Phân tích cảm xúc đánh giá sản phẩm*

**Mục tiêu**

- Kết hợp kiến thức EDA (Chương 4) và NLP (Chương 6).
- Áp dụng **Hugging Face pipeline** để phân tích cảm xúc review sản phẩm.
- Trực quan hóa kết quả bằng biểu đồ để rút ra insight.

***Bước 1: Chuẩn bị dữ liệu***

Tạo file reviews.csv hoặc nhập trực tiếp:

```
import pandas as pd

data = {
    "review": [
        "I love this product! It's really amazing.",
        "The quality is very bad, totally disappointed.",
        "Good value for money, will buy again.",
        "Terrible customer service, never coming back!",
        "Excellent packaging and fast delivery."
    ]
}

df = pd.DataFrame(data)
df.to_csv("reviews.csv", index=False)
print(df)
```

***Bước 2: Phân tích cảm xúc với Hugging Face***

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")

df['sentiment'] = df['review'].apply(lambda x:
classifier(x)[0]['label'])
df['score'] = df['review'].apply(lambda x:
classifier(x)[0]['score'])
print(df)
```

***Bước 3: Trực quan hóa kết quả***

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='sentiment', data=df, palette='Set2')
plt.title("Phân bố cảm xúc review sản phẩm")
plt.xlabel("Loại cảm xúc")
plt.ylabel("Số lượng")
plt.show()
```

***Bước 4: Diễn giải kết quả***

- Đếm số review tích cực/tiêu cực → có thể tính %:

  print(df['sentiment'].value_counts(normalize=True) * 100)

- Nếu phần lớn review là tiêu cực → đề xuất cải thiện chất lượng, dịch vụ.
- Nếu review tích cực cao → có thể đẩy mạnh marketing, upsell.

***Bước 5: Mở rộng bài tập***

1. **Thu thập dữ liệu thực tế**: yêu cầu sinh viên lấy ít nhất 20 review từ một sản phẩm trên Shopee/Lazada/Amazon.
2. **Phân tích sâu hơn**: nhóm review theo POSITIVE/NEGATIVE, tính điểm trung bình.
3. **Trực quan hóa nâng cao**: vẽ pie chart hoặc bar chart theo % review tích cực/tiêu cực.
4. **So sánh mô hình**: dùng một mô hình khác như distilbert-base-multilingual-cased để xem kết quả khác nhau thế nào.

### *6.8.3.2 Case Study B – Xây dựng Hệ thống Gợi ý Sản Phẩm Mini*

**Mục tiêu học tập**

- Hiểu **cách hoạt động của hệ thống gợi ý** (recommendation system).
- Áp dụng kỹ thuật **content-based filtering** đơn giản với dữ liệu nhỏ.
- Thực hành tiền xử lý dữ liệu, tính toán độ tương tự (similarity), và xuất ra danh sách gợi ý.
- Trực quan hóa kết quả và thảo luận về ưu/nhược điểm.

Một cửa hàng online muốn xây dựng hệ thống gợi ý sản phẩm cho khách hàng. Khi người dùng xem một sản phẩm, hệ thống sẽ gợi ý thêm các sản phẩm tương tự về mô tả hoặc thể loại để tăng doanh số.

Dữ liệu mẫu (products.csv) có các cột:

- product_id – mã sản phẩm
- product_name – tên sản phẩm
- category – thể loại
- description – mô tả sản phẩm

***Bước 1: Chuẩn bị dữ liệu***

Tạo file CSV hoặc nhập trực tiếp:

```
import pandas as pd


data = {
    "product_id": [1, 2, 3, 4, 5],
    "product_name": ["Laptop A", "Laptop B", "Headphone C", "Keyboard D", "Mouse E"],
    "category": ["Laptop", "Laptop", "Accessory", "Accessory", "Accessory"],
    "description": [
        "Lightweight laptop with Intel i5 CPU and 8GB RAM",
        "Powerful laptop with Intel i7 CPU and 16GB RAM",
        "Wireless over-ear headphone with noise cancelling",
        "Mechanical keyboard with RGB lighting",
        "Ergonomic wireless mouse with fast response"
    ]
}


df = pd.DataFrame(data)
```

```
df.to_csv("products.csv", index=False)
print(df)
```

***Bước 2: Biểu diễn văn bản (Text Vectorization)***

Sử dụng TF-IDF để biến đổi mô tả sản phẩm thành vector số:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['description'])
```

***Bước 3: Tính độ tương tự giữa các sản phẩm***

Sử dụng cosine similarity:

```
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

***Bước 4: Hàm gợi ý sản phẩm***

```
def recommend(product_name, top_n=3):
    idx = df[df['product_name'] == product_name].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:top_n+1]  # Bỏ sản phẩm chính
    recommended_products = df.iloc[[i[0] for i in sim_scores]]
    return recommended_products[['product_name', 'category']]

print("Gợi ý cho Laptop A:")
print(recommend("Laptop A"))
```

***Bước 5: Diễn giải kết quả***

- Sinh viên quan sát các sản phẩm gợi ý, kiểm tra tính hợp lý.
- Thảo luận: hệ thống đang dựa trên **mô tả sản phẩm** (content-based), không cần dữ liệu người dùng.
- Mở rộng: có thể kết hợp dữ liệu tương tác (lượt mua, lượt xem) → **hybrid recommendation**.

***Bước 6: Mở rộng bài tập***

1. **Thêm dữ liệu**: nhập 10–20 sản phẩm, đa dạng hơn về category.
2. **Trực quan hóa**: vẽ heatmap của ma trận similarity để quan sát nhóm sản phẩm gần nhau.
3. **Gợi ý nâng cao**: lọc gợi ý theo cùng category hoặc theo mức giá tương đồng.
4. **Mini-project**: xây dựng giao diện đơn giản (streamlit) cho phép chọn sản phẩm và nhận gợi ý trực tiếp.

# CHƯƠNG 7 ỨNG DỤNG, XU HƯỚNG VÀ THÁCH THỨC

Trong chương này, chúng ta sẽ tìm hiểu những ứng dụng nổi bật của khoa học dữ liệu, trí tuệ nhân tạo và học máy trong nhiều lĩnh vực khác nhau, từ kinh doanh, quản trị, khoa học, kỹ thuật cho đến đời sống hàng ngày. Đồng thời, chương cũng giới thiệu những xu hướng công nghệ mới đang định hình tương lai của lĩnh vực này, chẳng hạn như học sâu (deep learning), AI sinh tạo (generative AI), điện toán biên (edge computing), và tích hợp dữ liệu lớn với IoT. Bên cạnh đó, chúng ta cũng bàn đến các thách thức và những vấn đề đạo đức, pháp lý, xã hội khi áp dụng các công nghệ dữ liệu và AI trong thực tiễn.

Sau khi học xong chương này, sinh viên có thể:

- Nhận diện được các ứng dụng tiêu biểu của khoa học dữ liệu và AI trong các lĩnh vực khác nhau.
- Hiểu được những xu hướng công nghệ mới đang tác động đến lĩnh vực phân tích dữ liệu và AI.
- Phân tích được những thách thức, hạn chế và rủi ro đạo đức liên quan đến việc ứng dụng AI và dữ liệu.
- Liên hệ thực tế và đưa ra ví dụ ứng dụng trong lĩnh vực học tập hoặc công việc quan tâm.

## 7.1 Ứng dụng trong kinh doanh và quản trị

Khoa học dữ liệu và trí tuệ nhân tạo ngày càng đóng vai trò then chốt trong việc hỗ trợ các doanh nghiệp ra quyết định, tối ưu hóa hoạt động và nâng cao trải nghiệm khách hàng. Một số ứng dụng tiêu biểu có thể kể đến:

- **Phân tích khách hàng và thị trường**: Doanh nghiệp có thể khai thác dữ liệu giao dịch, hành vi trực tuyến và phản hồi của khách hàng để phân khúc thị trường, dự đoán xu hướng tiêu dùng, từ đó xây dựng chiến lược marketing và chăm sóc khách hàng hiệu quả hơn.
- **Dự báo và quản trị rủi ro**: Trong lĩnh vực tài chính – ngân hàng, các mô hình học máy được sử dụng để phát hiện gian lận (fraud detection), dự báo khả năng vỡ nợ (credit scoring) hoặc phân tích rủi ro đầu tư.
- **Quản lý chuỗi cung ứng (Supply Chain Management)**: Phân tích dữ liệu giúp tối ưu hóa tồn kho, dự báo nhu cầu, lên kế hoạch vận chuyển và giảm chi phí logistics.

- **Tự động hóa quy trình (Robotic Process Automation – RPA)**: AI có thể thay thế hoặc hỗ trợ con người trong các tác vụ lặp lại như nhập liệu, xử lý hóa đơn, kiểm tra chứng từ, giúp giảm thiểu sai sót và tiết kiệm thời gian.
- **Quyết định chiến lược dựa trên dữ liệu (Data-driven Decision Making)**: Nhà quản lý không còn chỉ dựa vào kinh nghiệm hay trực giác, mà có thể khai thác insight từ dữ liệu lớn để đưa ra quyết định có cơ sở khoa học hơn.

Những ứng dụng trên cho thấy dữ liệu và AI đang trở thành một phần không thể thiếu trong hoạt động quản trị và cạnh tranh của doanh nghiệp hiện đại.

## 7.2 Ứng dụng trong khoa học và kỹ thuật

Trong lĩnh vực khoa học và kỹ thuật, dữ liệu và trí tuệ nhân tạo (AI) không chỉ giúp xử lý khối lượng thông tin khổng lồ mà còn mở ra những phương pháp nghiên cứu mới, nhanh và chính xác hơn. Một số ứng dụng nổi bật gồm:

- **Y học và chăm sóc sức khỏe**: AI hỗ trợ chẩn đoán hình ảnh y khoa (X-quang, MRI, CT scan), phát hiện sớm bệnh lý, phân tích dữ liệu gen (genomics) để cá nhân hóa phác đồ điều trị, hay hỗ trợ nghiên cứu thuốc mới thông qua mô phỏng và dự báo hiệu quả.
- **Khoa học môi trường**: Các mô hình dự báo khí hậu, chất lượng không khí, thiên tai (bão, lũ lụt, hạn hán) đều dựa vào phân tích dữ liệu lớn và AI để tăng độ chính xác và hỗ trợ ra quyết định ứng phó.
- **Năng lượng và kỹ thuật công nghiệp**: Dữ liệu từ cảm biến (IoT) giúp theo dõi hoạt động của nhà máy, dự báo sự cố, tối ưu hóa tiêu thụ năng lượng và tăng hiệu suất sản xuất. Trong ngành năng lượng tái tạo, AI được dùng để dự đoán sản lượng điện gió, điện mặt trời và điều phối lưới điện thông minh.
- **Vật lý, hóa học và sinh học**: Phân tích dữ liệu thí nghiệm và mô phỏng trên siêu máy tính giúp các nhà khoa học rút ngắn thời gian nghiên cứu, từ khám phá vật liệu mới đến hiểu rõ hơn về cấu trúc phân tử và phản ứng sinh học.
- **Kỹ thuật giao thông và xây dựng**: Các hệ thống giao thông thông minh (ITS – Intelligent Transportation Systems) ứng dụng AI để dự báo lưu lượng, quản lý tín hiệu đèn giao thông, tối ưu hóa vận tải công cộng và hỗ trợ phát triển đô thị thông minh.

Nhờ sự hỗ trợ của dữ liệu lớn và AI, khoa học và kỹ thuật có thể tiến xa hơn trong việc giải quyết các vấn đề phức tạp mà trước đây khó tiếp cận, đồng thời tạo ra những đổi mới mang tính đột phá.

## 7.3 Ứng dụng trong đời sống

Trong đời sống hằng ngày, dữ liệu và trí tuệ nhân tạo (AI) ngày càng hiện diện rõ rệt, mang lại sự tiện lợi, cá nhân hóa trải nghiệm và hỗ trợ con người trong nhiều hoạt động thường nhật. Một số ứng dụng tiêu biểu:

- **Trợ lý ảo và dịch vụ thông minh**: Các trợ lý ảo như Siri, Google Assistant, Alexa hay ChatGPT hỗ trợ trả lời câu hỏi, nhắc lịch, tìm kiếm thông tin và điều khiển thiết bị thông minh trong gia đình.
- **Mạng xã hội và giải trí số**: AI phân tích sở thích của người dùng để gợi ý bạn bè, video, âm nhạc, hoặc nội dung phù hợp. Các nền tảng như Netflix, Spotify, YouTube đều ứng dụng AI trong hệ thống gợi ý.
- **Mua sắm trực tuyến**: Thương mại điện tử tận dụng AI để cá nhân hóa sản phẩm gợi ý, tối ưu quảng cáo và cung cấp chatbot hỗ trợ khách hàng 24/7.
- **Giáo dục và học tập trực tuyến**: Các nền tảng học tập sử dụng AI để đề xuất khóa học, chấm điểm tự động, đưa ra lộ trình học cá nhân hóa và thậm chí mô phỏng giảng viên ảo.
- **Giao thông và di chuyển**: Ứng dụng bản đồ thông minh (Google Maps, Grab, Be) sử dụng AI để dự báo tắc đường, gợi ý tuyến đi tối ưu, ước lượng thời gian và chi phí.
- **Sức khỏe và thể chất**: Thiết bị đeo thông minh (smartwatch, fitness tracker) theo dõi nhịp tim, giấc ngủ, số bước đi, lượng calo tiêu thụ, đồng thời đưa ra lời khuyên cải thiện sức khỏe.

Có thể thấy, AI và dữ liệu đang len lỏi vào nhiều khía cạnh của đời sống, từ học tập, làm việc, vui chơi đến chăm sóc sức khỏe, tạo nên một môi trường sống tiện nghi và hiện đại hơn.

## 7.4 Xu hướng công nghệ mới

Sự phát triển nhanh chóng của dữ liệu lớn (big data), trí tuệ nhân tạo (AI) và các công nghệ liên quan đã tạo ra nhiều xu hướng công nghệ mới, có ảnh hưởng sâu rộng đến kinh tế, xã hội và đời sống [17]. Một số xu hướng nổi bật:

- **Trí tuệ nhân tạo tổng hợp (generative AI)**: Các mô hình như ChatGPT, DALL·E, Stable Diffusion có khả năng tạo ra văn bản, hình ảnh, âm nhạc và video mới, mở ra tiềm năng ứng dụng trong sáng tạo nội dung, thiết kế, giáo dục và giải trí.
- **Điện toán đám mây kết hợp điện toán biên (cloud & edge computing)**: Cho phép xử lý dữ liệu vừa linh hoạt (trên đám mây) vừa nhanh chóng (gần nguồn dữ liệu), đặc biệt quan trọng trong IoT, xe tự lái, và thành phố thông minh.
- **Học máy tự động (automated machine learning - AutoML)**: Giúp đơn giản hóa quy trình xây dựng mô hình, từ tiền xử lý dữ liệu, chọn đặc trưng, đến huấn luyện và tối ưu mô hình, giúp ngay cả những người không chuyên cũng có thể khai thác sức mạnh AI.
- **Trí tuệ nhân tạo giải thích được (explainable AI - XAI)**: Hướng đến việc làm rõ cách thức mô hình đưa ra quyết định, nhằm tăng tính minh bạch, độ tin cậy và chấp nhận của người dùng trong các lĩnh vực nhạy cảm như y tế, tài chính.
- **Ứng dụng AI trong y tế chính xác (precision medicine)**: Kết hợp dữ liệu gen, bệnh án và thói quen sinh hoạt để đề xuất phác đồ điều trị phù hợp với từng cá nhân.
- **Tích hợp AI với Internet vạn vật (AIoT)**: Kết hợp AI và IoT để tạo ra hệ thống tự động thông minh, từ nhà thông minh, nhà máy thông minh, đến hạ tầng đô thị thông minh.
- **Robot thế hệ mới và xe tự hành**: Ngày càng được ứng dụng trong sản xuất, dịch vụ và vận tải, hứa hẹn thay đổi căn bản ngành công nghiệp và giao thông trong tương lai.

Các xu hướng này không chỉ thúc đẩy đổi mới sáng tạo mà còn đặt ra yêu cầu cấp thiết về quản trị, đạo đức và pháp lý để đảm bảo công nghệ phát triển theo hướng bền vững và có lợi cho con người.

## 7.5 Thách thức và vấn đề đạo đức

Bên cạnh những cơ hội to lớn, việc ứng dụng dữ liệu lớn và trí tuệ nhân tạo cũng đặt ra nhiều thách thức quan trọng về mặt kỹ thuật, xã hội và đạo đức [12], [13].

- **Chất lượng dữ liệu**: Dữ liệu thường bị thiếu, nhiễu hoặc thiên lệch (bias). Nếu không xử lý tốt, mô hình học máy có thể đưa ra kết quả sai lệch, ảnh hưởng nghiêm trọng đến quyết định thực tế.
- **Quyền riêng tư và bảo mật dữ liệu (privacy & security)**: Việc thu thập và khai thác dữ liệu cá nhân trong thương mại, y tế, tài chính… làm dấy lên lo ngại về quyền riêng tư và nguy cơ bị lạm dụng.
- **Thiên kiến thuật toán (algorithmic bias)**: Các mô hình có thể phản ánh hoặc khuếch đại định kiến xã hội nếu dữ liệu huấn luyện vốn đã thiên lệch, dẫn đến phân biệt đối xử trong tuyển dụng, cho vay, hay xét xử pháp luật.
- **Minh bạch và khả năng giải thích (explainability)**: Nhiều mô hình AI phức tạp như deep learning hoạt động như "hộp đen", khó giải thích cho người dùng hoặc cơ quan quản lý, từ đó gây khó khăn trong việc kiểm chứng và chấp nhận.
- **Tác động đến việc làm**: Tự động hóa và robot thông minh có thể thay thế một số ngành nghề, đặc biệt là các công việc lặp lại, dẫn đến lo ngại về thất nghiệp và bất bình đẳng xã hội.
- **Trách nhiệm pháp lý và đạo đức**: Khi AI đưa ra quyết định sai lầm (ví dụ trong y tế hoặc xe tự lái), câu hỏi đặt ra là ai sẽ chịu trách nhiệm – nhà phát triển, người sử dụng, hay cơ quan quản lý?
- **Nguy cơ lạm dụng AI**: Công nghệ AI có thể bị lợi dụng trong việc tạo tin giả (deepfake), thao túng dư luận, hoặc phát triển vũ khí tự động, gây ra rủi ro lớn cho an ninh và ổn định toàn cầu.

Để giải quyết những vấn đề này, các tổ chức quốc tế và chính phủ nhiều nước đang xây dựng **khung pháp lý và nguyên tắc phát triển AI có trách nhiệm (responsible AI)**, nhấn mạnh vào minh bạch, công bằng, bảo vệ quyền riêng tư, và ưu tiên lợi ích của con người.

## 7.6 Tóm tắt chương

Trong chương này, chúng ta đã tìm hiểu những ứng dụng rộng rãi của khoa học dữ liệu và trí tuệ nhân tạo trong nhiều lĩnh vực: **kinh doanh và quản trị**, **khoa học – kỹ thuật**, cũng như **đời sống hàng ngày**. Bên cạnh đó, chương cũng giới thiệu các **xu hướng công nghệ mới** như học sâu, điện toán biên, AI sinh sinh (generative AI), và phân tích dữ liệu theo thời gian thực, vốn đang mở ra nhiều cơ hội đổi mới sáng tạo.

Tuy nhiên, việc ứng dụng dữ liệu lớn và AI cũng đi kèm với **những thách thức quan trọng**, bao gồm chất lượng dữ liệu, quyền riêng tư, thiên kiến thuật toán, minh bạch, tác động xã hội và nguy cơ lạm dụng. Điều này cho thấy rằng để phát triển bền vững, AI cần được quản lý và ứng dụng theo hướng **có trách nhiệm, minh bạch và lấy con người làm trung tâm**.

## 7.7 Câu hỏi và Bài tập

### *7.7.1 Câu hỏi ôn tập*

1. Hãy nêu ba ứng dụng tiêu biểu của dữ liệu lớn và trí tuệ nhân tạo trong **kinh doanh và quản trị**.
2. Phân tích lợi ích của việc ứng dụng AI trong **khoa học – kỹ thuật**.
3. Trình bày một ví dụ cụ thể về AI trong **đời sống hàng ngày**.
4. Các xu hướng công nghệ mới như **AI sinh sinh (Generative AI)**, **điện toán biên (Edge Computing)** có ý nghĩa gì đối với sự phát triển của khoa học dữ liệu?
5. Thách thức đạo đức và xã hội nào được coi là quan trọng nhất trong ứng dụng AI hiện nay?

### *7.7.2 Hướng dẫn thực hành*

**Case Study 7A: Ứng dụng AI trong thương mại điện tử**

Một công ty thương mại điện tử muốn triển khai hệ thống gợi ý sản phẩm dựa trên hành vi mua sắm của khách hàng. Sinh viên hãy:

- Xác định các nguồn dữ liệu cần thiết (ví dụ: lịch sử giao dịch, hành vi duyệt web, phản hồi của khách hàng).

- Đề xuất mô hình AI phù hợp (ví dụ: hệ thống gợi ý dựa trên collaborative filtering).
- Thảo luận lợi ích và những rủi ro có thể phát sinh (quyền riêng tư, thiên kiến dữ liệu…).

**Case Study 7B: AI trong y tế**

Một bệnh viện áp dụng hệ thống hỗ trợ chẩn đoán bệnh dựa trên hình ảnh y khoa. Hãy thảo luận:

- Quy trình thu thập và xử lý dữ liệu y tế.
- Các lợi ích mà hệ thống mang lại cho bác sĩ và bệnh nhân.
- Những thách thức đạo đức cần lưu ý khi ứng dụng (ví dụ: bảo mật dữ liệu, trách nhiệm khi hệ thống sai sót).

### *7.7.3 Bài tập thực hành*

1. Tìm một ví dụ thực tế (ở Việt Nam hoặc quốc tế) về ứng dụng AI trong kinh doanh, khoa học hoặc đời sống. Viết báo cáo ngắn (1–2 trang) phân tích lợi ích và hạn chế.
2. Lập bảng so sánh ba xu hướng công nghệ mới trong AI (ví dụ: Generative AI, Edge Computing, AI explainability) về: đặc điểm, lợi ích, thách thức.
3. Thảo luận nhóm: “Làm thế nào để cân bằng giữa đổi mới công nghệ và bảo đảm đạo đức trong ứng dụng AI?”. Viết ra 3–5 giải pháp cụ thể.

# TÀI LIỆU THAM KHẢO

[1] J. VanderPlas, *Python Data Science Handbook*. O'Reilly Media, 2016.

[2] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2nd ed., 2019.

[3] W. McKinney, *Python for Data Analysis*. O'Reilly Media, 2nd ed., 2017.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd ed., 2009.

[7] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.

[8] Kaggle, "Datasets and Competitions," [Online]. Available: https://www.kaggle.com/. [Accessed: 11-Sep-2025].

[9] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/. [Accessed: 11-Sep-2025].

[10] TensorFlow Developers, "TensorFlow," [Online]. Available: https://www.tensorflow.org/. [Accessed: 11-Sep-2025].

[11] PyTorch Developers, "PyTorch," [Online]. Available: https://pytorch.org/. [Accessed: 11-Sep-2025].

[12] UNESCO, "Recommendation on the Ethics of Artificial Intelligence," [Online]. Available: https://unesdoc.unesco.org/. [Accessed: 11-Sep-2025].

[13] European Commission, "Artificial Intelligence Act," [Online]. Available: https://artificialintelligenceact.eu/. [Accessed: 11-Sep-2025].

[14] J. D. Hunter et al., "Matplotlib: Visualization with Python," [Online]. Available: https://matplotlib.org/. [Accessed: 11-Sep-2025].

[15] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

[16] McKinsey Global Institute, "The State of AI," McKinsey & Company, 2023. [Online]. Available: https://www.mckinsey.com/. [Accessed: 11-Sep-2025].

[17] Gartner, "Top Strategic Technology Trends," Gartner, 2024. [Online]. Available: https://www.gartner.com/. [Accessed: 11-Sep-2025].

# PHỤ LỤC A: PYTHON CƠ BẢN

## A.1. Giới thiệu ngôn ngữ Python

- Python là ngôn ngữ lập trình thông dịch, đa dụng, dễ học, cú pháp đơn giản.
- Được dùng rộng rãi trong khoa học dữ liệu, trí tuệ nhân tạo, web, tự động hóa.
- Cộng đồng lớn, nhiều thư viện mạnh mẽ (NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow...).

## A.2. Kiểu dữ liệu cơ bản

- **Số (Numbers)**: int, float
- **Chuỗi (String)**: "Hello"
- **Danh sách (List)**: [1, 2, 3]
- **Tuple**: (1, 2, 3)
- **Từ điển (Dictionary)**: {"ten": "An", "tuoi": 20}
- **Boolean**: True, False

Ví dụ:

```
x = 10
y = 3.5
ten = "An"
diem = [7, 8, 9]
sv = {"ten": "Bình", "tuoi": 21}
```

## A.3. Cấu trúc điều khiển

- **Câu lệnh rẽ nhánh**

```
diem = 8
if diem >= 5:
    print("Đậu")
else:
    print("Rớt")
```

- **Vòng lặp**

```
for i in range(5):
```

```
    print(i)

n = 3
while n > 0:
    print(n)
    n -= 1
```

### A.4. Hàm trong Python

```
def tinh_tong(a, b):
    return a + b

print(tinh_tong(5, 3))
```

### A.5. Làm quen với thư viện khoa học dữ liệu

- **NumPy**: Tính toán số học trên mảng

```
import numpy as np
a = np.array([1, 2, 3])
print(a.mean())
```

- **Pandas**: Làm việc với dữ liệu dạng bảng

```
import pandas as pd
data = {"Ten": ["An", "Bình"], "Tuoi": [20, 21]}
df = pd.DataFrame(data)
print(df)
```

- **Matplotlib**: Trực quan hóa dữ liệu

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [2, 4, 6, 8]
plt.plot(x, y)
plt.title("Biểu đồ tuyến tính")
plt.show()
```

### A.6. Thực hành nhỏ

1. Viết chương trình tính điểm trung bình của một danh sách điểm.
2. Tạo DataFrame lưu thông tin 5 sinh viên (tên, tuổi, điểm). Xuất ra màn hình.

Vẽ biểu đồ cột thể hiện điểm của các sinh viên.

# PHỤ LỤC B: THƯ VIỆN MẪU

## B.1. NumPy – Thư viện mảng số học

- Khởi tạo mảng:

```
import numpy as np
a = np.array([1, 2, 3, 4])
b = np.arange(0, 10, 2)   # [0, 2, 4, 6, 8]
c = np.linspace(0, 1, 5)  # [0., 0.25, 0.5, 0.75, 1.]
```

- Thao tác cơ bản:

```
print(a.shape)    # Kích thước mảng
print(a.mean())   # Trung bình
print(a.max())    # Giá trị lớn nhất
print(a + 10)     # Cộng 10 cho từng phần tử
```

## B.2. Pandas – Phân tích dữ liệu dạng bảng

- Tạo Series và DataFrame:

```
import pandas as pd
data = {"Ten": ["An", "Bình", "Chi"], "Tuoi": [20, 21, 22],
"Diem": [8.5, 7.0, 9.0]}
df = pd.DataFrame(data)
print(df.head())   # 5 dòng đầu
```

- Thao tác dữ liệu:

```
print(df["Diem"].mean())          # Điểm trung bình
print(df[df["Diem"] >= 8])        # Lọc sinh viên có điểm >= 8
df["XepLoai"] = ["Giỏi" if d >= 8 else "Khá" for d in
df["Diem"]]
```

- Đọc/Ghi file:

```
df.to_csv("sinhvien.csv", index=False)
df2 = pd.read_csv("sinhvien.csv")
```

### B.3. Matplotlib & Seaborn – Trực quan hóa dữ liệu

- Vẽ biểu đồ cơ bản:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [2, 4, 6, 8]
plt.plot(x, y, marker="o")
plt.title("Biểu đồ tuyến tính")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

- Biểu đồ cột và histogram:

```
plt.bar(df["Ten"], df["Diem"])
plt.hist(df["Diem"], bins=5)
```

- Seaborn (nâng cao):

```
import seaborn as sns
sns.boxplot(x=df["Diem"])
sns.scatterplot(x="Tuoi", y="Diem", data=df)
```

### B.4. Scikit-learn – Học máy cơ bản

- Tập dữ liệu mẫu:

```
from sklearn.datasets import load_iris
iris = load_iris()
print(iris.data[:5])   # 5 dòng đầu
```

- Chia dữ liệu huấn luyện và kiểm thử:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42
)
```

- Huấn luyện mô hình cây quyết định:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
print("Độ chính xác:", model.score(X_test, y_test))
```

**B.5. Thực hành nhỏ**

1. Dùng **NumPy** tạo mảng số ngẫu nhiên 10 phần tử, tính giá trị trung bình và độ lệch chuẩn.
2. Dùng **Pandas** đọc một file CSV nhỏ (vd: doanh thu bán hàng), tính tổng doanh thu theo từng mặt hàng.
3. Vẽ biểu đồ cột bằng **Matplotlib/Seaborn** để so sánh doanh thu các mặt hàng.

Dùng **Scikit-learn** huấn luyện mô hình phân loại cơ bản (Decision Tree hoặc KNN) trên dữ liệu Iris.

# PHỤ LỤC C: CÀI ĐẶT MÔI TRƯỜNG

## C.1. Cài đặt Python

1. **Dùng Anaconda (khuyên dùng cho khoa học dữ liệu)**

- Tải Anaconda từ: https://www.anaconda.com/download
- Chọn phiên bản phù hợp với hệ điều hành (Windows, macOS, Linux).
- Cài đặt theo hướng dẫn mặc định.
- Sau khi cài đặt, có thể mở **Anaconda Navigator** hoặc **Anaconda Prompt** để quản lý môi trường.

2. **Dùng Miniconda hoặc cài đặt Python trực tiếp**

- Tải Miniconda: https://docs.conda.io/en/latest/miniconda.html
- Hoặc tải Python từ: https://www.python.org/downloads/
- Cài thêm gói cần thiết bằng **pip** hoặc **conda**.

## C.2. Cài đặt Jupyter Notebook

- Nếu dùng Anaconda: Jupyter Notebook đã có sẵn.
- Nếu dùng Python + pip: cài đặt bằng lệnh

```
pip install notebook
```

- Mở Jupyter Notebook bằng lệnh:

```
jupyter notebook
```

- Trình duyệt sẽ mở giao diện, cho phép tạo file .ipynb để viết và chạy code Python trực tiếp.

## C.3. Cài đặt các thư viện cần thiết

Một số thư viện cơ bản cho môn học:

```
pip install numpy pandas matplotlib seaborn scikit-learn
```

Nếu học thêm deep learning:

```
pip install tensorflow keras torch torchvision
```

## C.4. Cài đặt và sử dụng Visual Studio Code cho Python

1. Cài đặt VS Code

- Tải từ trang chính thức: https://code.visualstudio.com/
- Cài đặt theo hướng dẫn mặc định.

2. Cài đặt Python Extension

- Mở VS Code → vào Extensions (Ctrl+Shift+X) → tìm Python → Install.
- Extension này hỗ trợ chạy Python, gợi ý code (IntelliSense), debug, linting, Jupyter Notebook.

3. Chọn môi trường Python

- Bấm Ctrl+Shift+P → gõ Python: Select Interpreter.
- Chọn môi trường Anaconda/Miniconda hoặc Python đã cài.

4. Tạo và chạy file Python

- Tạo file hello.py với nội dung:

print("Xin chào, Khoa học dữ liệu từ VS Code!")

- Chạy file bằng nút Run Python File (góc trên bên phải) hoặc gõ lệnh trong terminal:

python hello.py

5. Sử dụng Jupyter Notebook trong VS Code

- Cài extension Jupyter từ Marketplace.
- Mở hoặc tạo file .ipynb.
- Chọn kernel Python (môi trường đã cài đặt).
- Có thể viết code, chạy từng ô (cell) giống như trong Jupyter Notebook truyền thống.

## C.5. Quản lý môi trường ảo

Trong thực tế, mỗi dự án khoa học dữ liệu có thể yêu cầu các phiên bản thư viện khác nhau. Để tránh xung đột, nên tạo **môi trường ảo (virtual environment)** cho từng dự án. Có hai cách phổ biến:

### C.5.1. Dùng venv (Python có sẵn)

1. Tạo môi trường ảo:

```
python -m venv env
```

2. Kích hoạt môi trường:

- Windows:

```
env\Scripts\activate
```

- macOS/Linux:

```
source env/bin/activate
```

3. Cài thư viện trong môi trường này (chỉ ảnh hưởng dự án, không ảnh hưởng toàn hệ thống):

```
pip install numpy pandas matplotlib
```

4. Thoát môi trường:

```
deactivate
```

### C.5.2. Dùng conda (Anaconda/Miniconda)

1. Tạo môi trường mới:

```
conda create -n myenv python=3.10
```

2. Kích hoạt môi trường:

```
conda activate myenv
```

3. Cài thư viện:

```
conda install numpy pandas matplotlib
```

4. Xuất cấu hình môi trường (để chia sẻ với người khác):

```
conda env export > environment.yml
```

Người khác có thể tái tạo môi trường bằng:

```
conda env create -f environment.yml
```

Với cách này, mỗi nhóm bài tập hoặc dự án sẽ có môi trường thư viện riêng, tránh lỗi do "trộn" phiên bản