1.1

10 ví dụ nổi bật về AI theo Forbes

Bài viết *"10 Powerful Examples of Artificial Intelligence in Use Today"* trên **Forbes** trình bày nhiều ứng dụng nổi bật của AI trong cuộc sống hiện đại, từ những trợ lý ảo thân thuộc đến các hệ thống nâng cao trải nghiệm người dùng. Dù bài phát hành từ năm 2017, các ví dụ vẫn cho thấy AI đã được áp dụng đa dạng trong nhiều lĩnh vực

Một số ví dụ đáng chú ý:

> **Trợ lý giọng nói như Siri và Alexa**: Đây là điểm tiếp xúc phổ biến nhất giữa người dùng và AI, hỗ trợ thực hiện các lệnh bằng giọng nói.
>
> **Các thuật toán hành vi và gợi ý thông minh**: AI cung cấp từ các đề xuất cá nhân trên nền tảng số đến nội dung phù hợp dựa trên hành vi người dùng.
>
> Các ứng dụng sâu hơn trong phân tích dữ liệu, chăm sóc khách hàng và tự động hóa quy trình.

Những ví dụ này phản ánh một giai đoạn AI bắt đầu trở thành phần không thể thiếu trong trải nghiệm hàng ngày và hoạt động doanh nghiệp.

Siri – Trợ lý ảo thông minh và bảo mật từ Apple

Trang chính thức **Apple Siri** nhấn mạnh khả năng điều khiển thiết bị bằng giọng nói, giúp người dùng thực hiện tác vụ như gọi điện, nhắn tin, tìm đường đi, đặt lời nhắc, điều khiển nhà thông minh… chỉ qua khẩu lệnh như "Hey Siri"

Apple đặc biệt chú trọng đến **quyền riêng tư**: Siri sử dụng công nghệ **xử lý trực tiếp trên thiết bị** ("on-device intelligence"), đảm bảo rằng giọng nói và dữ liệu không rời khỏi thiết bị của người dùng, trừ khi người dùng chủ động chia sẻ

Mới đây, Siri còn kết hợp với **Apple Intelligence** để giúp tương tác tự nhiên hơn, cho phép đặt câu lệnh liên tục mà không cần lặp lại "Hey Siri," và tích hợp thêm với ChatGPT khi cần thông tin chuyên sâu (với sự cho phép của người dùng)

Ứng dụng AI từ Simplilearn – Tổng quan đa ngành

Simplilearn liệt kê nhiều ứng dụng thực tế cho AI trải dài qua các ngành:

Thương mại điện tử (E-commerce): AI sử dụng để gợi ý sản phẩm cá nhân hóa dựa trên hành vi người dùng

Giáo dục: Hệ thống học thích ứng (adaptive learning) điều chỉnh nội dung học cho từng cá nhân.

Xử lý ngôn ngữ tự nhiên (NLP): Hiển thị qua trợ lý ảo, dịch máy, phân tích cảm xúc, đánh máy như Grammarly.

Robot và tự động hóa, **mạng định vị (navigation)**, **tư vấn tài chính**, **chăm sóc sức khỏe**,… đều là những lĩnh vực AI có mặt rõ rệt

Ví dụ minh họa trong chăm sóc sức khỏe: AI dùng dữ liệu sức khỏe như huyết áp, nồng độ glucose để dự đoán bệnh tiểu đường bằng mô hình máy học

AI trong quản trị CNTT – Chuyên gia và suy luận tình huống

Báo cáo từ **IIT Bombay (CSI Communications)** nhấn mạnh vai trò của các kỹ thuật AI truyền thống như **expert systems** (hệ chuyên gia) và **case-based reasoning** (suy luận theo trường hợp).

Các hệ thông minh này tận dụng tri thức chuyên gia lưu trữ để xử lý sự cố trong quản trị CNTT—ví dụ tự động chẩn đoán và khắc phục vấn đề mạng, so sánh tình huống mới với các sự cố đã lưu để đề xuất giải pháp hợp lý

Điều lý thú là, những phương pháp này vẫn giữ tầm quan trọng, kết hợp với AI hiện đại để mang lại hiệu quả cao trong vận hành hệ thống phức tạp.

Khung lý thuyết về hệ thống thông minh từ arXiv

Giấy tờ *"What is an intelligent system?"* trên **arXiv (2020, cập nhật 2022)** đưa ra định nghĩa thực tiễn về hệ thống thông minh:

> Hệ thống **phải hoạt động như một agent**, tức có khả năng **cảm nhận (perceive)** môi trường và **tác động (act)** lên môi trường, đồng thời tương tác với các agents khác (như con người hoặc hệ thống khác).
>
> Phải **hành xử một cách hợp lý (rationally)**:
>
> > Hành động để **tối ưu hóa khả năng thành công** (acting rationally).
> >
> > Hoặc **lý giải được hành vi của mình (thinking rationally)** nếu cần

Ví dụ:

Xe tự lái — hiểu môi trường, xử lý tín hiệu, đưa hành động hợp lý.

Trợ lý ảo như Siri — nhận âm thanh, hiểu ngữ cảnh, đáp lại hợp lý.

Hệ chatbot tư vấn hoặc hệ chuyên gia — lý giải được quyết định đề xuất hay hành động của mình.

Khung này là hướng dẫn lý thuyết rất hữu ích để thiết kế và đánh giá các intelligent systems trong kỹ thuật.

Tổng hợp cấu trúc và ứng dụng thực tiễn

Dưới đây là bảng tổng quát kết hợp lý thuyết và ứng dụng thực tế:

Thành phần khung	Ví dụ thực tế	Giá trị nổi bật
Cảm nhận môi trường	Tesla tự lái nhận diện tín hiệu và phương tiện xung quanh	Xử lý phức tạp, phản hồi nhanh
Tương tác & hành động	Siri gửi tin nhắn, điều khiển thiết bị	Tiện lợi, truy cập dễ dàng
Hành vi hợp lý (rational)	Cogito điều chỉnh giọng nói chăm sóc khách hàng	Tăng hiệu quả tương tác, cảm xúc
Lý giải hành động (thinking)	Hệ chuyên gia khắc phục sự cố CNTT	Tin cậy, minh bạch quyết định
Học tập và thích nghi	Mô hình y tế dự báo bệnh tiểu đường	Cá nhân hóa chăm sóc sức khỏe

Xu hướng phát triển tương lai

Một số hướng phát triển nổi bật:

On-device AI: Như Siri, giúp bảo mật và phản hồi nhanh; có thể mở rộng sang y tế, tài chính – các lĩnh vực nhạy cảm dữ liệu.

Hybrid AI: Kết hợp giữa học máy, tri thức chuyên gia và lý luận để tạo hệ thống vừa tự học, vừa giải thích được.

AI có thể kiểm soát và minh bạch (Explainable AI): Theo Simplilearn, lĩnh vực giải thích hành vi AI (Explainable AI) đang được quan tâm nhằm tăng sự tin tưởng và dễ quản trị

Generative AI: Mở rộng khả năng sáng tạo – như ChatGPT, tạo nội dung, trợ lý viết hỗ trợ doanh nghiệp và cá nhân; thị trường dự đoán đạt giá trị khổng lồ trong tương lai gần

Kết luận

Những ví dụ nổi bật như **Siri**, **xe tự lái**, **Cogito**, hệ chuyên gia trong IT, và nền tảng học máy trong y tế minh chứng AI đang hiện diện sâu rộng trong đời sống và công việc. Trong khi đó, các nghiên cứu lý thuyết như arXiv cung cấp khung xây dựng hệ thống thông minh một cách bền vững, linh hoạt và có khả năng giải thích.

Tương lai của hệ thống thông minh (intelligent systems) nằm ở:

> Sự kết hợp nhuần nhuyễn giữa **nhận thức ngữ cảnh**, **tri thức chuyên gia**, và **học máy thích ứng**.
>
> Phát triển theo hướng **bảo mật, minh bạch, kiểm soát được (explainable, on-device)**.
>
> Áp dụng ở nhiều ngành nhạy cảm như y tế, tài chính, hệ thống điều khiển...

1.2

Khái niệm Hệ thống Thông minh

Định nghĩa theo IGI Global

Theo IGI Global:

> Một hệ thống thông minh là “một máy với một máy tính nhúng kết nối Internet có khả năng thu thập và phân tích dữ liệu và giao tiếp với hệ thống khác”
>
> Hoặc “một hệ thống có thể mô phỏng, tự động hóa một số hành vi thông minh của con người. Ví dụ: hệ thống chuyên gia (expert systems) và hệ thống dựa trên tri thức (knowledge-based systems)”

Khái niệm theo hệ thống mô hình tác tử (Agent-Based)

Một định nghĩa khác từ IGI Global đánh giá hệ thống thông minh như một “hệ thống có tổ hợp các thành phần và tiểu hệ thống phối hợp mục tiêu giống như sinh vật sống”

Tổng quan từ ngữ nghĩa học Wikipedia

Theo Wikipedia, trí tuệ nhân tạo (AI) là khả năng của máy tính trong các nhiệm vụ cần đến trí thông minh con người như **học hỏi, lý luận, giải quyết vấn đề, nhận thức và ra quyết định**

Định nghĩa ấn tượng nhất và lý do

Trong các định nghĩa được trình bày, mình ấn tượng nhất với mô tả từ **IGI Global** về hệ thống thông minh như:

> *"Một hệ thống kết hợp nhiều thành phần hoạt động phối hợp như sinh vật sống, có khả năng thu nhận dữ liệu, phân tích, suy luận và thực thi hành động dựa trên mục tiêu..."*

Tại sao định nghĩa này nổi bật?

Tính hợp thành: Mô tả hệ thống như một thể thống nhất từ nhiều thành phần, tương tự tổ hợp sinh học như cơ thể động vật — thể hiện tính kiến trúc rõ ràng.

Hướng mục tiêu rõ ràng: Không chỉ hành xử ngẫu nhiên mà luôn có mục đích hướng đến kết quả cụ thể.

Phối hợp linh hoạt giữa nhận thức, phân tích và hành động — thể hiện bản chất "thông minh" thực thụ.

Định nghĩa này phản ánh chiều sâu của một hệ thống không chỉ hoạt động, mà hiểu và tồn tại trong môi trường phức hợp.

Các ví dụ phong phú của Hệ thống Thông minh

Dưới đây là các ví dụ minh họa trải dài từ đời sống cá nhân đến các hệ thống công nghiệp – mỗi ví dụ phản ánh ít nhất một khía cạnh của định nghĩa hệ thống thông minh.

Thiết bị hỗ trợ đời sống – Roomba và hệ sinh thái nhà thông minh

Robot hút nhà Roomba có cảm biến và định vị để điều hướng, tránh chướng ngại, và làm sạch hiệu quả

Nhà thông minh (smart home): thiết bị như ổ cắm, điều hòa, hệ thống chiếu sáng tự điều chỉnh dựa theo hành vi và điều kiện môi trường

Nhân diện sinh trắc (Biometrics)

Công nghệ **nhận diện khuôn mặt** (Face ID), **vân tay, mống mắt** là hệ thống sử dụng thị giác máy tính và học máy để xác thực người dùng

Định vị & Gợi ý thông minh

Các **ứng dụng bản đồ, điều hướng** như Google Maps sử dụng AI dự đoán tắc đường và đề xuất lộ trình tối ưu

Hệ thống gợi ý sản phẩm/ nội dung dựa trên hành vi người dùng, ví dụ YouTube, Netflix, Amazon

Trợ lý ảo (Virtual Assistants)

Siri, Alexa, Google Assistant sử dụng NLP, AI để hiểu yêu cầu bằng giọng nói và thực hiện lệnh như đặt lịch, tìm thông tin

Lọc Spam & Bảo mật hệ thống

Hệ thống **lọc email rác (spam filtering)** dùng học máy để phân tích mẫu thư và loại bỏ hiệu quả

Dịch ngôn ngữ tự động

Ứng dụng như **Google Translate, DeepL** chuyển ngữ nhờ thuật toán dịch máy học (statistical hoặc neural)

Hệ thống giao thông thông minh

Hệ thống giao thông thông minh (ITS) như camera nhận diện xe, tín hiệu giao thông tự điều khiển, cảnh báo tai nạn, chỉ đường theo thời tiết

Tổng hợp, Nhận xét và Định hướng tương lai

Tổng hợp các đặc tính thể hiện trong ví dụ

Quan sát môi trường: cảm biến Roomba, camera ITS, dữ liệu GPS.

Phân tích thông tin: học máy trong lọc spam, gợi ý nội dung, dự báo tắc đường.

Ra quyết định hợp lý: Roomba điều hướng, trợ lý ảo thực thi lệnh.

Hành động phù hợp: bật đèn, dọn nhà, đề xuất tuyến đường.

Học hỏi & thích nghi: hệ thống gợi ý càng dùng càng chính xác; Roomba điều chỉnh bản đồ theo nhà.

Những gì làm nên một hệ thống thực sự “thông minh”

Hệ thống không chỉ hoạt động theo kịch bản cố định; nó:

Hiểu môi trường, **tự điều chỉnh** và **học hỏi**.

Có **mục tiêu cụ thể** và khả năng **phối hợp nhiều thành phần** để đạt mục tiêu đó.

Có thể **ra quyết định hợp lý** và **hành động hiệu quả**.

Triển vọng tương lai & Thách thức

Sự hội tụ giữa AI, IoT và hệ thống đa tác nhân (multi-agent systems) sẽ tạo **hệ sinh thái thông minh chân thực** như thành phố thông minh, trợ lý cá nhân luôn sẵn sàng và toàn diện.

Tuy nhiên, cần giải quyết **vấn đề đạo đức, bảo mật, minh bạch (explainable AI)**, và **lỗi ngữ cảnh (common sense reasoning)**—AI thường chưa hiểu được những tình huống phức hợp như con người

Kết luận

Định nghĩa về hệ thống thông minh như một **hệ thống hợp thành với mục tiêu rõ ràng và khả năng thích nghi** khiến mình ấn tượng mạnh vì chiều sâu và tổng hợp được nhiều yếu tố quản lý, kỹ thuật và hành vi.

Các ví dụ từ đời sống — như **Roomba, trợ lý ảo, điều hướng, lọc spam, giao thông thông minh** — minh chứng thực tế cho những đặc trưng thông minh mà định nghĩa đề cập.

Trong tương lai, khi sự thông minh càng lan tỏa vào mọi khía cạnh của đời sống, chúng ta cần hướng đến xây dựng hệ thống **hiểu sâu hơn**, **thích nghi tốt hơn** và **lành mạnh hơn** cho xã hội.

1.3

Ứng dụng trong các lĩnh vực chính

Ứng dụng theo ngành nghề — từ Built In

Trang Built In “86 Artificial Intelligence Examples Shaking Up Business Across Industries” liệt kê rất nhiều ví dụ AI đang làm chuyển biến các ngành công nghiệp, bao gồm:

Trợ lý ảo (virtual assistants): hỗ trợ người dùng, tối ưu hoá dịch vụ khách hàng.

Hệ thống dựa trên generative AI: tạo nội dung, hình ảnh, văn bản đều dùng trí tuệ nhân tạo.

Xe tự hành (autonomous vehicles): ứng dụng AI trong giao thông thông minh.

Hệ thống phát hiện gian lận (fraud detection): bảo mật tài chính, chống rủi ro.

Ngoài ra, các ngành AI ứng dụng rộng rãi hiện nay còn bao gồm: y tế, tài chính, bán lẻ, robot, sản xuất, logistics...

Ứng dụng theo UNR — các ví dụ đa dạng khác

Theo trang của Đại học Nevada, Reno (UNR), hệ thống thông minh (intelligent systems) có thể cảm nhận và phản hồi môi trường, với nhiều ứng dụng thực tiễn như:

Tự động hóa nhà máy (factory automation)

Robot hiện trường và dịch vụ (field and service robotics)

Robot hỗ trợ (assistive robotics)

Ứng dụng quân sự và chăm sóc y tế

Giáo dục, giải trí, kiểm tra bằng thị giác, nhận dạng ký tự (OCR), sinh trắc học (biometrics như nhận diện khuôn mặt, vân tay, mống mắt, lòng bàn tay)

Giám sát bằng hình ảnh, vận tải thông minh (intelligent transportation)

Bảng tổng hợp ứng dụng theo ngành

Ngành / Lĩnh vực	Ứng dụng cụ thể
Công nghiệp & sản xuất	Tự động hóa, kiểm tra chất lượng, bảo trì thông minh
Giao thông & vận tải	Xe tự hành, hệ thống giao thông thông minh, giảm ùn tắc
Chăm sóc sức khỏe & y tế	Hỗ trợ chẩn đoán hình ảnh, phân tích bệnh nhân, phẫu thuật robot
Tài chính & ngân hàng	Phát hiện gian lận, quản lý rủi ro, tối ưu dịch vụ khách hàng
Bán lẻ & thương mại điện tử	Cá nhân hóa trải nghiệm shopping, hệ thống đề xuất sản phẩm
Giáo dục & đào tạo	Hệ thống dạy kèm thông minh (ITS), bài tập tự học cá nhân hóa
Giải trí & truyền thông	Nội dung tạo tự động, chatbot tương tác, đề xuất phim/sách
An ninh & nhận dạng sinh trắc học	Nhận dạng khuôn mặt, giám sát thông minh, kiểm tra an ninh
Robotics (đa dạng)	Robot hỗ trợ, robot dịch vụ, robot trong công nghiệp

Các kỹ thuật AI thường dùng

Kỹ thuật từ nguồn học thuật — ELEKS (Industry 4.0)

Các **kỹ thuật AI phổ biến** được áp dụng trong sản xuất và công nghiệp tự động hóa gồm:

Computer Vision: để kiểm tra chất lượng sản phẩm, theo dõi quy trình, phát hiện lỗi theo thời gian thực.

Natural Language Processing (NLP): dùng để tự động báo cáo, dịch chuông báo lỗi, trợ lý ảo hỗ trợ vận hành.

Time Series Analysis: dự báo sự cố, giá cả, sản lượng... dựa trên dữ liệu thời gian.

Signal Processing: xử lý âm thanh, giọng nói, nhận diện, tổng hợp tiếng nói.

Recommender Systems: đề xuất sản phẩm hoặc tài nguyên dựa trên hành vi người dùng.

Anomaly Detection: phát hiện bất thường, lỗi hệ thống qua mô hình học không giám sát.

General Optimization: tối ưu hóa tuyến đường, phân bổ nhân lực, tăng doanh thu.

Expert Systems: hệ thống chuyên gia dự đoán lỗi, đưa ra hướng khắc phục.

Các kỹ thuật logic và kiến thức từ Wikipedia

Expert System: hệ thống dựa trên kiến thức (thường là các luật if–then) và inference engine để suy luận như chuyên gia.

Inference Engine: thành phần ứng dụng quy tắc logic lên knowledge base để suy ra thông tin mới; cũng có nghĩa mô tả inference trong mô hình neural network, xử lý real-time.

Học máy và học sâu (Machine Learning & Deep Learning)

Machine Learning: hệ thống học từ dữ liệu để tự động xây dựng mô hình phân tích và giải quyết vấn đề.

Deep Learning: dựa trên mạng neural để xử lý, học các biểu diễn dữ liệu sâu hơn và thường mạnh mẽ hơn ML truyền thống.

Tổng hợp – Áp dụng trong các hệ thống thông minh

Mối liên hệ giữa kỹ thuật và ứng dụng

Computer Vision giúp robots, hệ thống giám sát thị giác, nhận dạng và kiểm tra sản xuất.

NLP phục vụ chatbot, hệ thống tương tác trong giáo dục, báo cáo trong sản xuất.

Time Series / Signal Processing ứng dụng trong bảo trì thông minh, y tế theo dõi bệnh nhân, giám sát hệ thống.

Expert Systems / Inference Engines: dùng trong pháp lý, y tế, hỗ trợ ra quyết định tự động.

ML / DL: rộng khắp từ nhận diện khuôn mặt, tự lái, chẩn đoán bệnh, phân tích thị trường...

Ví dụ minh hoạ

Manufacturing (Industry 4.0): sử dụng predictive maintenance, kiểm soát chất lượng bằng computer vision, tối ưu hóa chuỗi cung ứng bằng phân tích thời gian và học máy.

Giao thông: xe tự hành nhờ học sâu và thị giác máy tính, hệ thống đèn giao thông thông minh tối ưu lưu lượng.

Giáo dục: Hệ thống dạy kèm thông minh (Intelligent Tutoring Systems — ITS) cung cấp hướng dẫn cá nhân hóa, tương tác tự động.

Y tế: chẩn đoán bằng hình ảnh, hỗ trợ lập kế hoạch điều trị bằng deep learning và NLP.

Kết luận & triển vọng

Hệ thống thông minh (intelligent systems) kết hợp các kỹ thuật như computer vision, NLP, học máy, hệ thống chuyên gia... để phục vụ các mục tiêu cụ thể trong nhiều lĩnh vực.

Giao thông, y tế, giáo dục, sản xuất, tài chính, an ninh đều là những lĩnh vực đang thay đổi rõ rệt nhờ AI.

Trong tương lai, sự hợp nhất giữa nhiều kỹ thuật (như DL + NLP + Vision) sẽ tạo nên hệ thống AI mạnh mẽ, linh hoạt và “thông minh” hơn — đáng để theo dõi.

1.4

Phân loại theo năng lực (Capabilities-Based Classification)

Reactive Machines (Máy phản ứng thuần túy)

Đặc điểm: Không có bộ nhớ; chỉ phản ứng theo tình huống hiện tại, không ghi nhớ hay học hỏi từ quá khứ.

Ví dụ tiêu biểu: IBM Deep Blue – máy cờ vua đã đánh bại Kasparov bằng cách xử lý tức thì thế cờ mà không có khái niệm về kinh nghiệm chơi trước đó.

Đây là dạng AI đơn giản và có khả năng xử lý tốt trong các môi trường tách biệt, không cần thích nghi hay phản hồi theo thời gian.

Limited Memory (Bộ nhớ giới hạn)

Đặc điểm: Có thể sử dụng dữ liệu quá khứ một cách giới hạn để đưa ra quyết định trong tương lai gần.

Ví dụ tiêu biểu: Xe tự lái — hệ thống nhận diện môi trường xung quanh liên tục và điều chỉnh hành vi dựa vào dữ liệu vừa thu thập.

Đây là dạng AI phổ biến hiện nay vì có thể cải thiện hiệu suất dựa trên ngữ cảnh ngắn hạn.

Theory of Mind (Lý thuyết tâm trí)

Đặc điểm: Có khả năng nhận biết và hiểu được cảm xúc, động cơ, niềm tin của con người hoặc thực thể khác.

Hiện trạng: Hiện vẫn chưa có hệ thống AI nào đạt đến cấp độ này — chỉ tồn tại trong nghiên cứu lý thuyết.

Self-Aware (Tự nhận thức)

Đặc điểm: Không chỉ hiểu được ý thức của người khác mà còn có ý thức về bản thân; sở hữu cảm xúc, tự suy nghĩ và ra hành động có mục đích.

Hiện trạng: Đây là mức độ AI lý tưởng trong tương lai; hiện tại vẫn chưa tồn tại ngoài giả tưởng khoa học viễn tưởng.

Phân loại theo mục tiêu phát triển (Ambition-Based Classification)

Artificial Narrow Intelligence (ANI) / Weak AI

Định nghĩa: AI thực hiện tốt một hoặc một vài nhiệm vụ cụ thể; không có hiểu biết tổng quát hay nhận thức.

Ví dụ:

Trợ lý ảo như Siri, Alexa

Hệ thống nhận diện hình ảnh

Công nghệ đề xuất nội dung (Netflix, Amazon)

Đây là dạng AI phổ biến nhất hiện nay.

Artificial General Intelligence (AGI) / Strong AI

Định nghĩa: AI sở hữu khả năng tư duy, học hỏi và giải quyết vấn đề trên nhiều lĩnh vực như con người.

Hiện trạng: Chưa có ví dụ thực tế nào đạt được AGI — vẫn là mục tiêu nghiên cứu dài hạn

Artificial Superintelligence (ASI)

Định nghĩa: AI vượt trội hơn con người về mọi mặt – từ sáng tạo, trí tuệ, đến khả năng giải quyết vấn đề.

Hiện trạng: Chỉ là giả thuyết, thường xuất hiện trong văn hóa và viễn tưởng khoa học.

So sánh & Các dạng kết hợp mở rộng

So sánh giữa hai phân loại

Phân loại theo năng lực	Phân loại theo mục tiêu	Tương quan
Reactive Machines	ANI (Weak AI)	Đơn nhiệm vụ, không học hỏi
Limited Memory	ANI	Có học hỏi ngắn hạn, ứng dụng thực tiễn
Theory of Mind	Tiềm ẩn hướng tới AGI	Hiểu ngữ cảnh & cảm xúc, không có thực thể
Self-Aware	Tiềm ẩn hướng tới AGI/ASI	Có ý thức, tự suy nghĩ — chỉ giả tưởng

Một số dạng mở rộng hoặc bổ sung

Hybrid Intelligence: Kết hợp sức mạnh của AI và con người để xử lý tác vụ phức tạp hơn hiệu quả hơn mỗi bên riêng lẻ.

Neuro-symbolic AI: Kết hợp mạng neural (deep learning) với lập luận dựa trên biểu tượng (symbolic reasoning) để xử lý cả học máy và logic trừu tượng.

BriSe AI (Brain-inspired & Self-based AI): Khái niệm AI lấy cảm hứng từ cấu trúc nhận thức và ý thức của con người, gồm nhiều "tầng" như tự nhận thức, tương tác xã hội, lý trí... để tiến đến AGI.

Kết luận và Ứng dụng Thực tế

Tại sao cần hiểu các phân loại này?

Giúp **định hướng nghiên cứu và phát triển dự án**: biết rõ mình đang hướng đến dạng AI nào để chọn kỹ thuật, dữ liệu và kiến trúc phù hợp.

Hiểu rõ giới hạn: **hiện nay chủ yếu là ANI**, còn AGI và ASI vẫn là mục tiêu tương lai, cần cân nhắc các vấn đề đạo đức và kỹ thuật đúng đắn.

Ứng dụng thực tế hiện nay – chủ yếu là ANI và Limited Memory

Tự động hóa sản xuất: robot kiểm tra linh kiện, điều chỉnh theo dữ liệu sản xuất gần nhất.

Trợ lý ảo & Chatbot: phản hồi người dùng dựa trên ngữ cảnh tương tác ngắn hạn (Limited Memory).

Xe tự lái: nhận diện môi trường và phản ứng nhanh (Limited Memory); AI phản ứng đơn giản như cảnh báo tai nạn (Reactive).

Hệ thống chăm sóc khách hàng: gợi ý sản phẩm, phân tích giọng nói để tư vấn (Limited Memory ANI).

Xu hướng tương lai

Hợp nhất các dạng như **Neuro-symbolic AI** và **Hybrid Intelligence** để tiến gần hơn đến AGI — hệ thống có cả khả năng học từ dữ liệu và suy luận trừu tượng.

Phát triển các mô hình có **tương tác xã hội và cảm xúc (Theory of Mind)** — giúp ứng dụng AI trở nên tự nhiên và hữu ích hơn trong chăm sóc sức khỏe, giáo dục, hỗ trợ tinh thần.

Xây dựng khung đạo đức và kiểm soát để đối phó với những mối nguy từ AGI/ASI tiềm năng — đặc biệt là trong việc ra quyết định tự chủ và xử lý các tình huống phức tạp.

Kết luận

Phân loại theo năng lực (Reactive, Limited Memory, Theory of Mind, Self-Aware) giúp hiểu cấp độ phức tạp trong cách AI tương tác với thế giới.

Phân loại theo mục tiêu (ANI, AGI, ASI) xác định tham vọng và tiềm năng phát triển của hệ thống.

Các hướng mở như Hybrid Intelligence, Neuro-symbolic AI… đang tiến tới khả năng hiểu sâu hơn và hành xử thông minh hơn.

Hiện tại, ANI – đặc biệt dạng Limited Memory – là phổ biến nhất trong ứng dụng thực tiễn. Và tương lai đang chờ đợi những bước đột phá vào AGI, như mô hình BriSe AI, bằng cách kết hợp nhiều kỹ thuật và lý luận triết học về ý thức.

1.5

Giới thiệu

Trong kỷ nguyên cách mạng công nghiệp 4.0, **hệ thống thông minh (Intelligent System)** ngày càng đóng vai trò trung tâm trong nhiều lĩnh vực của đời sống và sản xuất. Khác với các hệ thống truyền thống, hệ thống thông minh có khả năng **cảm nhận môi trường, suy luận, ra quyết định và tương tác** với con người hoặc với các hệ thống khác. Chúng thường thể hiện những đặc tính gắn liền với trí tuệ con người như học tập, thích nghi, giải quyết vấn đề và giao tiếp bằng ngôn ngữ tự nhiên.

Trong bài báo *“What is an Intelligent System?”* (Molina, 2020), tác giả đưa ra **Hình 7** như một minh họa điển hình cho các loại ứng dụng của hệ thống thông minh. Nó trình bày rõ hai nhóm lớn:

Hệ thống thông minh tự chủ (Intelligent autonomous system)

Hệ thống thông minh tư vấn (Intelligent advisor system)

Từ hai nhánh này, nhiều loại ứng dụng cụ thể được liệt kê như robot tự hành, xe tự lái, trợ lý ảo, chatbot, hệ thống gợi ý, và các hệ thống chuyên gia. Báo cáo này sẽ lần lượt phân tích các ứng dụng trong từng nhánh, nhằm làm rõ vai trò thực tiễn của hệ thống thông minh.

Hệ thống thông minh tự chủ (Intelligent autonomous system)

Tự chủ trong môi trường thực (Autonomous in a real environment)

Đây là nhóm hệ thống được triển khai trực tiếp trong thế giới vật lý, nơi chúng phải đối mặt với nhiều yếu tố không chắc chắn và thay đổi liên tục.

Robot tự hành (Autonomous robot)
Ví dụ: **Curiosity** – robot thăm dò không gian của NASA. Curiosity có khả năng tự lập kế hoạch, phân tích địa hình và đưa ra quyết định di chuyển mà không cần con người điều khiển trực tiếp. Ứng dụng của loại robot này rất rộng, từ khám phá không gian, dò tìm môi trường nguy hiểm cho đến nghiên cứu dưới biển sâu.

Robot cộng tác (Collaborative robot)
Ví dụ: **Baxter**, một robot công nghiệp có thể làm việc an toàn bên cạnh con người. Các robot cộng tác hỗ trợ sản xuất linh hoạt, có thể học các thao tác mới và thích nghi với môi trường sản xuất hiện đại.

Xe tự lái (Self-driving vehicle)
Đây là một trong những ứng dụng nổi bật nhất hiện nay. Xe tự lái sử dụng hệ thống cảm biến, thị giác máy tính và trí tuệ nhân tạo để điều hướng an toàn trên đường. Tương lai, xe tự lái hứa hẹn sẽ giảm tai nạn giao thông, tối ưu hóa giao thông đô thị và thay đổi ngành vận tải.

Tự động hóa trong gia đình (Home automation)
Ví dụ: hệ thống **điều khiển năng lượng trong nhà**. Các thiết bị thông minh như cảm biến nhiệt độ, hệ thống chiếu sáng tự động hay điều hòa thông minh giúp tối ưu hóa việc sử dụng năng lượng và nâng cao tiện nghi cuộc sống.

Tự động hóa công nghiệp thông minh (Intelligent industrial automation)
Ví dụ: hệ thống **chẩn đoán và kiểm soát sửa chữa trong dây chuyền sản xuất**. Các nhà máy hiện đại ứng dụng AI để phát hiện lỗi sớm, giảm thiểu thời gian ngừng máy, tăng năng suất và đảm bảo chất lượng sản phẩm.

Tự chủ trong môi trường ảo (Autonomous in a virtual environment)

Khác với môi trường thực, các hệ thống này vận hành chủ yếu trong không gian số hoặc phần mềm.

Phần mềm robot (Software robot)
Ví dụ: **Robotic Process Automation (RPA)** – robot phần mềm có khả

năng tự động hóa các tác vụ lặp lại trong văn phòng như xử lý hóa đơn, nhập dữ liệu, quản lý email. Điều này giúp giảm sai sót và tiết kiệm chi phí vận hành.

Nhân vật ảo (Virtual character)
Ví dụ: nhân vật dùng để **đào tạo hoặc giải trí**. Trong giáo dục, nhân vật ảo có thể đóng vai trò người huấn luyện mô phỏng. Trong giải trí, nhân vật ảo mang lại trải nghiệm nhập vai trong game hoặc môi trường thực tế ảo.

Hệ thống gia sư thông minh (Intelligent tutoring system)
Ví dụ: **Practical Algebra Tutor (PAT)**. Đây là công cụ giáo dục có thể tương tác với học sinh, đánh giá mức độ hiểu bài và đưa ra gợi ý phù hợp. Điều này minh chứng cho tiềm năng to lớn của AI trong giáo dục cá nhân hóa.

Trợ lý ảo cá nhân (Personal assistant)
Ví dụ: **Siri, Alexa**. Đây là các hệ thống có khả năng nhận diện giọng nói, hiểu ngôn ngữ tự nhiên và thực hiện lệnh. Chúng giúp người dùng điều khiển thiết bị thông minh, tìm kiếm thông tin và quản lý lịch trình.

Chatbot giao dịch (Transactional chatbot)
Ví dụ: chatbot trong thương mại điện tử. Những chatbot này hỗ trợ khách hàng chọn sản phẩm, trả lời câu hỏi thường gặp, thậm chí xử lý giao dịch. Điều này cải thiện trải nghiệm khách hàng và giảm tải cho bộ phận hỗ trợ.

Hệ thống thông minh tư vấn (Intelligent advisor system)

Khác với hệ thống tự chủ, hệ thống tư vấn tập trung vào việc **hỗ trợ con người ra quyết định** thông qua việc phân tích dữ liệu và cung cấp lời khuyên.

Hệ thống hỏi đáp (Question answering system)
Ví dụ: **Watson của IBM** hoặc **ChatGPT**. Các hệ thống này có khả năng hiểu ngôn ngữ tự nhiên, tìm kiếm và tổng hợp thông tin để đưa ra câu trả lời chính xác. Chúng có ứng dụng rộng rãi trong dịch vụ khách hàng, y tế, giáo dục và nghiên cứu.

Hệ thống hỗ trợ quản lý (Management support system)
Ví dụ: hệ thống **quản lý đội xe buýt**. Thông qua việc thu thập dữ liệu GPS và các thông số vận hành, hệ thống đưa ra khuyến nghị tối ưu hóa lộ trình, giảm chi phí và nâng cao chất lượng dịch vụ công cộng.

Hệ thống gợi ý (Recommender system)
Ví dụ: gợi ý sách hoặc tin tức. Các hệ thống này dựa trên dữ liệu hành vi của người dùng để đề xuất nội dung phù hợp, từ đó cá nhân hóa trải nghiệm và tăng mức độ hài lòng. Chúng là nền tảng cốt lõi của các dịch vụ thương mại điện tử, mạng xã hội và nền tảng giải trí trực tuyến.

Hệ thống chuyên gia thế hệ đầu (First-generation expert system)
Ví dụ: **Mycin** – hệ thống chẩn đoán bệnh truyền nhiễm. Đây là một trong những minh chứng đầu tiên cho việc máy tính có thể hỗ trợ ra quyết định trong lĩnh vực y tế. Mặc dù còn hạn chế, Mycin đặt nền móng cho sự phát triển của các hệ thống tư vấn hiện đại.

Kết luận

Hình 7 trong bài báo đã chỉ ra một bức tranh đa dạng về ứng dụng của hệ thống thông minh, được chia thành hai nhóm lớn: **tự chủ** và **tư vấn**.

Hệ thống tự chủ cho thấy sức mạnh của AI trong việc vận hành độc lập, từ robot thám hiểm sao Hỏa, xe tự lái, cho đến trợ lý ảo trong môi trường số.

Hệ thống tư vấn khẳng định vai trò của AI trong việc hỗ trợ con người phân tích dữ liệu, gợi ý và ra quyết định, điển hình là Watson, ChatGPT hay hệ thống gợi ý sản phẩm.

Qua các ví dụ cụ thể, có thể thấy hệ thống thông minh đã và đang thâm nhập vào hầu hết mọi lĩnh vực: khoa học, công nghiệp, giáo dục, y tế, thương mại và dịch vụ công. Chúng không chỉ cải thiện hiệu quả và năng suất, mà còn góp phần nâng cao chất lượng cuộc sống.

Trong tương lai, sự kết hợp giữa hai nhóm này – **tự chủ** và **tư vấn** – sẽ tạo ra những hệ thống ngày càng hoàn thiện, có khả năng vừa tự hành động vừa hỗ trợ con người một cách thông minh. Điề u này mở ra triển vọng về một xã hội nơi con người và trí tuệ nhân tạo cùng hợp tác, bổ sung cho nhau trong mọi hoạt động.

1.6

Numpy

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

Which Language is NumPy written in?

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

NumPy Illustrated Examples

Creating Arrays

```
import numpy as np

# 1D array
arr1 = np.array([1, 2, 3, 4, 5])

# 2D array
arr2 = np.array([[1, 2, 3], [4, 5, 6]])

print("1D array:", arr1)
print("2D array:\n", arr2)
```

Output:

```
1D array: [1 2 3 4 5]

2D array:

 [[1 2 3]

  [4 5 6]]
```

NumPy arrays are like super-efficient lists that can be multi-dimensional.

Array Properties

print("Shape:", arr2.shape) # rows, columns

print("Size:", arr2.size) # total elements

print("Data type:", arr2.dtype)

Output:

Shape: (2, 3)

Size: 6

Data type: int64

shape tells you the structure, .size is the total number of elements.

Vectorized Operations (fast math!)

a = np.array([1, 2, 3, 4])

b = np.array([10, 20, 30, 40])

print("a + b:", a + b) # element-wise addition

print("a * b:", a * b) # element-wise multiplication

print("a ** 2:", a ** 2) # square each element

Output:

a + b: [11 22 33 44]

a * b: [10 40 90 160]

a ** 2: [1 4 9 16]

Instead of looping, NumPy applies operations to the entire array at once (much faster).

Indexing and Slicing

```
arr = np.array([10, 20, 30, 40, 50])


print(arr[0])    # first element

print(arr[-1])   # last element

print(arr[1:4])  # slice from index 1 to 3
```

Output:

```
10

50

[20 30 40]
```

Works like Python lists, but with more power in multi-dimensional arrays.

Matrix Operations

```
A = np.array([[1, 2], [3, 4]])

B = np.array([[5, 6], [7, 8]])


print("Matrix multiplication:\n", np.dot(A, B))

print("Transpose of A:\n", A.T)
```

Output:

```
Matrix multiplication:
 [[19 22]
  [43 50]]
Transpose of A:
 [[1 3]
  [2 4]]
```

NumPy is widely used for **linear algebra**, like multiplying matrices and transposing them.

Random Numbers

```
rand_arr = np.random.rand(3, 3)   # 3x3 random floats between 0 and 1

print(rand_arr)
```

Example Output:

[[0.34 0.78 0.21]

[0.55 0.12 0.91]

[0.67 0.45 0.89]]

Useful in simulations, machine learning, or testing.

Statistics

data = np.array([5, 10, 15, 20, 25])

print("Mean:", np.mean(data))

print("Standard deviation:", np.std(data))

print("Sum:", np.sum(data))

Output:

Mean: 15.0

Standard deviation: 7.0710678118654755

Sum: 75

NumPy makes math on datasets easy.

Pandas

What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

What Can Pandas Do?

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

Pandas Illustrated Examples

Creating a DataFrame

```
import pandas as pd


data = {
   "Name": ["Alice", "Bob", "Charlie", "David"],
   "Age": [25, 30, 35, 40],
   "City": ["New York", "London", "Paris", "Tokyo"]
}


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

Output:

```
     Name  Age      City
0   Alice   25  New York
1     Bob   30    London
```

2 Charlie 35 Paris

3 David 40 Tokyo

A **DataFrame** is like an Excel table: rows + columns.

Accessing Data

```
print(df["Name"])      # Get a column
print(df.iloc[1])      # Get row by index
print(df.loc[2, "City"])  # Get specific cell
```

Output:

0 Alice

1 Bob

2 Charlie

3 David

Name: Name, dtype: object

Name Bob

Age 30

City London

Name: 1, dtype: object

Paris

iloc = index position, loc = label-based.

Filtering Data

```
print(df[df["Age"] > 30])  # filter rows where Age > 30
```

Output:

  Name Age City

2 Charlie   35 Paris

3   David   40 Tokyo

Filtering works like applying conditions in Excel.

Adding New Column

df["Age in 5 Years"] = df["Age"] + 5

print(df)

Output:

      Name Age   City Age in 5 Years

0   Alice   25 New York   30

1   Bob   30   London   35

2 Charlie   35   Paris   40

3   David   40   Tokyo   45

You can calculate new values column-wise.

Grouping and Aggregation

data = {

    "Department": ["IT", "IT", "HR", "HR", "Finance"],

    "Salary": [5000, 6000, 4000, 4500, 7000]

}

df2 = pd.DataFrame(data)

print(df2.groupby("Department")["Salary"].mean())

Output:

Department

Finance   7000.0

HR   4250.0

IT 5500.0

Name: Salary, dtype: float64

Great for summaries like **average salary by department**.

Handling Missing Data

```
df3 = pd.DataFrame({
    "Name": ["Eva", "Frank", "Grace"],
    "Age": [28, None, 33]
})


print(df3.fillna(0))   # Replace missing values with 0
print(df3.dropna())    # Drop rows with missing values
```

Output:

```
    Name   Age
0    Eva  28.0
1  Frank   0.0
2  Grace  33.0


    Name   Age
0    Eva  28.0
2  Grace  33.0
```

Pandas makes missing value handling very easy.

Reading CSV Files

```
# Example: reading from a CSV file
df_csv = pd.read_csv("data.csv")
print(df_csv.head())   # show first 5 rows
```

Pandas can read/write many formats: **CSV, Excel, SQL, JSON, etc.**

Visualization with Pandas

import matplotlib.pyplot as plt

df["Age"].plot(kind="bar")

plt.show()

This will show a **bar chart** of Ages. Pandas integrates directly with Matplotlib.

Mathplotlib

What is Matplotlib?

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Matplotlib Illustrated Examples

Basic Line Plot

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

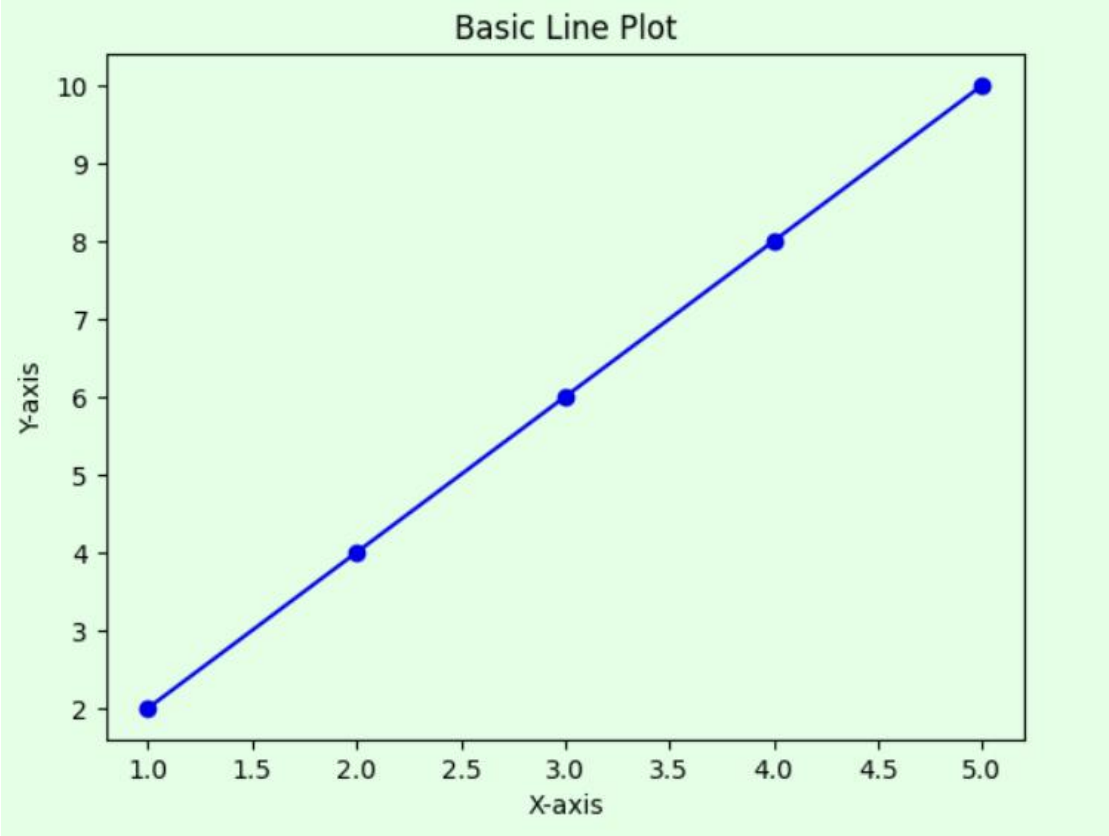
y = [2, 4, 6, 8, 10]

plt.plot(x, y, marker="o", linestyle="-", color="b")

plt.title("Basic Line Plot")

```
plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.show()
```

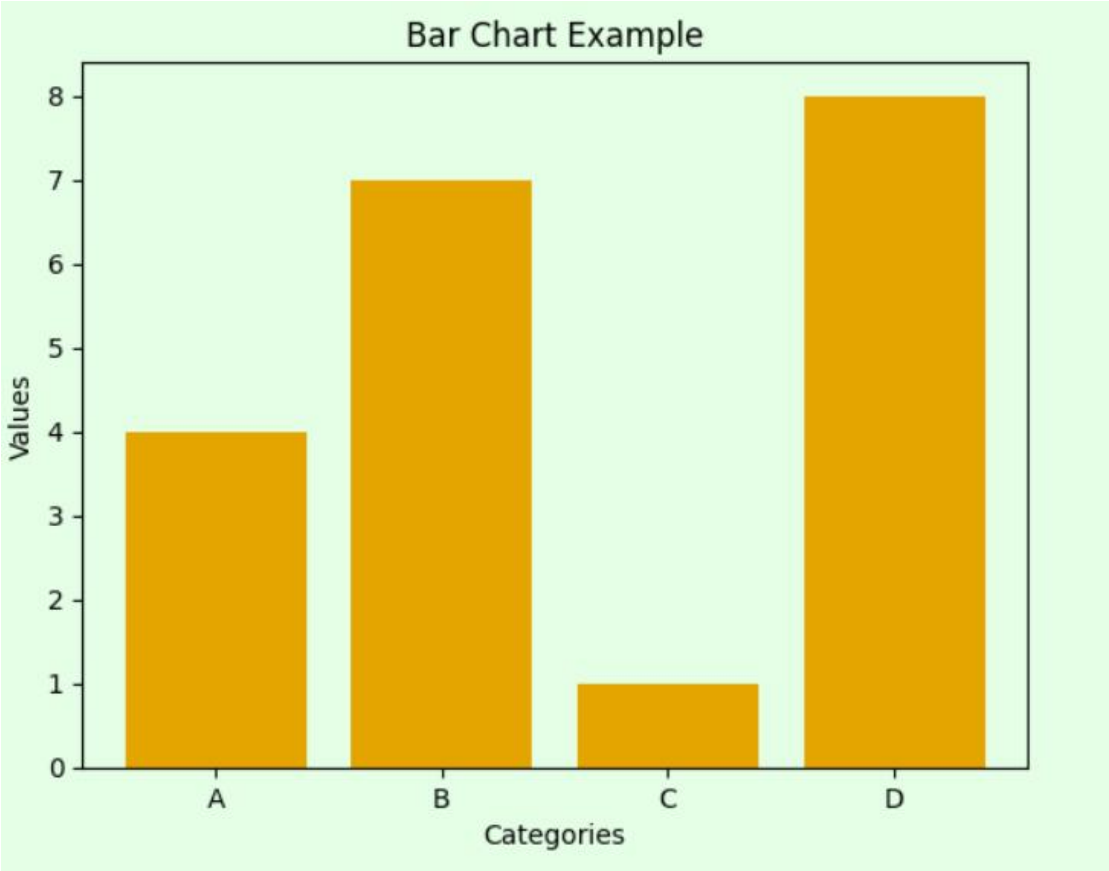
Output



Bar Chart

```
categories = ["A", "B", "C", "D"]

values = [4, 7, 1, 8]


plt.bar(categories, values, color="orange")

plt.title("Bar Chart Example")

plt.xlabel("Categories")

plt.ylabel("Values")

plt.show()
```

Output:



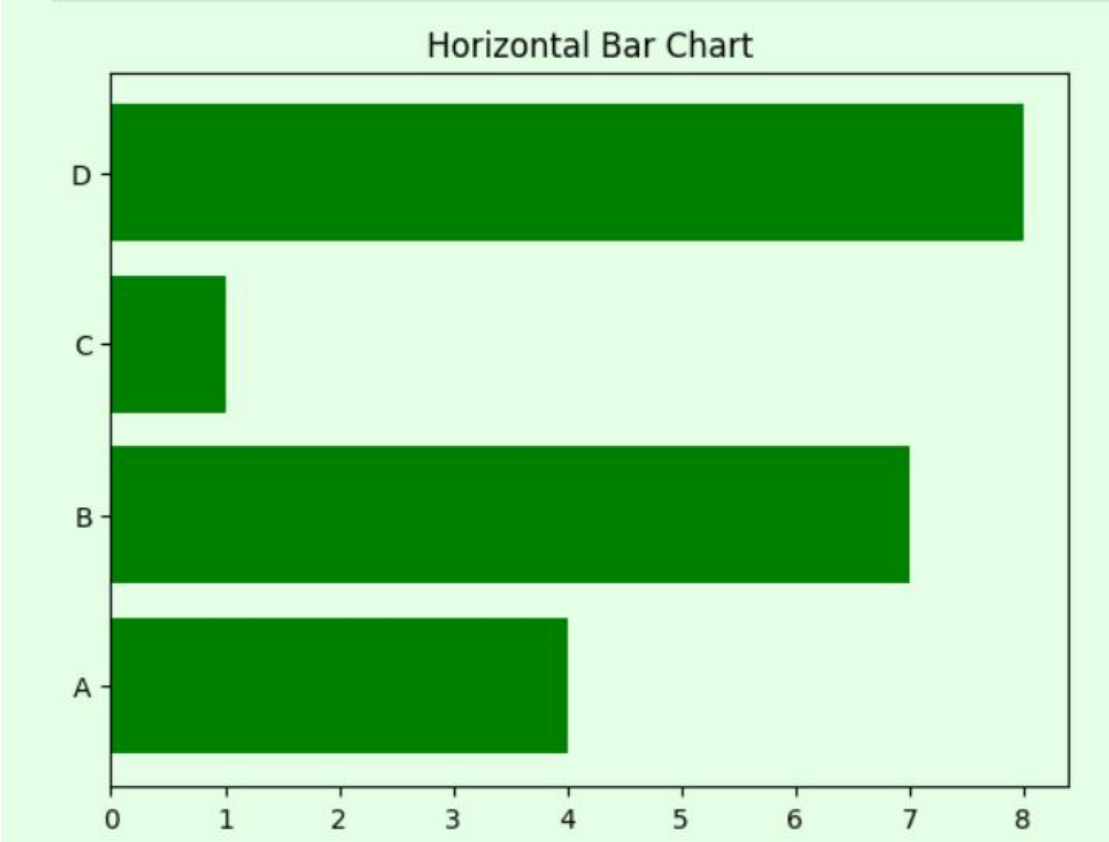
Horizontal Bar Chart

plt.barh(categories, values, color="green")

plt.title("Horizontal Bar Chart")

plt.show()

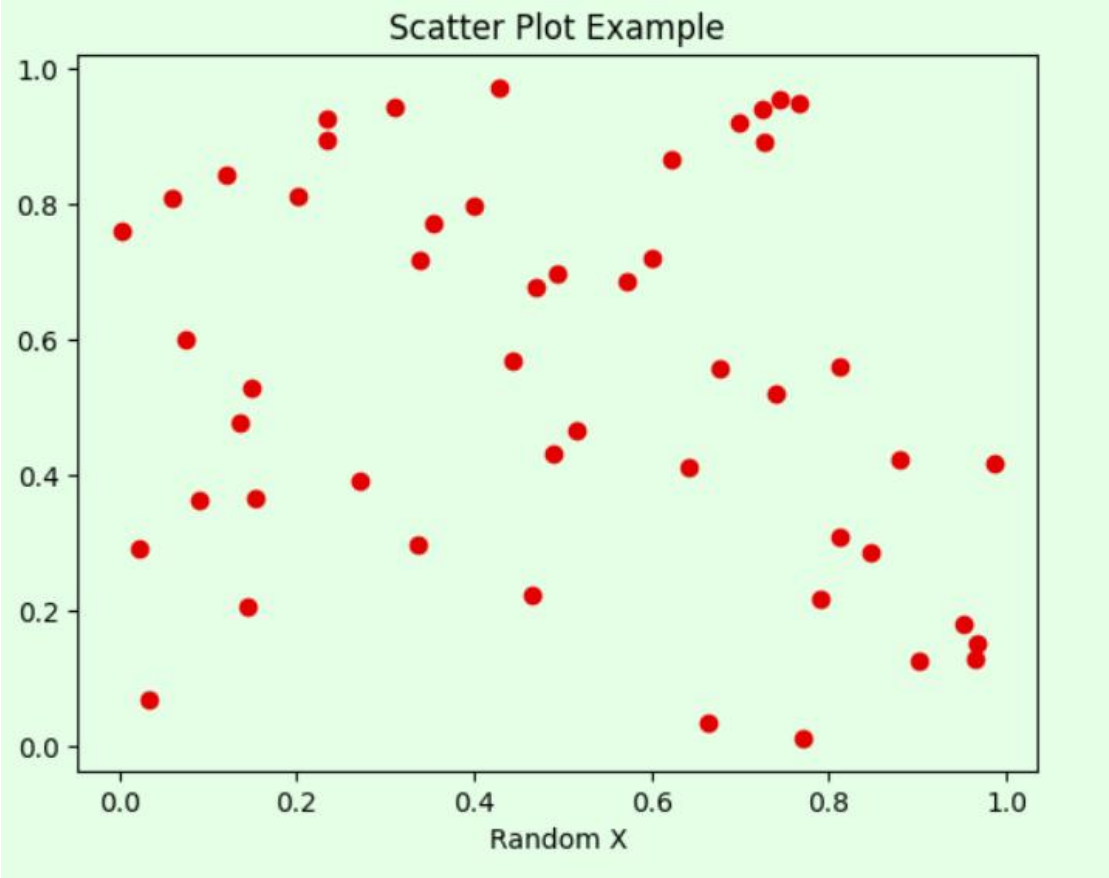
Output:



Scatter Plot

```
import numpy as np

x = np.random.rand(50)
y = np.random.rand(50)

plt.scatter(x, y, color="red")
plt.title("Scatter Plot Example")
plt.xlabel("Random X")
plt.ylabel("Random Y")
plt.show()
```

Output:

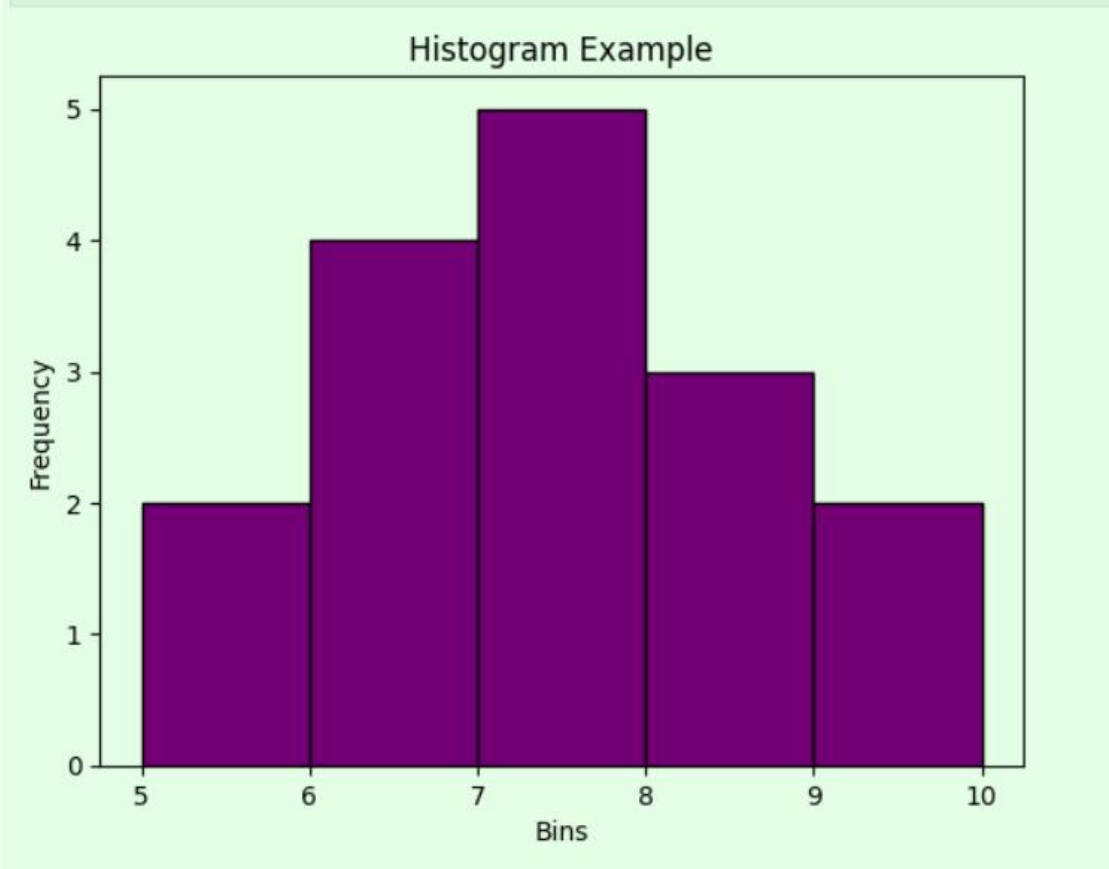


Histogram

data = [7, 8, 5, 6, 6, 7, 9, 10, 6, 5, 7, 8, 7, 8, 6, 7]

plt.hist(data, bins=5, color="purple", edgecolor="black")
plt.title("Histogram Example")
plt.xlabel("Bins")
plt.ylabel("Frequency")
plt.show()

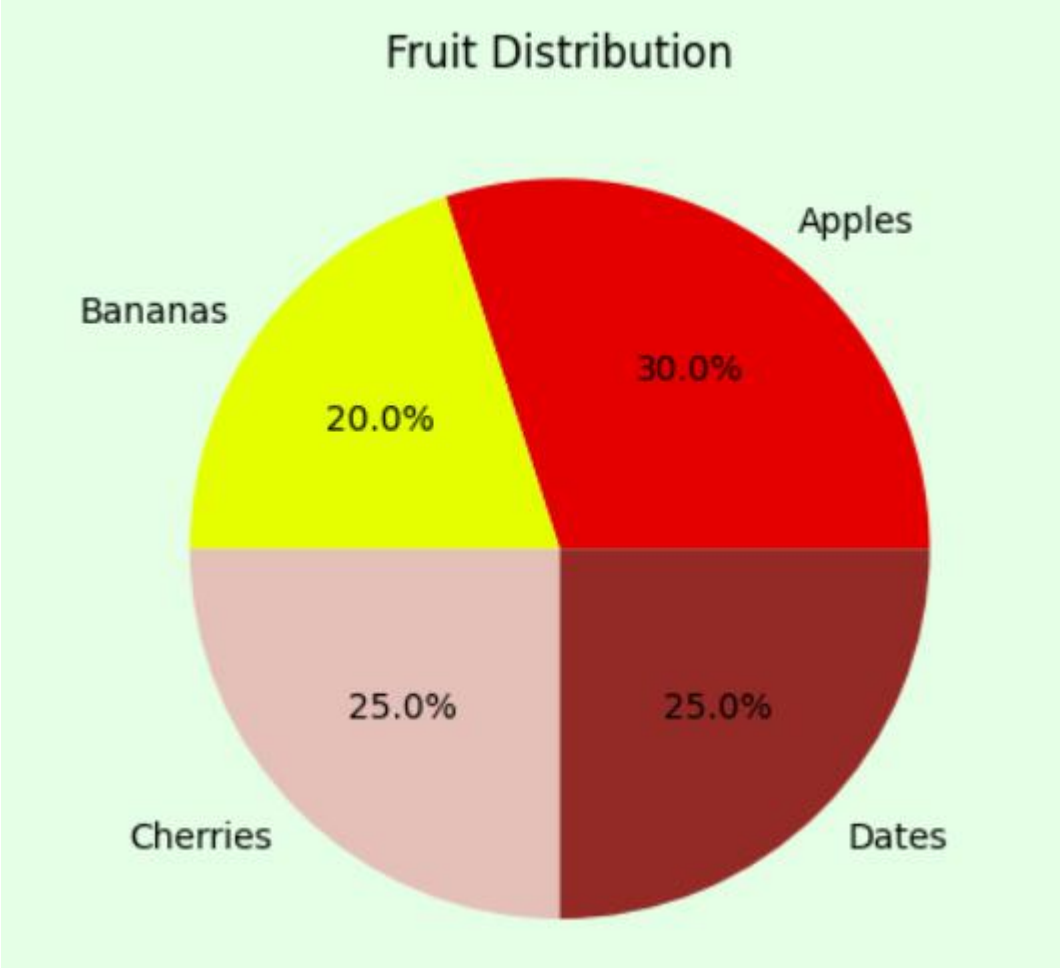
Output:



Pie Chart

```
sizes = [30, 20, 25, 25]
labels = ["Apples", "Bananas", "Cherries", "Dates"]
colors = ["red", "yellow", "pink", "brown"]

plt.pie(sizes, labels=labels, colors=colors, autopct="%1.1f%%")
plt.title("Fruit Distribution")
plt.show()
```

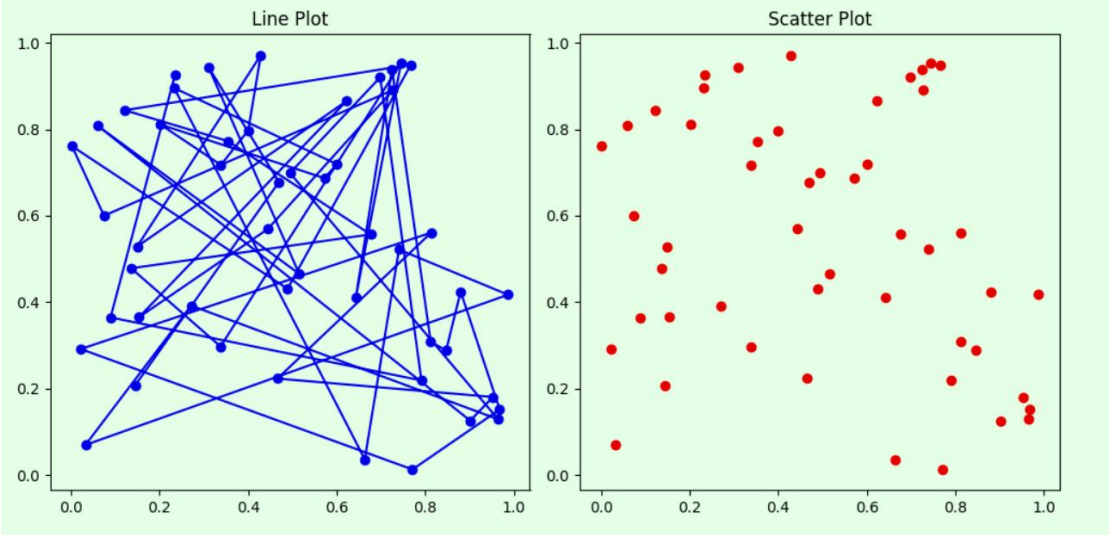
Output:



Multiple Plots in One Figure

```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)  # 1 row, 2 columns, first subplot
plt.plot(x, y, "bo-")
plt.title("Line Plot")

plt.subplot(1,2,2)  # second subplot
plt.scatter(x, y, color="red")
plt.title("Scatter Plot")

plt.tight_layout()
plt.show()
```

Output:



Styling with Grid

x = [1,2,3,4,5]
y = [1,4,9,16,25]

plt.plot(x, y, "g^--") # green, triangle markers, dashed line
plt.grid(True)
plt.title("Styled Plot with Grid")
plt.show()

Output:

Scikit-learn (sklearn)

Purpose

Scikit-learn provides simple and efficient tools for:

Data preprocessing (cleaning, transforming, scaling data)

Model selection (choosing the best ML algorithm)

Training ML models (classification, regression, clustering, etc.)

Model evaluation (accuracy, precision, confusion matrix, etc.)

Deployment-ready ML pipelines

It is built on **NumPy, SciPy, and Matplotlib**, making it powerful yet easy to use.

Key Features

Supervised learning

Classification (predict categories, e.g., spam/not spam)

Regression (predict continuous values, e.g., house prices)

Unsupervised learning

Clustering (e.g., K-Means, group similar customers)

Dimensionality reduction (e.g., PCA)

Model evaluation & validation

Train/test split, cross-validation, accuracy, confusion matrix

Preprocessing utilities

Feature scaling (StandardScaler, MinMaxScaler)

Encoding categorical data (OneHotEncoder, LabelEncoder)

Pipelines

Combine preprocessing + model training steps

Illustrated Examples

Classification (Iris Dataset)

Predict flower type from features (famous Iris dataset).

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score


# Load dataset

iris = load_iris()

X, y = iris.data, iris.target


# Split dataset

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


# Train model

model = KNeighborsClassifier(n_neighbors=3)

model.fit(X_train, y_train)


# Predict

y_pred = model.predict(X_test)


print("Accuracy:", accuracy_score(y_test, y_pred))
```

Output: Accuracy score (e.g., ~0.97).
Purpose: Classify flowers (Setosa, Versicolor, Virginica).

Regression (Predict House Prices)

```
import numpy as np

from sklearn.linear_model import LinearRegression


# Example dataset (house size vs price)

X = np.array([[50], [60], [80], [100], [120]])  # size in m²

y = np.array([150, 200, 300, 400, 500])         # price in $1000


# Train linear regression

model = LinearRegression()

model.fit(X, y)


# Predict price for 90m² house

prediction = model.predict([[90]])

print("Predicted price for 90m² house:", prediction)
```

Output: e.g., Predicted price for 90m² house: [350]
Purpose: Regression to predict continuous values.

Clustering (K-Means)

```
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import make_blobs


# Generate synthetic data
```

```
X, _ = make_blobs(n_samples=200, centers=3, cluster_std=0.7,
random_state=42)



# KMeans clustering

kmeans = KMeans(n_clusters=3, random_state=42)

y_kmeans = kmeans.fit_predict(X)



# Plot clusters

plt.scatter(X[:,0], X[:,1], c=y_kmeans, cmap="viridis")

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=200,
c="red", marker="X")

plt.title("KMeans Clustering Example")

plt.show()
```

Output: Scatter plot with **3 clusters** and red X for cluster centers.
Purpose: Group unlabeled data into clusters.

Dimensionality Reduction (PCA)

```
from sklearn.decomposition import PCA
# Reduce Iris dataset from 4D → 2D for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:,0], X_pca[:,1], c=y, cmap="viridis")
plt.title("PCA Dimensionality Reduction")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```

Output: A 2D scatter plot showing flower classes separated by PCA components.
Purpose: Visualize high-dimensional data.

1.7

```python
# 1.7
# Nguyễn Việt Quang
import numpy as np
# Underweight: BMI < 18.5
# Overweight: BMI > 25
# Normal: 18.5 <= BMI <= 25
names = np.array(['Ann', 'Joe', 'Mark'])
heights = np.array([1.5, 1.78, 1.6])
weights = np.array([65, 46, 59])
bmi = weights/(heights**2)
print(bmi)
print("Overweight: {}".format(names[bmi > 25]))
print("Underweight: {}".format(names[bmi < 18.5]))
print("Normal: {}".format(names[(bmi >= 18.5) & (bmi <= 25)]))
```

```
[28.88888889 14.51836889 23.046875  ]
Overweight: ['Ann']
Underweight: ['Joe']
Normal: ['Mark']
```

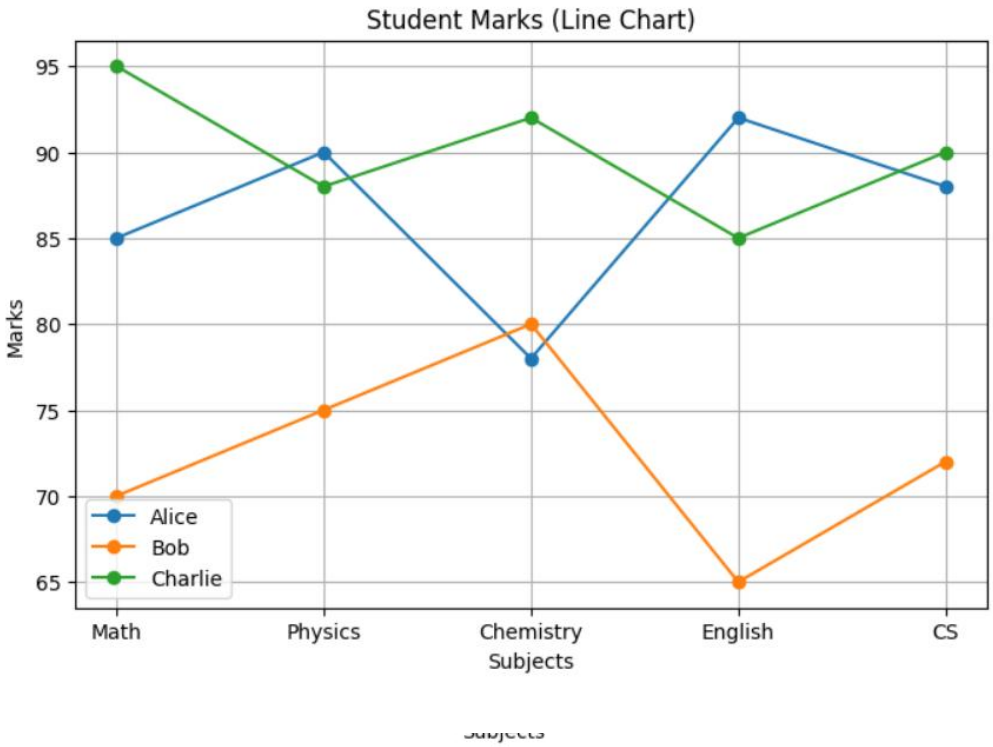
1.8

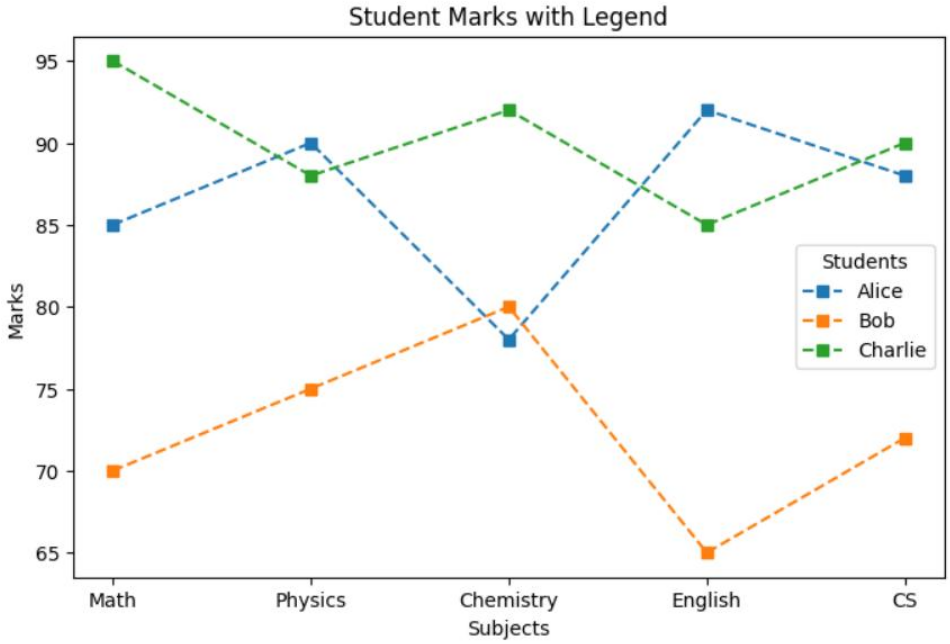
```python
# 1.8
# Nguyễn Việt Quang
import matplotlib.pyplot as plt
import numpy as np

# ===== Example Data (replace with your team's data) =====
students = ["Alice", "Bob", "Charlie"]
subjects = ["Math", "Physics", "Chemistry", "English", "CS"]

marks = {
    "Alice": [85, 90, 78, 92, 88],
    "Bob": [70, 75, 80, 65, 72],
    "Charlie": [95, 88, 92, 85, 90]
}

# Convert subjects to numeric x-axis
x = np.arange(len(subjects))
```

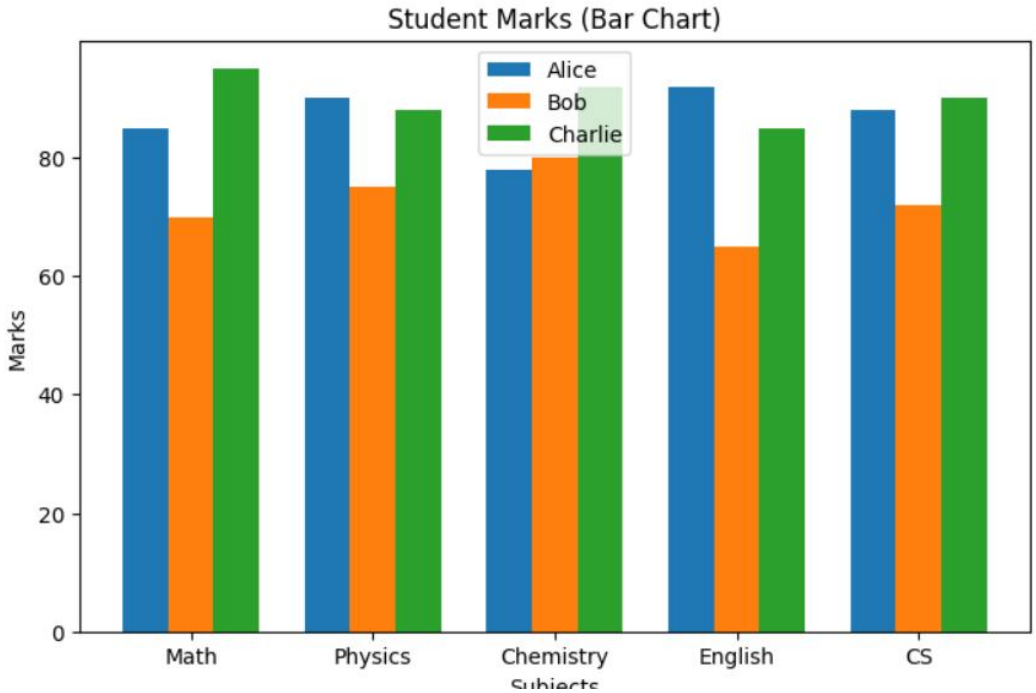
```
# ===== 1. Plotting Multiple Lines in the Same Chart =====
plt.figure(figsize=(8, 5))
for student, scores in marks.items():
    plt.plot(x, scores, marker="o", label=student)  # multiple lines
plt.xticks(x, subjects)
plt.title("Student Marks (Line Chart)")
plt.xlabel("Subjects")
plt.ylabel("Marks")
plt.legend()
plt.grid(True)
plt.show()
```

```
# ===== 2. Adding a Legend (already shown above, but here's a focused version) =====
plt.figure(figsize=(8, 5))
for student, scores in marks.items():
    plt.plot(x, scores, marker="s", linestyle="--", label=student)
plt.xticks(x, subjects)
plt.title("Student Marks with Legend")
plt.xlabel("Subjects")
plt.ylabel("Marks")
plt.legend(title="Students")  # Legend with title
plt.show()
```

```
# ===== 3. Plotting Bar Charts =====
width = 0.25  # width of bars
plt.figure(figsize=(8, 5))
for i, (student, scores) in enumerate(marks.items()):
    plt.bar(x + i*width, scores, width, label=student)
plt.title("Student Marks (Bar Chart)")
plt.xticks(x + width, subjects)
plt.xlabel("Subjects")
plt.ylabel("Marks")
plt.legend()
plt.show()
```

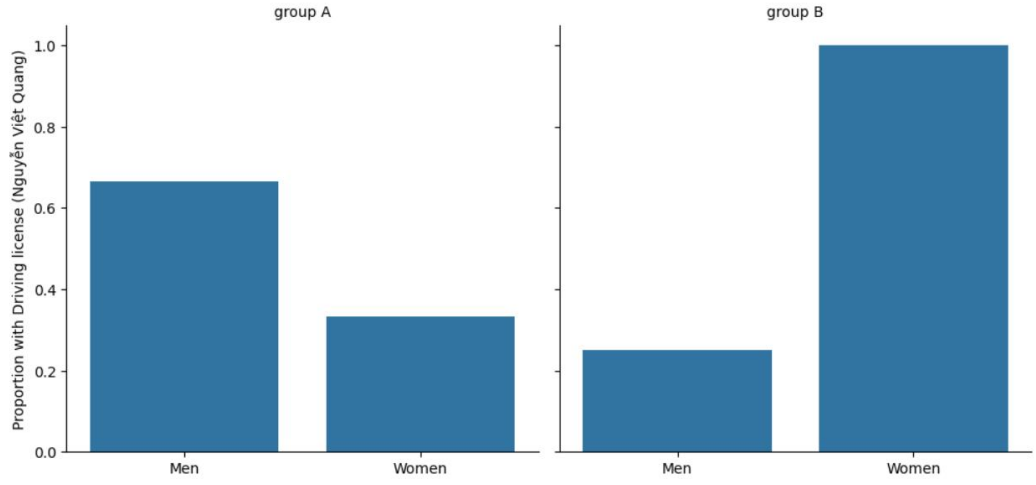

Student Marks (Line Chart)
Marks
95
90
85
80
75
70
65
Alice
Bob
Charlie
Math
Physics
Chemistry
English
CS
Subjects


Student Marks with Legend
Marks
95
90
85
80
75
70
65
Students
Alice
Bob
Charlie
Math
Physics
Chemistry
English
CS
Subjects



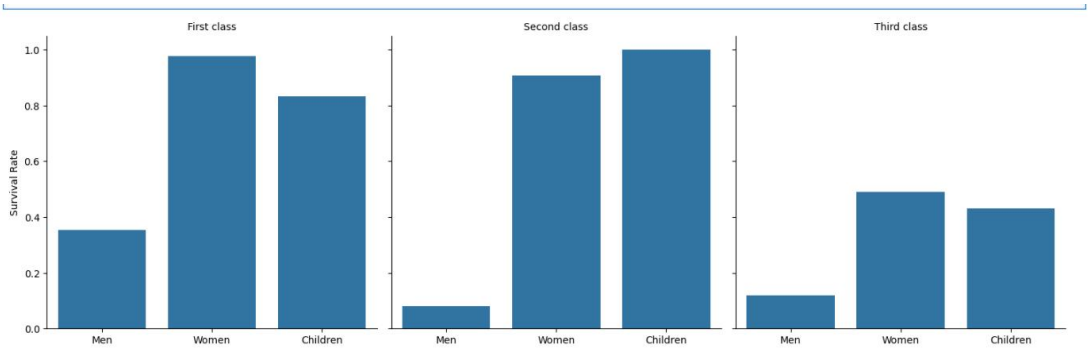
1.9

```
]: # 1.9
#Nguyễn Việt Quang
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
#---load data---
data = pd.read_csv('drivinglicense.csv')
#---plot a factorplot---
g = sns.catplot(x="gender", y="license", col="group", data=data, kind="bar", errorbar=None, aspect=1.0)

#---set the labels---
g.set_axis_labels("", "Proportion with Driving license (Nguyễn Việt Quang)")
g.set_xticklabels(["Men", "Women"])
g.set_titles("{col_var} {col_name}")
#---show plot---
plt.show()
```



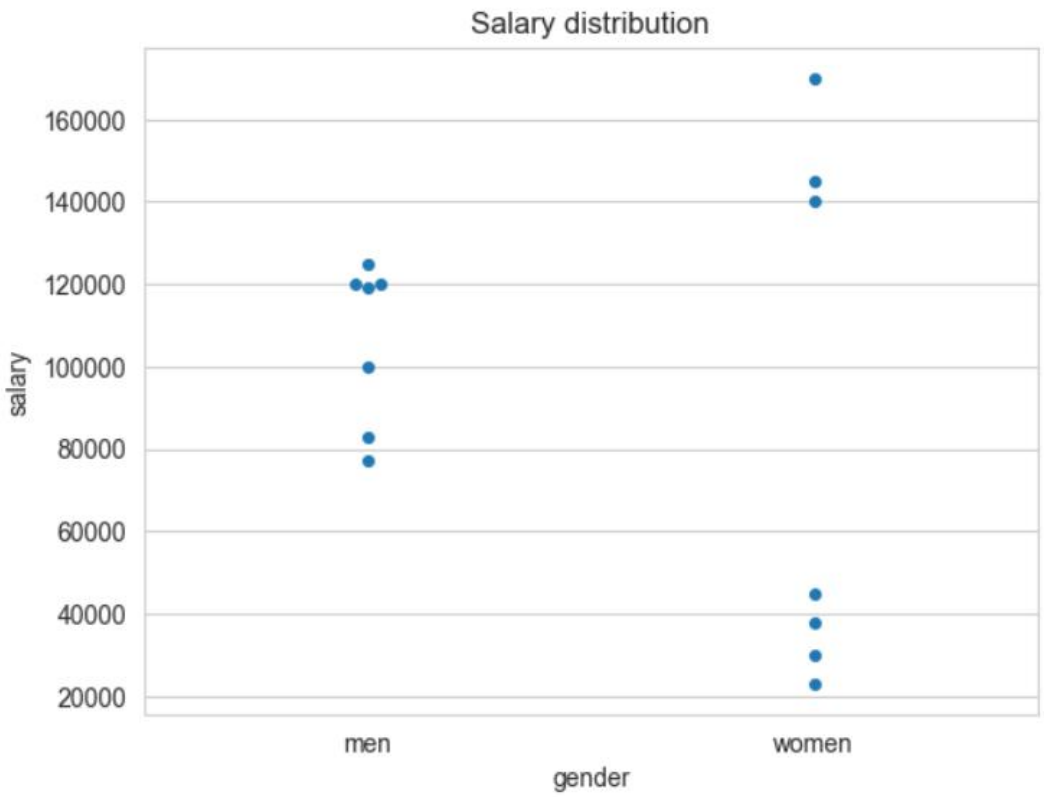
1.10

```
]: #1.10
   # Nguyễn Việt Quang
   import matplotlib.pyplot as plt
   import seaborn as sns
   titanic = sns.load_dataset("titanic")
   g = sns.catplot(x="who", y="survived", col="class",
    data=titanic, kind="bar", errorbar=None, aspect=1)
   g.set_axis_labels("", "Survival Rate")
   g.set_xticklabels(["Men", "Women", "Children"])
   g.set_titles("{col_name} {col_var}")
   #---show plot---
   plt.show()
```



1.11

```python
# 1.11
# Nguyễn Việt Quang
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
sns.set_style("whitegrid")
#---load data---
data = pd.read_csv('salary.csv')
#---plot the swarm plot---
sns.swarmplot(x="gender", y="salary", data=data)
ax = plt.gca()
ax.set_title("Salary distribution")
#---show plot---
plt.show()
```



1.12

```python
#1.12
# Nguyễn Việt Quang
import numpy as np
from sklearn.linear_model import LinearRegression

# Data
X = np.array([50, 60, 65, 70, 75, 80, 85]).reshape(-1, 1)  # diện tích (m2)
y = np.array([2.5, 3, 3.5, 3.8, 4, 4.5, 5])               # giá nhà (tỷ)

# Train linear regression model
model = LinearRegression()
model.fit(X, y)
print(X)
# Predict
X_test = np.array([55, 68, 76, 90]).reshape(-1, 1)
y_pred = model.predict(X_test)

# Show results
for area, price in zip(X_test.flatten(), y_pred):
    print(f"Diện tích {area} m² -> Dự đoán giá: {price:.2f} tỷ")
```

```
[[50]
 [60]
 [65]
 [70]
 [75]
 [80]
 [85]]
Diện tích 55 m² -> Dự đoán giá: 2.75 tỷ
Diện tích 68 m² -> Dự đoán giá: 3.67 tỷ
Diện tích 76 m² -> Dự đoán giá: 4.23 tỷ
Diện tích 90 m² -> Dự đoán giá: 5.21 tỷ
```

1.13

```python
# 1.13
# Nguyễn Việt Quang
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# represents the heights of a group of people in meters
heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]
# represents the weights of a group of people in kgs
weights = [[60], [65], [72.3], [75], [80]]

# Create and fit the model
model = LinearRegression()
model.fit(X=heights, y=weights)
# make prediction
weight = model.predict([[1.75]])[0][0]
print(round(weight,2))
```

```
76.04
```

1.14

```python
# 1.14
# Nguyễn Việt Quang
from sklearn.datasets import fetch_openml
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split


boston = fetch_openml(name="boston", version=1, as_frame=True)
df = boston.frame
print(df.head())
df.info()
print(df.isnull().sum())
corr = df.corr()
print(corr)
#---get the top 3 features that has the highest correlation---
print(df.corr().abs().nlargest(3, 'MEDV').index)
#---print the top 3 correlation values---
print(df.corr().abs().nlargest(3, 'MEDV').values[:,13])
```

```
x = pd.DataFrame(np.c_[df['LSTAT'], df['RM']], columns = ['LSTAT','RM'])
Y = df['MEDV']

# We will split the dataset into 70 percent for training and 30 percent for testing:
x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size = 0.3,
 random_state=5)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train, Y_train)

price_pred = model.predict(x_test)
print('R-Squared: %.4f' % model.score(x_test,Y_test))
```

```
      CRIM    ZN  INDUS CHAS    NOX     RM   AGE     DIS RAD    TAX  PTRATIO  \
0  0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900   1  296.0     15.3
1  0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242.0     17.8
2  0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242.0     17.8
3  0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222.0     18.7
4  0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222.0     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    category
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    category
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
```

```
 13  MEDV     506 non-null    float64
dtypes: category(2), float64(12)
memory usage: 49.0 KB
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
             CRIM        ZN     INDUS      CHAS       NOX        RM       AGE  \
CRIM     1.000000 -0.200469  0.406583 -0.055892  0.420972 -0.219247  0.352734
ZN      -0.200469  1.000000 -0.533828 -0.042697 -0.516604  0.311991 -0.569537
INDUS    0.406583 -0.533828  1.000000  0.062938  0.763651 -0.391676  0.644779
CHAS    -0.055892 -0.042697  0.062938  1.000000  0.091203  0.091251  0.086518
NOX      0.420972 -0.516604  0.763651  0.091203  1.000000 -0.302188  0.731470
RM      -0.219247  0.311991 -0.391676  0.091251 -0.302188  1.000000 -0.240265
AGE      0.352734 -0.569537  0.644779  0.086518  0.731470 -0.240265  1.000000
DIS     -0.379670  0.664408 -0.708027 -0.099176 -0.769230  0.205246 -0.747881
RAD      0.625505 -0.311948  0.595129 -0.007368  0.611441 -0.209847  0.456022
TAX      0.582764 -0.314563  0.720760 -0.035587  0.668023 -0.292048  0.506456
PTRATIO  0.289946 -0.391679  0.383248 -0.121515  0.188933 -0.355501  0.261515
B       -0.385064  0.175520 -0.356977  0.048788 -0.380051  0.128069 -0.273534

B       -0.385064  0.175520 -0.356977  0.048788 -0.380051  0.128069 -0.273534
LSTAT    0.455621 -0.412995  0.603800 -0.053929  0.590879 -0.613808  0.602339
MEDV    -0.388305  0.360445 -0.483725  0.175260 -0.427321  0.695360 -0.376955

               DIS       RAD       TAX   PTRATIO         B     LSTAT      MEDV
CRIM    -0.379670  0.625505  0.582764  0.289946 -0.385064  0.455621 -0.388305
ZN       0.664408 -0.311948 -0.314563 -0.391679  0.175520 -0.412995  0.360445
INDUS   -0.708027  0.595129  0.720760  0.383248 -0.356977  0.603800 -0.483725
CHAS    -0.099176 -0.007368 -0.035587 -0.121515  0.048788 -0.053929  0.175260
NOX     -0.769230  0.611441  0.668023  0.188933 -0.380051  0.590879 -0.427321
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -0.376955
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -0.381626
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.543993 -0.468536
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.374044 -0.507787
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.366087  0.333461
LSTAT   -0.496996  0.488676  0.543993  0.374044 -0.366087  1.000000 -0.737663
MEDV     0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.737663  1.000000
Index(['MEDV', 'LSTAT', 'RM'], dtype='object')
[1.         0.73766273 0.69535995]
R-Squared: 0.6162
```