Bài 1.1:

I. Giới thiệu

Trí tuệ Nhân tạo (Artificial Intelligence - AI) đã từng là một khái niệm chỉ tồn tại trong tưởng tượng khoa học viễn tưởng, nhưng ngày nay, nó đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta. AI là lĩnh vực nghiên cứu và phát triển các máy móc hoặc chương trình máy tính có khả năng thực hiện các nhiệm vụ thông minh mà trước đây chỉ có con người mới có thể thực hiện. Bài viết này sẽ giới thiệu về AI, lịch sử phát triển, các loại AI, ứng dụng rộng rãi và những thách thức hiện đang đặt ra.

II. Lịch sử phát triển

Khái niệm về AI xuất hiện từ thập kỷ 1950, khi các nhà khoa học và nhà toán học đầu tiên bắt đầu nghiên cứu về khả năng tạo ra các máy tính có khả năng "suy nghĩ" như con người. Trong những năm đầu, sự phát triển chậm chạp và hạn chế về công nghệ đã khiến cho AI không thể thực sự phổ biến.

Tuy nhiên, những nỗ lực không ngừng nghỉ của cộng đồng nghiên cứu đã dẫn đến những tiến bộ đáng kể. Đặc biệt, trong những năm gần đây, các thuật toán học máy và mạng nơ-ron sâu đã giúp AI đạt được những thành tựu đột phá, từ việc nhận diện hình ảnh, ngôn ngữ tự nhiên cho đến tự động lái xe.

III. Các loại AI

AI có thể được chia thành hai loại chính: Trí tuệ Nhân tạo hẹp (Narrow AI) và Trí tuệ Nhân tạo mạnh (General AI).

Trí tuệ Nhân tạo hẹp: Loại này là các hệ thống AI được thiết kế để thực hiện một nhiệm vụ cụ thể một cách hiệu quả, như chatbot, hệ thống dự đoán thời tiết, hoặc xe tự lái. Chúng chỉ hoạt động trong phạm vi rất hẹp và không thể thực hiện các nhiệm vụ ngoài lĩnh vực đã được lập trình.

Trí tuệ Nhân tạo mạnh: Đây là mục tiêu cuối cùng của nghiên cứu về AI, là một hệ thống có khả năng thực hiện mọi nhiệm vụ mà con người có thể thực hiện. Tuy nhiên, đây vẫn là một mục tiêu xa vời và đầy thách thức.

IV. Ứng dụng rộng rãi

AI đã và đang được áp dụng rộng rãi trong nhiều lĩnh vực cuộc sống:

Y tế: AI giúp phân tích hình ảnh y khoa, chẩn đoán bệnh và dự đoán tình trạng sức khỏe.

Công nghiệp: Các robot và hệ thống tự động giúp tăng năng suất và hiệu quả sản xuất.

Giao thông: Công nghệ tự động lái xe dựa trên AI đang dần trở nên thực tế.

Tài chính: AI được sử dụng để dự đoán thị trường tài chính, quản lý rủi ro và giao dịch tự động.

Ngôn ngữ tự nhiên: Chatbot và trợ lý ảo giúp tương tác và hỗ trợ người dùng.

Thương mại điện tử: Nhiều ứng dụng của AI trong thương mại điện tử được nêu ra, bao gồm gợi ý mua sắm cá nhân, trợ lý dựa trên AI và ngăn chặn gian lận. Những ứng dụng này cải thiện trải nghiệm của khách hàng, cải thiện các gợi ý và giảm thiểu vấn đề như gian lận thẻ tín dụng và đánh giá giả mạo.

Giáo dục: Nội dung giải thích cách AI từ từ bước vào lĩnh vực giáo dục. Nó thảo luận về việc tự động hóa các nhiệm vụ hành chính cho giáo viên, tạo nội dung thông minh, sử dụng trợ lý giọng nói cho việc học và trải nghiệm học tập cá nhân. AI được mô tả như một công cụ để nâng cao quy trình giáo dục.

V. Thách thức và Triển vọng

Mặc dù AI mang lại nhiều tiện ích, nhưng cũng đặt ra nhiều thách thức:

An ninh và quyền riêng tư: Sự gia tăng của AI đặt ra câu hỏi về quyền riêng tư và nguy cơ lạm dụng thông tin.

Thất nghiệp: Sự tự động hóa có thể dẫn đến việc mất việc làm cho một số ngành công nghiệp.

Trách nhiệm và đạo đức: AI có thể đưa ra quyết định không đạo đức hoặc không rõ nguồn gốc.

Tuy nhiên, triển vọng của AI vẫn rất sáng sủa. Sự phát triển của nó có thể mang lại những giải pháp sáng tạo cho các vấn đề toàn cầu như biến đổi khí hậu, y tế và năng suất lao động.

VI. Kết luận

Trí tuệ Nhân tạo không chỉ là một lĩnh vực nghiên cứu mà đã trở thành một phần không thể thiếu của cuộc sống hiện đại. Với những tiến bộ đáng kể trong các thuật toán và công nghệ, AI đang mang lại nhiều tiện ích và thay đổi cách chúng ta sống và làm việc. Tuy nhiên, việc sử dụng AI cần được thực hiện một cách cân nhắc và đảm bảo tính bền vững, đạo đức và an toàn.

Bài 1.2:

Trong các định nghĩa về trí tuệ nhân tạo, định nghĩa để lại ấn tượng nhất là:

"Thuyết và phát triển của các hệ thống máy tính có khả năng thực hiện các nhiệm vụ thường đòi hỏi trí tuệ của con người, như nhận thức hình ảnh, nhận dạng giọng nói, ra quyết định và dịch thuật giữa các ngôn ngữ."

Về cơ bản, trí tuệ nhân tạo là phương pháp mà máy tính có khả năng hoạt động dựa trên dữ liệu thông qua phân tích thống kê, cho phép nó hiểu, phân tích và học từ dữ liệu thông qua các thuật toán được thiết kế cụ thể. Đây là một quá trình tự động. Các máy có trí tuệ nhân tạo có thể ghi nhớ các mẫu hành vi và thích nghi với các phản hồi của họ để tuân thủ các hành vi đó hoặc khuyến khích thay đổi chúng. Đây là một định nghĩa ngắn gọn và có thể nói nhiều hơn về trí tuệ nhân tạo.

Các công nghệ quan trọng nhất tạo nên trí tuệ nhân tạo là học máy (Machine Learning - ML), học sâu (Deep Learning) và xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP).

Học máy là quá trình mà máy móc học cách phản hồi tốt hơn dựa trên các tập dữ liệu lớn có cấu trúc và phản hồi liên tục từ con người và thuật toán.

Học sâu thường được coi là một loại học máy tiên tiến hơn vì nó học thông qua biểu diễn, nhưng dữ liệu không cần phải có cấu trúc.

Xử lý ngôn ngữ tự nhiên (NLP) là một công cụ ngôn ngữ trong khoa học máy tính. Nó cho phép máy móc đọc và hiểu ngôn ngữ của con người. NLP cho phép máy tính dịch ngôn ngữ con người thành đầu vào máy tính.

Một số ví dụ về ứng dụng của hệ thống thông minh:

- Bản đồ và điều hướng

Trí tuệ nhân tạo đã cải thiện đáng kể việc di chuyển. Thay vì phải dựa vào bản đồ in hoặc hướng dẫn, bạn có thể sử dụng Google hoặc Apple Maps trên điện thoại và nhập địa điểm đích của bạn.

Vậy làm thế nào ứng dụng biết đi đâu? Và hơn nữa, lộ trình tối ưu, rào cản đường và tắc nghẽn giao thông? Không lâu trước đây, chỉ có GPS dựa trên vệ tinh, nhưng hiện nay, trí tuệ nhân tạo được tích hợp để mang đến trải nghiệm tốt hơn cho người dùng.

Sử dụng học máy, các thuật toán ghi nhớ các biên của các tòa nhà mà nó đã học, cho phép có hình ảnh tốt hơn trên bản đồ và nhận biết, hiểu về số nhà và tòa nhà. Ứng dụng cũng đã được dạy để hiểu và xác định sự thay đổi trong luồng giao thông để nó có thể đề xuất một lộ trình tránh các chướng ngại vật và tắc nghẽn.

- Phát hiện và nhận dạng khuôn mặt

Sử dụng các bộ lọc ảo trên khuôn mặt khi chụp ảnh và sử dụng nhận dạng khuôn mặt để mở khóa điện thoại là hai ví dụ về trí tuệ nhân tạo mà hiện đã trở thành một phần của cuộc sống hàng ngày của chúng ta. Phần trước sử dụng phát hiện khuôn mặt, có nghĩa là nhận diện bất kỳ khuôn mặt người nào. Phần sau sử dụng nhận dạng khuôn mặt thông qua việc nhận dạng một khuôn mặt cụ thể. Nhận dạng khuôn mặt cũng được sử dụng cho giám sát và bảo mật tại các cơ sở chính phủ và sân bay.

- Thuật toán tìm kiếm và đề xuất

Có thể đã sử dụng các công cụ như Grammarly khi làm bài tập để kiểm tra bài luận cuối cùng trước khi nộp cho giáo viên hoặc có thể sử dụng nó ngay bây giờ để kiểm tra chính tả trong email gửi đến sếp của bạn. Đây là một ví dụ khác về trí tuệ nhân tạo. Các thuật toán AI sử dụng học máy, học sâu và xử lý ngôn ngữ tự nhiên để nhận biết việc sử dụng sai ngôn ngữ và đề xuất các sửa lỗi trong các ứng dụng xử lý văn bản, ứng dụng nhắn tin và mọi phương tiện viết khác, có vẻ thế. Ngôn ngữ học và nhà khoa học máy làm việc cùng nhau để dạy máy móc ngữ pháp, giống như bạn đã được dạy ở trường. Các thuật toán được dạy qua dữ liệu ngôn ngữ chất lượng cao, vì vậy khi bạn sử dụng dấu phẩy sai, trình soạn thảo sẽ bắt được.

- Chatbot

Dưới tư cách là khách hàng, tương tác với dịch vụ khách hàng có thể mất thời gian và gây căng thẳng. Đối với các công ty, đó là một phòng ban không hiệu quả thường tốn kém và khó quản lý. Một giải pháp trí tuệ nhân tạo ngày càng phổ biến cho việc này là sử dụng chatbot AI. Các thuật toán được lập trình cho phép máy móc trả lời các câu hỏi thường được hỏi, tiếp nhận và theo dõi đơn hàng, và điều hướng cuộc gọi.

Các chatbot được dạy làm giả vờ như phong cách trò chuyện của các đại diện dịch vụ khách hàng thông qua xử lý ngôn ngữ tự nhiên (NLP). Các chatbot tiên tiến không còn yêu cầu định dạng đầu vào cụ thể nữa (ví dụ: câu hỏi có/không). Chúng có thể trả lời các câu hỏi phức tạp đòi hỏi phản hồi chi tiết. Thực tế, nếu bạn đánh giá xấu cho phản hồi bạn nhận được, bot sẽ xác định sai sót mà nó đã gây ra và sửa nó cho lần sau, đảm bảo sự hài lòng tối đa của khách hàng.

Bài 1.3:

Các hệ thống thông minh đề cập đến những máy móc được phát triển với công nghệ tiên tiến có khả năng nhận thức và phản ứng đối với môi trường xung quanh. Những hệ thống này có thể bao gồm nhiều loại thiết bị, từ những thiết bị tự động đơn giản như máy hút bụi robot Roomba đến các hệ thống phức tạp như phần mềm nhận diện khuôn mặt và các thuật toán gợi ý mua sắm cá nhân của Amazon. Các hệ thống thông minh được phát triển với mục tiêu cho phép máy móc hiểu môi trường của chúng và tương tác một cách hiệu quả với nó.

Có hai khía cạnh chính mà các hệ thống thông minh tập trung vào: việc nhận thức về môi trường và tương tác với nó. Khả năng của những hệ thống này trong việc nhận thức môi trường của họ là quan trọng để đưa ra các quyết định thông thái và thực hiện các hành động phù hợp. Khả năng nhận thức này thường liên quan đến việc xử lý thông tin hình ảnh, hiểu hình ảnh và video, và hiểu dữ liệu thu thập từ môi trường.

Các hệ thống thông minh cũng nhấn mạnh sự tương tác giữa máy móc và người dùng trong môi trường động và thay đổi. Không giống như các hệ thống robot trước đây có sự tự động hạn chế và thực hiện các hành động đã định trước, các robot tự động hiện đại có khả năng cảm nhận môi trường xung quanh và ra quyết định dựa trên môi trường để đạt được các mục tiêu cụ thể.

Ứng dụng của Các Hệ thống Thông minh:

1. Tự động hóa nhà máy: Các hệ thống thông minh đóng vai trò quan trọng trong việc tự động hóa quy trình sản xuất, tạo ra sự hiệu quả cao, độ chính xác và giảm sự can thiệp của con người.

2. Robot trong Lĩnh vực và Dịch vụ: Những hệ thống này được sử dụng trong nhiều ngành như nông nghiệp, xây dựng và khai thác mỏ để thực hiện các nhiệm vụ trong môi trường khó khăn.

3. Robot Hỗ trợ: Các hệ thống thông minh được sử dụng để hỗ trợ những người có khuyết tật trong các hoạt động hàng ngày, nâng cao chất lượng cuộc sống của họ.

4. Ứng dụng Quân sự: Các robot quân sự được sử dụng cho việc do thám, giám sát, tiêu hủy bom và các nhiệm vụ nguy hiểm khác.

5. Chăm sóc Y tế: Robot được sử dụng trong phẫu thuật, chăm sóc bệnh nhân và phục hồi, tăng cường sự chính xác và giảm thiểu rủi ro.

6. Giáo dục: Các hệ thống hướng dẫn thông minh cung cấp trải nghiệm học tập cá nhân cho sinh viên, thích nghi với nhu cầu và tốc độ học của họ.

7. Giải trí: Robot và nhân vật ảo được sử dụng trong ngành giải trí để tạo ra trải nghiệm tương tác và mô phỏng.

8. Kiểm tra Hình ảnh: Các hệ thống kiểm tra hình ảnh tự động được sử dụng để phát hiện khuyết điểm trong sản xuất và đảm bảo chất lượng sản phẩm.

9. Nhận dạng Ký tự: Nhận dạng ký tự quang học (OCR) được sử dụng để chuyển đổi văn bản in hoặc viết tay thành dữ liệu có thể đọc được bởi máy.

10. Nhận dạng Người: Các phương pháp nhận dạng sinh trắc học như nhận dạng khuôn mặt, dấu vân tay và thể kính được sử dụng để xác thực an toàn.

11. Giao Thông Thông minh: Các hệ thống như ô tô tự lái và hệ thống quản lý giao thông cải thiện hiệu suất giao thông và an toàn.

Những Thách thức trong Các Hệ thống Thông minh:

1. Không chắc chắn: Dữ liệu từ cảm biến có thể có nhiễu và không chính xác, gây ra sự không chắc trong quá trình ra quyết định.

2. Thế giới Động: Môi trường thay đổi liên tục, đòi hỏi phải thích nghi và ra quyết định trong thời gian thực.

3. Thời gian tính toán: Các tính toán phức tạp cho quyết định có thể mất thời gian, ảnh hưởng đến phản hồi trong thời gian thực.

4. Bản đồ: Việc chuyển đổi thông tin thế giới 3D thành dữ liệu 2D để xử lý có thể dẫn đến các thách thức trong việc xử lý sự thay đổi góc nhìn, biến đổi ánh sáng và nhiều yếu tố khác.

Nghiên cứu Các Hệ thống Thông minh:

Nghiên cứu về các hệ thống thông minh đòi hỏi kiến thức từ nhiều lĩnh vực khác nhau:

- Lập trình
- Cấu trúc Dữ liệu
- Thuật toán
- Nhận dạng Mẫu
- Học Máy
- Trí tuệ Nhân tạo
- Vật lý
- Phương pháp Số
- Tâm lý học

Kỹ năng toán học vững vàng về lượng giác, đại số tuyến tính, phép tính, thống kê và xác suất là quan trọng. Việc làm quen với Linux cũng rất hữu ích.

Cơ hội trong Các Hệ thống Thông minh:

Hiện nay, có nhu cầu mạnh mẽ về các chuyên gia có kiến thức về công nghệ hệ thống thông minh và biết cách áp dụng nó vào các vấn đề thực tế. Có cơ hội làm việc trong lĩnh vực học thuật, các phòng thí nghiệm quốc gia và chính phủ, cũng như các ngành công nghiệp như các tập đoàn công nghệ lớn (Google,

Microsoft, Intel, IBM). Các tốt nghiệp có khả năng ứng dụng công nghệ hệ thống thông minh vào các vấn đề thực tế đang được tìm kiếm mạnh mẽ trên thị trường việc làm hiện nay.

Bài 1.4:

Hệ thống thông minh đã và đang là một phần quan trọng của cuộc sống hiện đại, từ ứng dụng thông minh như trợ lý ảo đến công nghệ tự lái trong xe hơi. Các hệ thống thông minh không chỉ tồn tại trong môi trường máy tính, mà còn trải dài qua nhiều lĩnh vực, từ y tế đến sản xuất. Bài luận này sẽ đi vào chi tiết phân loại các hệ thống thông minh dựa trên các tiêu chí khác nhau và cung cấp ví dụ về mỗi loại. Dưới đây là một số ví dụ về các hệ thống thông minh

- **Theo mức độ tự động hóa**
    - **Hệ thống Thông Minh Hoàn Toàn (Fully Intelligent Systems):** Đây là loại hệ thống mà hoàn toàn độc lập và có khả năng thực hiện mọi nhiệm vụ mà con người có thể thực hiện. Ví dụ, một hệ thống thông minh hoàn toàn có thể tự động làm mọi công việc trong một nhà máy sản xuất mà không cần sự can thiệp của con người.
    - Hệ thống Thông Minh Phần Cơ Bản (Partially Intelligent Systems): Đây là loại hệ thống mà chỉ có khả năng thực hiện một số nhiệm vụ cụ thể mà chúng được lập trình hoặc đào tạo để thực hiện. Chúng không có khả năng tự động hóa tất cả các nhiệm vụ như hệ thống thông minh hoàn toàn. Ví dụ, một robot công nghiệp có thể được xem là hệ thống thông minh phần cơ bản vì nó được lập trình để thực hiện một loạt các nhiệm vụ sản xuất cụ thể.

- **Phân loại theo khả năng học hỏi và tự cải thiện**
    - **Hệ thống Thông minh có khả năng học hỏi:** Đây là những hệ thống có khả năng tự động học từ dữ liệu và cải thiện hiệu suất thông qua thời gian. Ví dụ, các thuật toán học máy và mạng nơ-ron nhân tạo.
    - **Hệ thống Thông minh không có khả năng học hỏi:** Đây là những hệ thống được lập trình cố định và không có khả năng tự cải thiện thông qua kinh nghiệm. Ví dụ, hệ thống điều khiển tạm thời trong một số ứng dụng công nghiệp.

- **Phân loại theo mức độ tương tác với con người**
    - **Hệ thống Thông minh tương tác yếu:** Đây là các hệ thống mà tương tác với con người hạn chế và thường chỉ theo dạng thông báo cố định. Ví dụ, hệ thống cung cấp thông báo thời tiết hàng ngày.
    - **Hệ thống Thông minh tương tác mạnh:** Đây là các hệ thống có khả năng tương tác chủ động và linh hoạt với con người. Ví dụ, trợ lý ảo như Siri và Google Assistant có khả năng hiểu và đáp ứng yêu cầu của người dùng.

- **Phân loại theo lĩnh vực ứng dụng**
    - Hệ thống Thông minh trong y tế: Các ứng dụng hỗ trợ chẩn đoán bệnh, dự đoán dịch bệnh, và quản lý thông tin sức khỏe cá nhân.
    - Hệ thống Thông minh trong giao thông: Các hệ thống tự lái trong xe hơi, quản lý giao thông và thông tin lưu lượng giao thông.
    - Hệ thống Thông minh trong sản xuất: Các ứng dụng tự động hóa quy trình sản xuất, theo dõi hiệu suất máy móc, và dự đoán lỗi kỹ thuật.
    - Hệ thống Thông minh trong tài chính: Các công cụ dự đoán thị trường tài chính, quản lý rủi ro và giao dịch tự động.

- **Phân loại theo cách thức hoạt động**
    - Hệ thống Thông minh dựa trên luật: Các hệ thống dựa trên quy tắc và logic lập trình sẵn để đưa ra quyết định.
    - Hệ thống Thông minh dựa trên dữ liệu: Các hệ thống sử dụng dữ liệu lớn và thuật toán học máy để dự đoán và tối ưu hóa.

Bài 1.5:

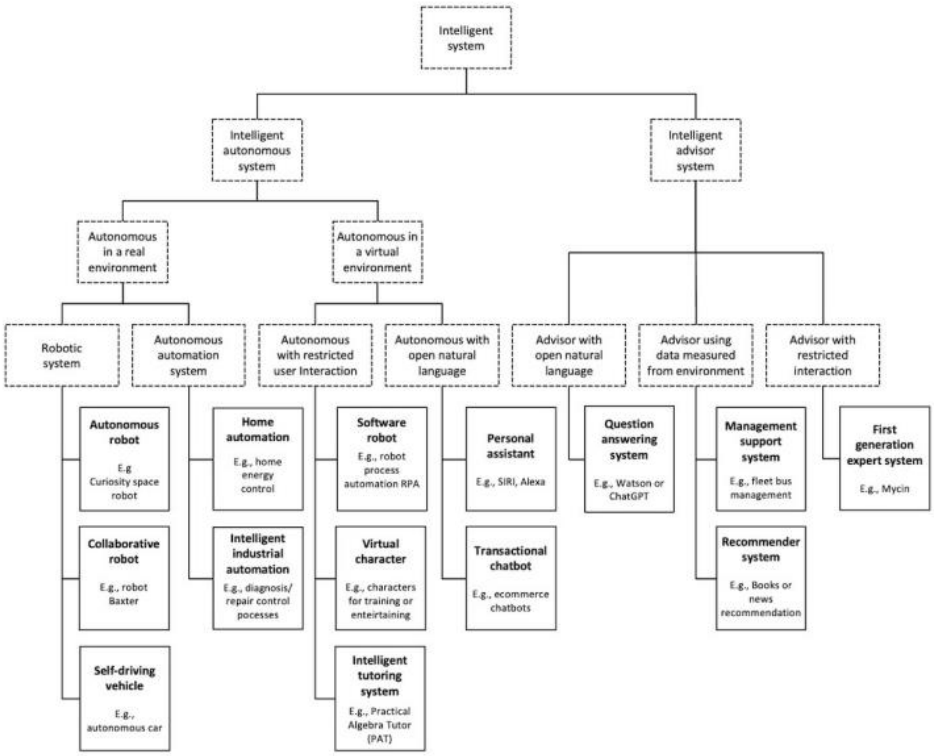


Figure 7: Examples of intelligent systems.

1. Hệ thống tự xử lý thông minh:
   - Trong môi trường thật
     - Hệ thống robot
       - Robot tự xử lí: Những robot này được thiết kế để có khả năng tự động thực hiện nhiều tác vụ phức tạp mà trước đây yêu cầu sự can thiệp của con người. Từ việc thu thập dữ liệu, xử lí thông tin, đưa ra quyết định, đến thực hiện các hành động vật lý, hệ thống robot tự xử lí đã mở ra nhiều triển vọng mới trong các lĩnh vực như sản xuất, y tế, nông nghiệp và nhiều ứng dụng khác.
       - Robot cộng tác: Những hệ thống này kết hợp khả năng tự động hoá và tương tác của robot để làm việc cùng con người trong các môi trường khác nhau. Sự kết hợp này tạo ra một hình thức công nghệ đột phá, mang lại sự linh hoạt, hiệu suất và khả năng giải quyết vấn đề đa dạng. Cùng với sự phát triển của trí tuệ nhân tạo, robot cộng tác đang thúc đẩy sự tiến bộ trong các lĩnh

vực như sản xuất, y tế, nông nghiệp và nhiều ứng dụng quan trọng khác.

        - Xe tự lái: Hệ thống xe tự lái đại diện cho một bước tiến đột phá trong lĩnh vực ô tô và công nghệ, mở ra tương lai hứa hẹn với khả năng thay đổi toàn bộ cách chúng ta di chuyển. Xe tự lái không chỉ là sự kết hợp giữa công nghệ thông minh và ngành công nghiệp ô tô, mà còn là một biểu tượng của sự phát triển vượt bậc trong trí tuệ nhân tạo, xử lý dữ liệu và các hệ thống cảm biến tiên tiến. Bằng cách giảm thiểu tác động của yếu tố con người trong việc lái xe, hệ thống xe tự lái hứa hẹn cải thiện sự an toàn, hiệu suất giao thông và tiện ích cuộc sống hàng ngày của chúng ta.
    - Hệ thống tự động hóa
        - Nhà thông minh, tự động: Đây là các hệ thống tự động hoá được áp dụng trong ngôi nhà, cho phép điều khiển và quản lý các thiết bị và hệ thống như ánh sáng, nhiệt độ, an ninh, âm thanh, và các thiết bị gia đình khác. Nhà tự động tạo ra sự tiện lợi và tiết kiệm năng lượng cho người dùng.
        - Tự động hóa công nghiệp thông minh: Đây là hệ thống tự động hoá được áp dụng trong môi trường công nghiệp để tự động hóa các quy trình và hoạt động sản xuất. Hệ thống này sử dụng các cảm biến, máy móc và phần mềm để kiểm soát và điều khiển các máy móc, dây chuyền sản xuất và quy trình công việc, từ đó nâng cao hiệu suất và độ chính xác.
- Trong môi trường ảo
    - Tự động hóa với tương tác người dùng bị hạn chế
        - Phần mềm robot: Phần mềm được phát triển để mô phỏng và điều khiển các robot trong môi trường ảo.
        - Nhân vật ảo: Là các đại diện số hóa của con người hoặc hệ thống, để tương tác giữa người dùng với người dùng hoặc người dùng với hệ thống trong môi trường ảo.
        - Hệ thống hướng dẫn thông minh: Cung cấp hướng dẫn và hỗ trợ thông minh cho người dùng trong môi trường ảo.
    - Tự động hóa sử dụng ngôn ngữ tự nhiên mở

        - Trợ lý cá nhân: Các hệ thống AI được tích hợp sẵn trong các thiết bị di động, thiết bị cá nhân. Ví dụ tiêu biểu là Siri và Alexa.
        - Chatbot: Được thiết kế để tự động trò chuyện và giao tiếp cơ bản với người dùng qua ứng dụng tin nhắn hoặc trang web.
2. Hệ thống đề xuất, tư vấn thông minh:
    - Tư vấn sử dụng ngôn ngữ tự nhiên mở:
        - Hệ thống trả lời câu hỏi: Đây là các hệ thống trí tuệ nhân tạo được phát triển để trả lời các câu hỏi từ người dùng thông qua ngôn ngữ tự nhiên. Hệ thống này có khả năng hiểu và phân tích câu hỏi, tìm kiếm thông tin, và cung cấp câu trả lời chính xác và đáng tin cậy.
    - Tư vấn sử dụng dữ liệu đo lường từ môi trường:
        - Hệ thống hỗ trợ quản lý: Đây là các hệ thống trí tuệ nhân tạo được sử dụng để thu thập, phân tích và quản lý dữ liệu từ môi trường. Chúng có khả năng cung cấp thông tin và gợi ý để hỗ trợ quyết định và quản lý hiệu quả các hoạt động và quy trình.
        - Hệ thống gợi ý: Đây là các hệ thống trí tuệ nhân tạo có khả năng phân tích dữ liệu và dự đoán để đưa ra gợi ý và khuyến nghị cho người dùng. Chúng có thể cung cấp gợi ý về sản phẩm, nội dung, hoặc hành động dựa trên sự phân tích dữ liệu và mô hình học máy.
    - Tư vấn với tương tác bị hạn chế:
        - Hệ thống chuyên gia thế hệ đầu tiên: Đây là các hệ thống được phát triển dựa trên kiến thức chuyên gia trong một lĩnh vực cụ thể. Chúng có khả năng cung cấp lời khuyên, đánh giá và hướng dẫn với mức độ tương tác hạn chế từ người dùng. Hệ thống chuyên gia thế hệ đầu tiên thường được sử dụng trong các lĩnh vực như y tế, kỹ thuật, tài chính, luật pháp và quản lý tri thức.

Bài 1.6:

1. **Numpy**:
   Giới thiệu: NumPy (Numerical Python) là một thư viện quan trọng trong ngôn ngữ lập trình Python, chuyên về xử lý và tính toán khoa học số học và dữ liệu đa chiều. Với NumPy, người dùng có thể thực hiện các phép toán số học, đại số tuyến tính và xử lý dữ liệu mảng một cách hiệu quả.

Đặc trưng: NumPy chú trọng vào việc sử dụng mảng (array) đa chiều, gọi là **ndarray** (n-dimensional array). Các mảng này cho phép lưu trữ dữ liệu trong các cấu trúc đa chiều giúp tận dụng hiệu suất tính toán của máy tính.
Mục đích: NumPy được thiết kế để hỗ trợ xử lý dữ liệu khoa học và toán học một cách hiệu quả. Điều này bao gồm các tính toán vector và ma trận, phép toán số học, biến đổi Fourier, thống kê, và nhiều tính năng khác. Ngoài ra, NumPy cũng là một phần quan trọng trong việc tích hợp Python với các thư viện và framework khoa học khác như SciPy, pandas, và scikit-learn.
Ví dụ

```
import numpy as np

# Tạo một mảng 1 chiều từ danh sách
array_1d = np.array([1, 2, 3, 4, 5])

# Tạo một ma trận 2 chiều
matrix_2d = np.array([[1, 2, 3], [4, 5, 6]])

# Phép toán trên mảng
result = array_1d + 10

# Tích vô hướng giữa hai mảng
dot_product = np.dot(array_1d, result)

# Tính trung bình cộng các phần tử trong mảng
mean_value = np.mean(array_1d)

# Tính độ lệch chuẩn của mảng
std_deviation = np.std(array_1d)
```

2. **Pandas**:
   Giới thiệu: Pandas là một thư viện quan trọng trong ngôn ngữ lập trình Python dành cho xử lý và phân tích dữ liệu. Thư viện này cung cấp các cấu trúc dữ liệu và công cụ mạnh mẽ để làm việc với dữ liệu có cấu trúc, như dữ liệu dạng bảng.
   Đặc trưng: Pandas tập trung vào hai cấu trúc dữ liệu chính là Series và DataFrame:

- **Series:** Là một mảng một chiều có khả năng gán nhãn (label), giúp dễ dàng truy cập và xử lý dữ liệu theo tên các nhãn.
- **DataFrame:** Là một bảng hai chiều dạng lưới, giúp tổ chức và xử lý dữ liệu dạng bảng với các cột có thể có kiểu dữ liệu khác nhau.

Mục đích: Pandas được thiết kế để giúp người dùng thực hiện các công việc liên quan đến xử lý dữ liệu và phân tích dữ liệu một cách dễ dàng và hiệu quả. Thư viện này hỗ trợ việc đọc và ghi dữ liệu từ nhiều nguồn khác nhau, lọc, sắp xếp, thống kê, và thực hiện các phép biến đổi phức tạp trên dữ liệu.

Ví dụ:

```
import pandas as pd

# Tạo một Series từ danh sách
data = [10, 20, 30, 40, 50]
series = pd.Series(data, index=['A', 'B', 'C', 'D', 'E'])

# Tạo một DataFrame từ dictionary
data_dict = {'Name': ['Alice', 'Bob', 'Charlie'],
             'Age': [25, 30, 22]}
data_frame = pd.DataFrame(data_dict)

# Lọc dữ liệu trong DataFrame
filtered_data = data_frame[data_frame['Age'] > 24]

# Đọc dữ liệu từ file CSV
csv_data = pd.read_csv('data.csv')

# Tính tổng các giá trị trong Series
total_sum = series.sum()

# Tính trung bình cộng tuổi trong DataFrame
average_age = data_frame['Age'].mean()
```

3. **Matplotlib:**

Giới thiệu: Matplotlib là một thư viện trong ngôn ngữ lập trình Python được sử dụng rộng rãi để tạo ra các biểu đồ, đồ thị và hình ảnh trực quan từ dữ liệu. Thư viện này cung cấp các công cụ mạnh mẽ để hiển thị dữ liệu dưới dạng hình ảnh, giúp người dùng dễ dàng thể hiện và trình bày thông tin.

Đặc trưng: Matplotlib hỗ trợ tạo ra các loại biểu đồ và đồ thị phong phú, bao gồm biểu đồ đường, cột, scatter, 3D, hình ảnh, đồ thị trực quan, và nhiều loại khác. Thư viện này cũng cho phép tùy chỉnh hoàn toàn giao diện đồ họa của biểu đồ, từ màu sắc, font chữ cho đến tiêu đề và chú thích.
Mục đích: Mục đích chính của Matplotlib là giúp người dùng dễ dàng tạo ra các biểu đồ và đồ thị trực quan từ dữ liệu số học, thống kê và khoa học. Thư viện này thường được sử dụng trong việc trình bày kết quả nghiên cứu, phân tích dữ liệu, và thể hiện mô hình số học.
Ví dụ:

```
import matplotlib.pyplot as plt
import numpy as np

# Tạo dữ liệu mẫu
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Tạo biểu đồ đường
plt.plot(x, y, label='sin(x)')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Sine Function')
plt.legend()
plt.show()
```

4. **Scikit-learn**

Giới thiệu: Scikit-learn là một thư viện mã nguồn mở phát triển trên nền tảng Python, tập trung vào học máy (machine learning) và khai phá dữ liệu. Thư viện này cung cấp các công cụ và thuật toán tiêu chuẩn cho nhiều nhiệm vụ liên quan đến học máy và phân tích dữ liệu.
Đặc trưng: Scikit-learn cung cấp một loạt các thuật toán học máy phổ biến, từ học có giám sát đến học không giám sát, và từ phân loại đến hồi quy. Nó cũng chứa các công cụ để tiền xử lý dữ liệu, chọn đặc trưng, và đánh giá mô hình.
Mục đích: Scikit-learn nhằm giúp các nhà nghiên cứu, phân tích dữ liệu và nhà phát triển xây dựng và đánh giá các mô hình học máy một cách dễ dàng và hiệu quả. Thư viện này tập trung vào việc đơn giản hóa quá trình xây

dựng mô hình học máy, giúp người dùng tập trung vào việc hiểu và áp dụng các thuật toán.

Ví dụ:

```
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

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

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

# Xây dựng mô hình phân loại K-Nearest Neighbors
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train, y_train)

# Dự đoán trên tập test
y_pred = knn_model.predict(X_test)

# Đánh giá hiệu suất mô hình
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

1 Homework

1.1 Import

[3]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style
from sklearn.datasets import fetch_openml, make_classification
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from mpl_toolkits.mplot3d import Axes3D
```

1.2 1.7

[4]:
```
def check_bmi(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif bmi < 25:
        return 'Normal weight'
    elif bmi < 30:
        return 'Overweight'
```

[5]:
```
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
bmi
```

[5]:
```
array([28.88888889, 14.51836889, 23.046875  ])
```

[6]:
```
df = pd.DataFrame({'Name': names, 'Height': heights, 'Weight': weights, 'BMI':␣
↪bmi})
df
```

[6]:

	Name	Height	Weight	BMI
0	Ann	1.50	65	28.888889
1	Joe	1.78	46	14.518369
2	Mark	1.60	59	23.046875

[7]:
```
classify = np.vectorize(check_bmi)
classify(bmi)
```

[7]:
```
array(['Overweight', 'Underweight', 'Normal weight'], dtype='<U13')
```

[8]:
```
df2 = pd.DataFrame({
  'Name': names,
  'Height': heights,
  'Weight': weights,
  'BMI': bmi,
  'Classify': classify(bmi)})
df2
```

[8]:

	Name	Height	Weight	BMI	Classify
0	Ann	1.50	65	28.888889	Overweight
1	Joe	1.78	46	14.518369	Underweight
2	Mark	1.60	59	23.046875	Normal weight

1.2.1 Data from group

[9]:
```
data = pd.read_csv('data/no1_7.csv')
data
```

[9]:

	name	height	weight
0	anhnt	1.66	72
1	vphuong	1.78	65
2	vu	1.68	60
3	nam	1.69	65
4	dphuong	1.67	60

[10]:
```
names = data['name'].values
heights = data['height'].values
weights = data['weight'].values
```

[11]:
```
bmi = weights / heights ** 2
bmi
```

[11]:
```
array([26.12861083, 20.51508648, 21.2585034 , 22.75830678, 21.51385851])
```

[12]:
```
df3 = pd.DataFrame({
  'Name': names,
  'Height': heights,
  'Weight': weights,
  'BMI': bmi,
  'Classify': classify(bmi)})
df3
```

[12]:

	Name	Height	Weight	BMI	Classify
0	anhnt	1.66	72	26.128611	Overweight
1	vphuong	1.78	65	20.515086	Normal weight
2	vu	1.68	60	21.258503	Normal weight
3	nam	1.69	65	22.758307	Normal weight
4	dphuong	1.67	60	21.513859	Normal weight

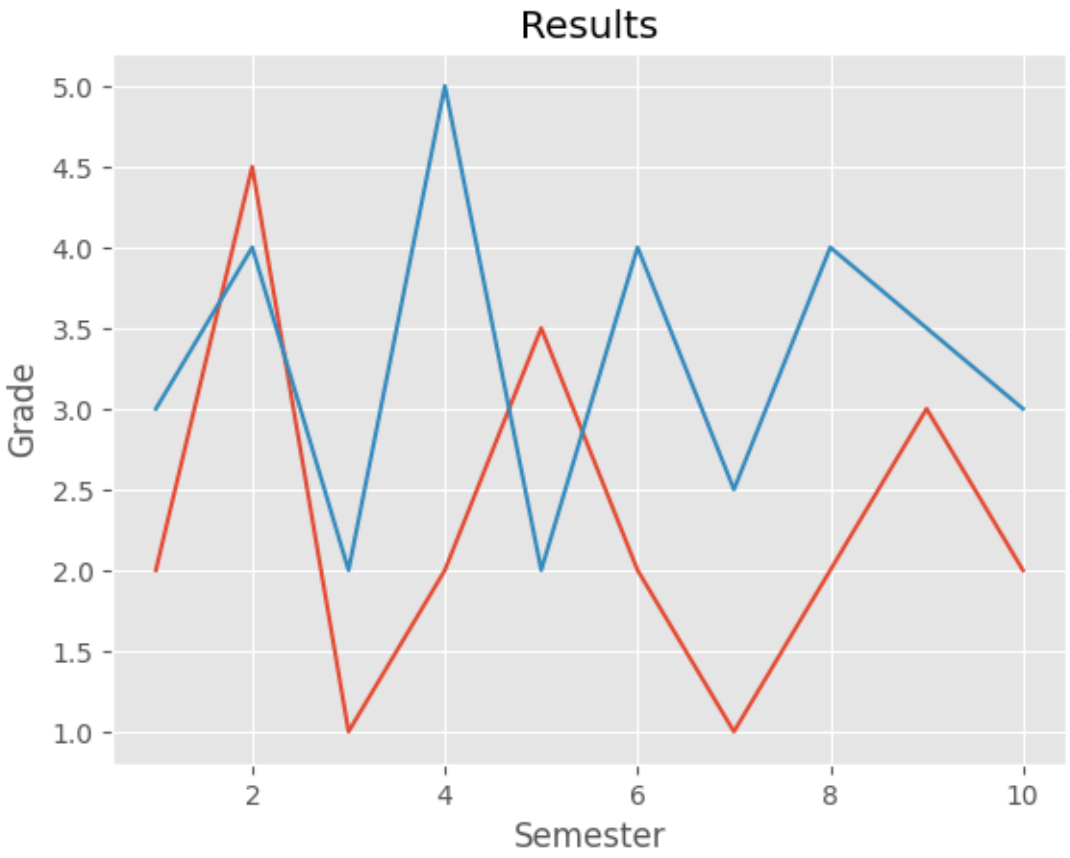
1.3 1.8

Plotting multiple Lines in the same chart

[13]:
```
style.use('ggplot')
```

[14]:
```
plt.plot(
   [1,2,3,4,5,6,7,8,9,10],
   [2,4.5,1,2,3.5,2,1,2,3,2]
)

plt.plot(
   [1,2,3,4,5,6,7,8,9,10],
   [3,4,2,5,2,4,2.5,4,3.5,3]
)

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
```

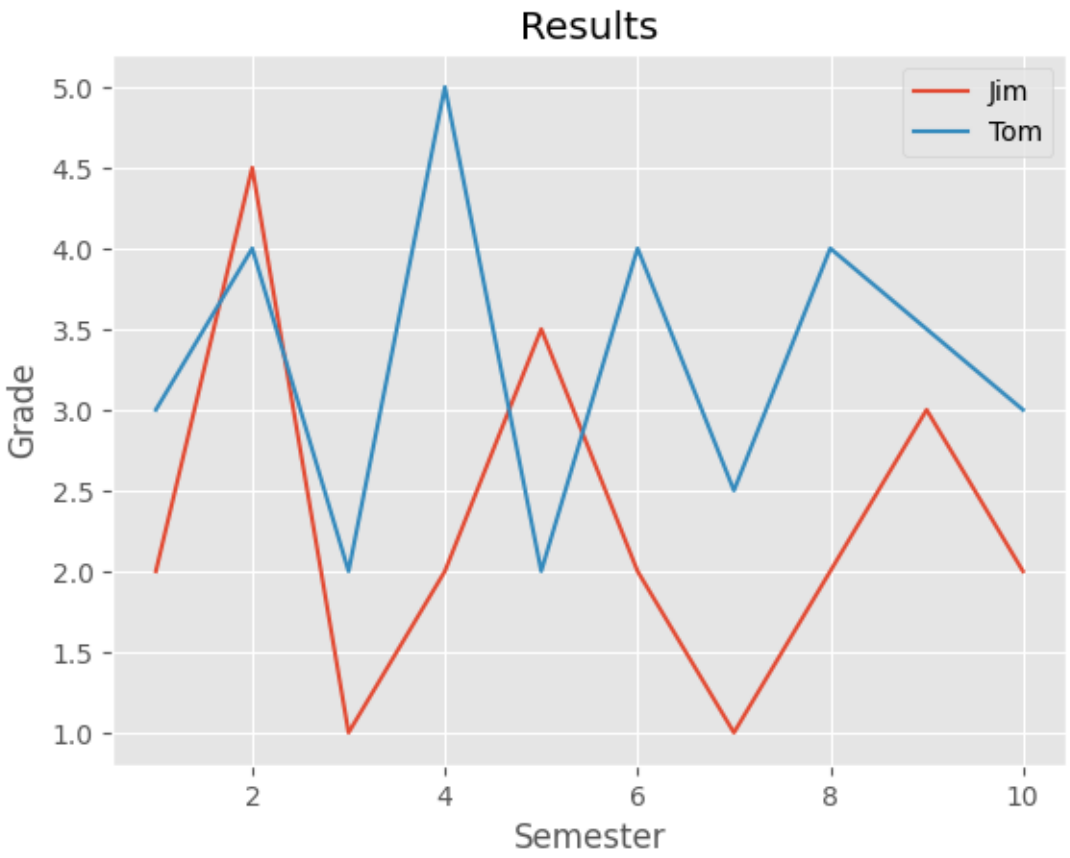
[14]:
```
Text(0, 0.5, 'Grade')
```



Adding a Legend

[15]:
```
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [2,4.5,1,2,3.5,2,1,2,3,2],
  label="Jim"
)
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [3,4,2,5,2,4,2.5,4,3.5,3],
  label="Tom"
)

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
plt.legend()
```

[15]:
```
<matplotlib.legend.Legend at 0x1bfce2274c0>
```



Plotting bar charts

[16]:
```
plt.bar(
    [1,2,3,4,5,6,7,8,9,10],
    [2,4.5,1,2,3.5,2,1,2,3,2],
    label = "Jim",
    color = "m",                        # m for magenta
    align = "center"
)

plt.title("Results")
plt.xlabel("Semester")
plt.ylabel("Grade")

plt.legend()
plt.grid(True, color="y")
```

1.3.1 Data from team

[17]:
```
df = pd.read_csv('./data/no1_8.csv')
df
```

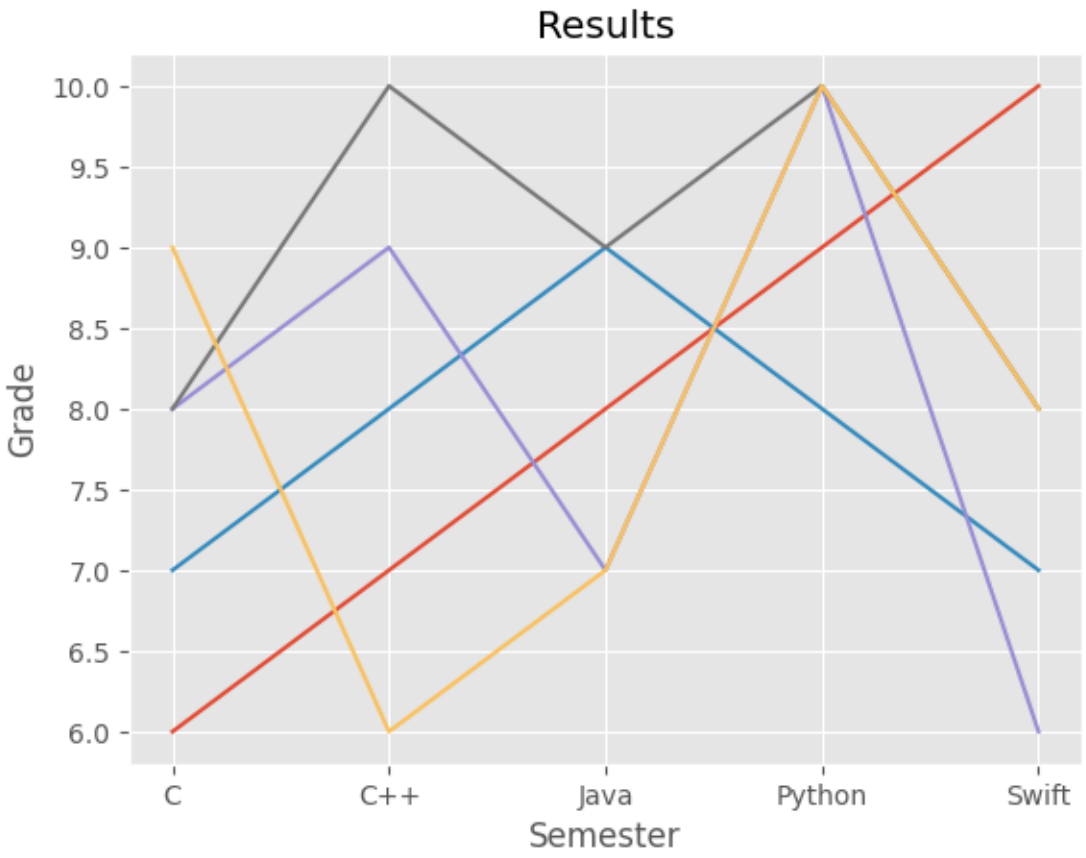
[17]:

	Subject	tienanh	vphuong	vu	nam	dphuong
0	C	6	7	8	8	9
1	C++	7	8	9	10	6
2	Java	8	9	7	9	7
3	Python	9	8	10	10	10
4	Swift	10	7	6	8	8

[18]:
```
subjects = df["Subject"]
tienanh_grade = df["tienanh"]
vphuong_grade = df["vphuong"]
vu_grade = df["vu"]
nam_grade = df["nam"]
dphuong_grade = df["dphuong"]
```

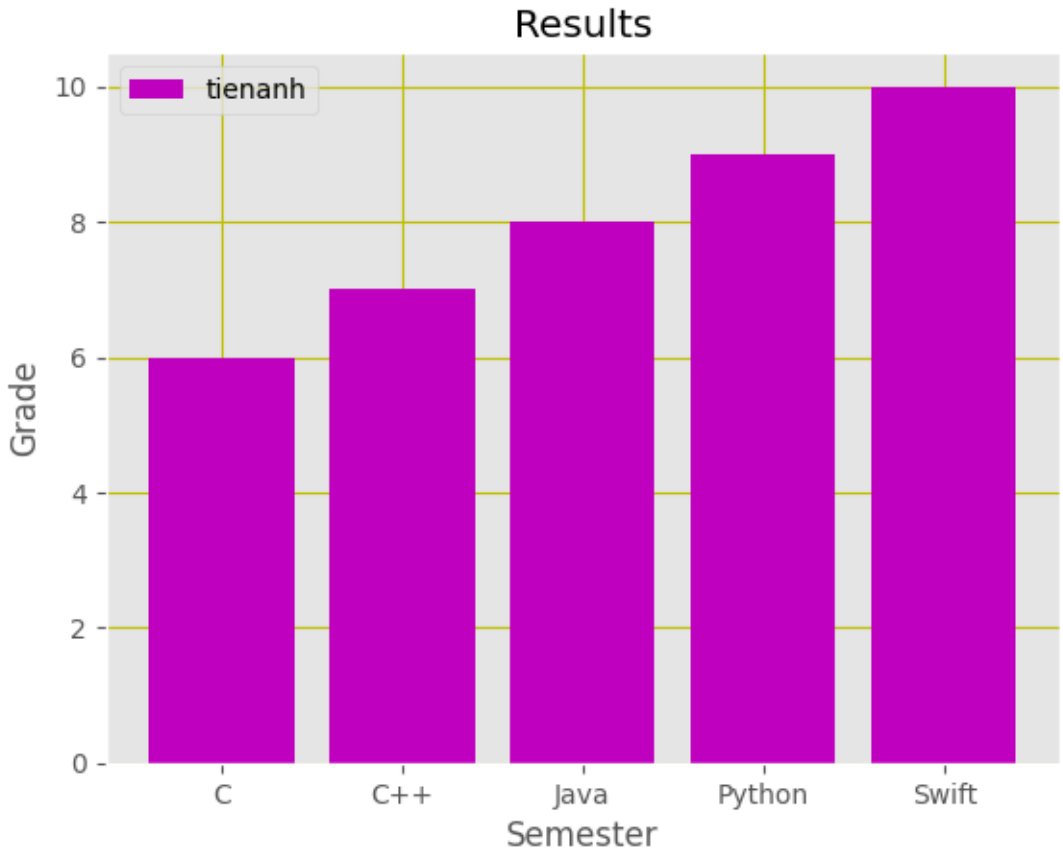
[19]:
```
plt.plot(
  subjects,
  tienanh_grade,
  label="tienanh",
  )
plt.plot(
  subjects,
  vphuong_grade,
  label="vphuong",
  )
plt.plot(
  subjects,
  vu_grade,
  label="vu",
  )
plt.plot(
  subjects,
  nam_grade,
  label="nam",
  )
plt.plot(
  subjects,
  dphuong_grade,
  label="dphuong",
  )

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
```

[19]: Text(0, 0.5, 'Grade')



[20]:
```
style.use('ggplot')

subjects = subjects.tolist()
subjects

plt.bar(
  subjects,
  tienanh_grade,
  label="tienanh",
  color = "m",
  align = "center"
  )

plt.title('Results')
plt.xlabel('Semester')
plt.ylabel('Grade')
```

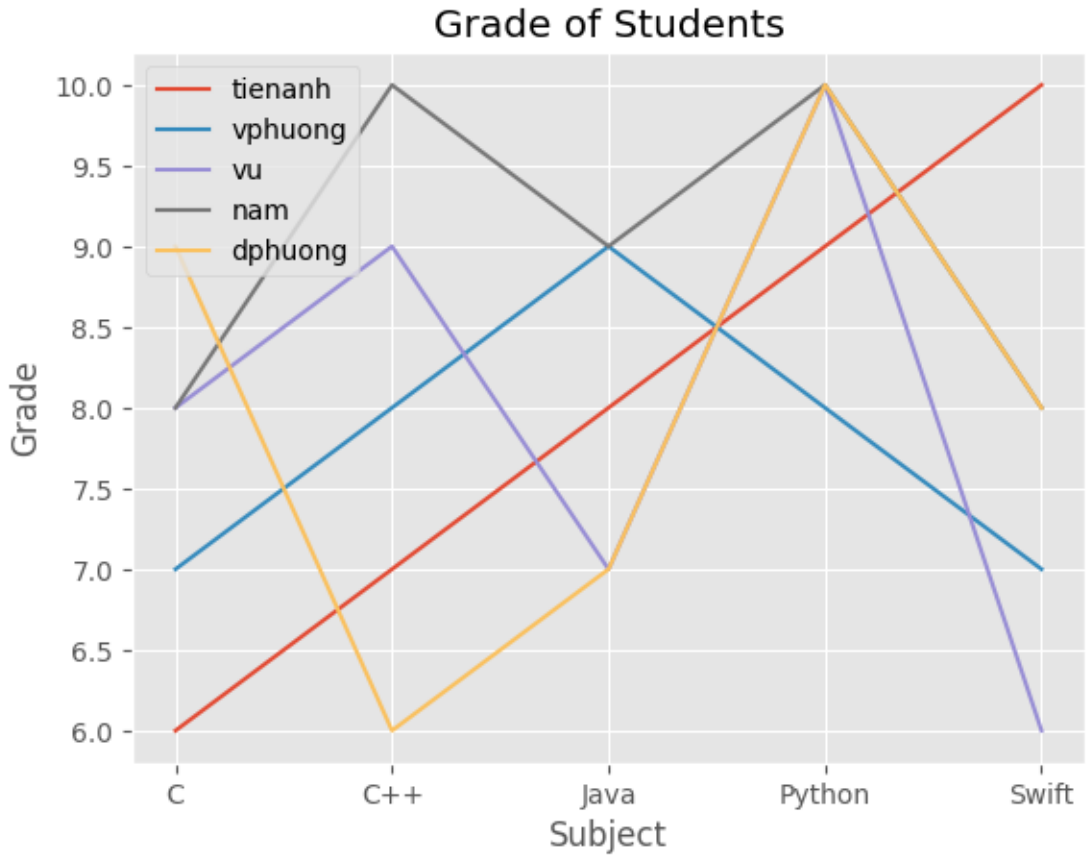
```
plt.legend()
plt.grid(True, color="y")
```



[21]:
```
plt.plot(
  subjects,
  tienanh_grade,
  label="tienanh",
  )
plt.plot(
  subjects,
  vphuong_grade,
  label="vphuong",
  )
plt.plot(
  subjects,
  vu_grade,
  label="vu",
  )
plt.plot(
```

```
  subjects,
  nam_grade,
  label="nam",
  )
plt.plot(
  subjects,
  dphuong_grade,
  label="dphuong",
  )

plt.xlabel("Subject")
plt.ylabel("Grade")
plt.title("Grade of Students")
plt.legend()
```

[21]: <matplotlib.legend.Legend at 0x1bfcf2cbeb0>



1.4 1.9

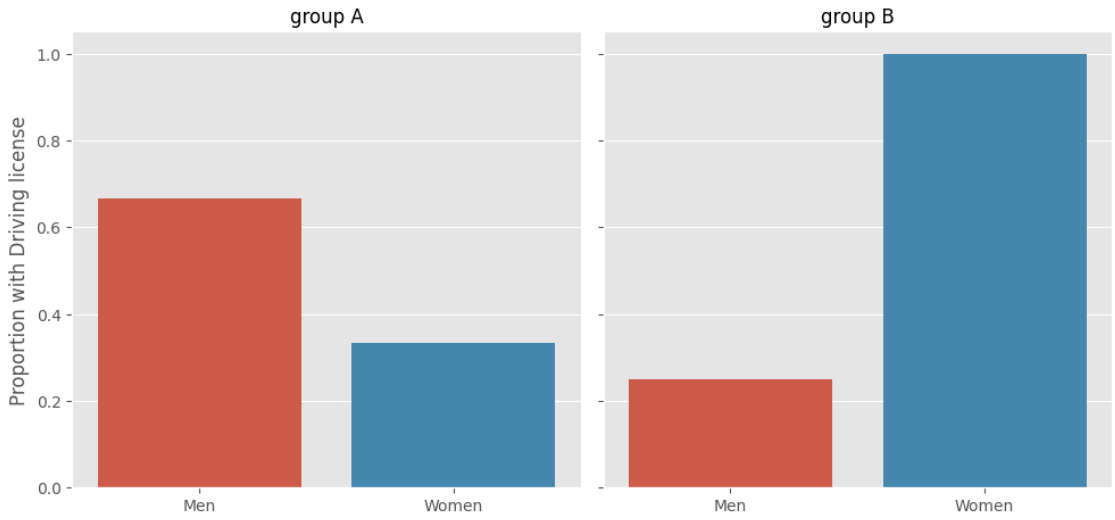
[22]:
```
df = pd.read_csv('data/no1_9.csv')
df
```

[22]:

	gender	group	license
0	men	A	1
1	men	A	0
2	men	A	1
3	women	A	1
4	women	A	0
5	women	A	0
6	men	B	0
7	men	B	0
8	men	B	0
9	men	B	1
10	women	B	1
11	women	B	1
12	women	B	1
13	women	B	1

[23]:
```
g = sns.catplot(x="gender", y="license", col="group",
                data = df, kind="bar", errorbar=None, aspect=1.0)

#--- set the labels ---
g.set_axis_labels("", "Proportion with Driving license")
g.set_xticklabels(["Men", "Women"])
g.set_titles("{col_var} {col_name}")

#--- show plot ---
plt.show()
```

```
c:\Users\tien2\miniconda3\lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



1.5 1.10

[24]:
```
df = pd.read_csv('data/no1_10.csv')
df
```

[24]:

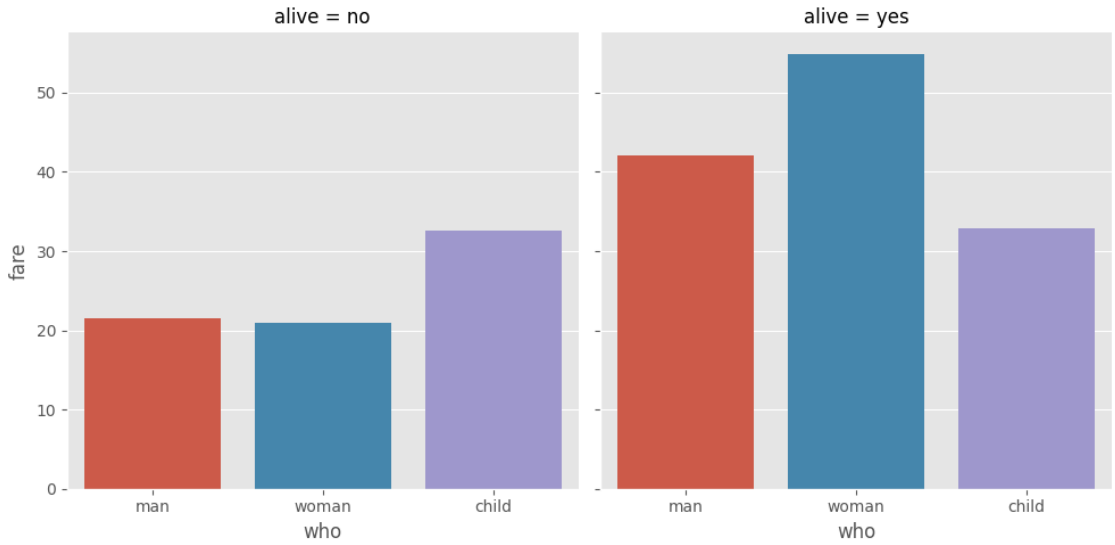
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	
..	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	
887	1	1	female	19.0	0	0	30.0000	S	First	
888	0	3	female	NaN	1	2	23.4500	S	Third	
889	1	1	male	26.0	0	0	30.0000	C	First	
890	0	3	male	32.0	0	0	7.7500	Q	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True
..	...	...	...	...	...	...
886	man	True	NaN	Southampton	no	True
887	woman	False	B	Southampton	yes	True
888	woman	False	NaN	Southampton	no	False
889	man	True	C	Cherbourg	yes	True

```
890    man        True  NaN    Queenstown    no    True

[891 rows x 15 columns]
```

[25]:
```
g = sns.catplot(x="who", y="fare", col="alive",
                data=df, kind="bar", errorbar=None, aspect=1.0)
```

```
c:\Users\tien2\miniconda3\lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



1.6 1.11

[26]:
```
sns.set_style("whitegrid")

#---load data---
data = pd.read_csv('data/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()
```

1.7 1.12

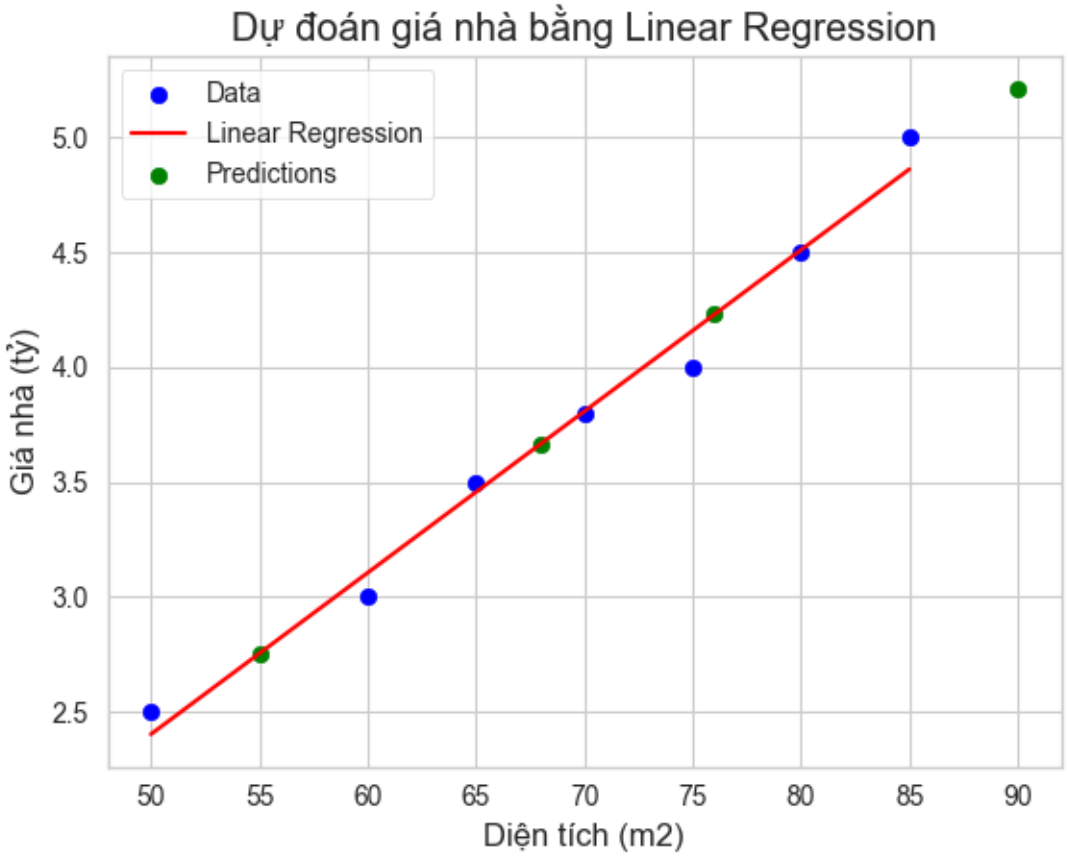
[27]:
```
data = np.array([(50, 2.5), (60, 3), (65, 3.5), (70, 3.8), (75, 4), (80, 4.5),␣
↪(85, 5)])
data
```

[27]:
```
array([[50. ,  2.5],
       [60. ,  3. ],
       [65. ,  3.5],
       [70. ,  3.8],
       [75. ,  4. ],
       [80. ,  4.5],
       [85. ,  5. ]])
```

[28]:
```
X = data[:,0].reshape(-1,1)
y = data[:,1]

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

```
new_areas = np.array([55, 68, 76, 90]).reshape(-1, 1)
predicted_prices = model.predict(new_areas)

for area, price in zip(new_areas, predicted_prices):
    print(f"Din tích: {area} m2, Giá d đoán: {price:.2f} t")

plt.scatter(X, y, color='blue', label='Data')
plt.plot(X, model.predict(X), color='red', label='Linear Regression')
plt.scatter(new_areas, predicted_prices, color='green', label='Predictions')
plt.xlabel('Din tích (m2)')
plt.ylabel('Giá nhà (t)')
plt.title('D đoán giá nhà bng Linear Regression')
plt.legend()
plt.show()
```

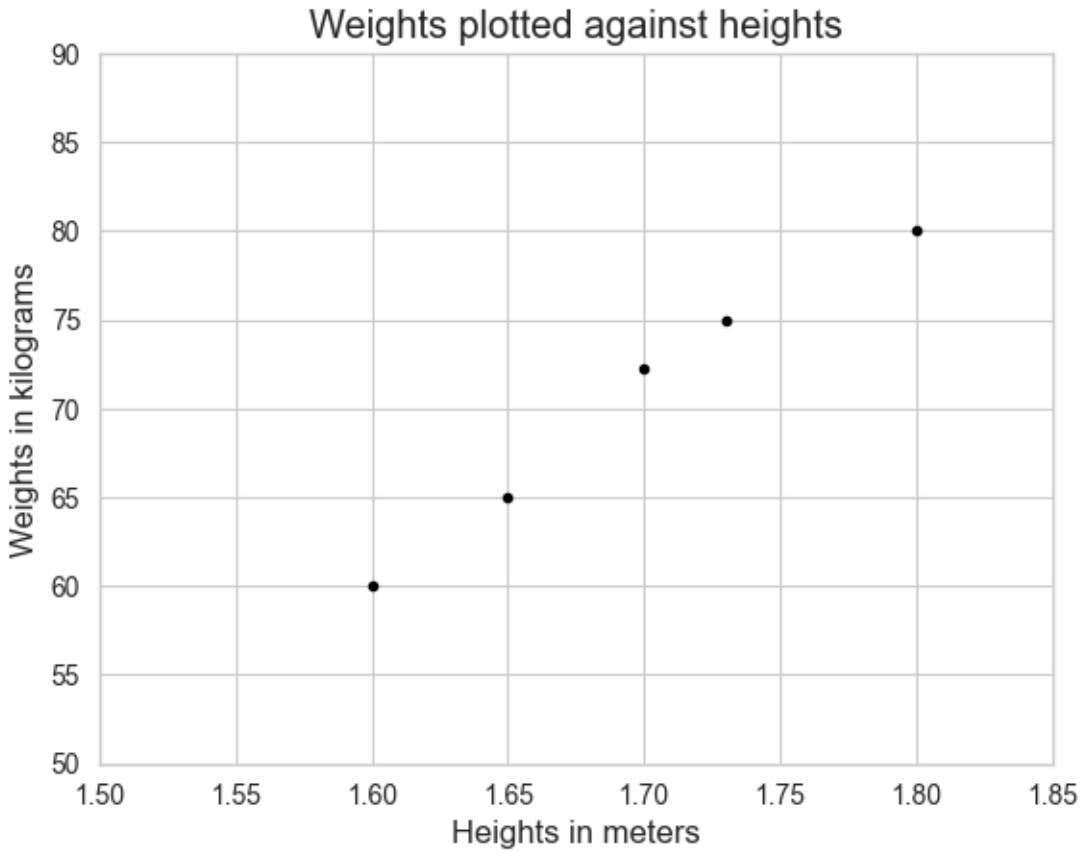
```
Din tích: [55] m2, Giá d đoán: 2.75 t
Din tích: [68] m2, Giá d đoán: 3.67 t
Din tích: [76] m2, Giá d đoán: 4.23 t
Din tích: [90] m2, Giá d đoán: 5.21 t
```



1.8 1.13

[29]:
```
heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]

weights = [[60], [65], [72.3], [75], [80]]
```

[30]:
```
# represents the weights of a group of people in kgs
weights = [[60], [65], [72.3], [75], [80]]

plt.title('Weights plotted against heights')
plt.xlabel('Heights in meters')
plt.ylabel('Weights in kilograms')

plt.plot(heights, weights, 'k.')

# axis range for x and y
plt.axis([1.5, 1.85, 50, 90])
plt.grid(True)
```

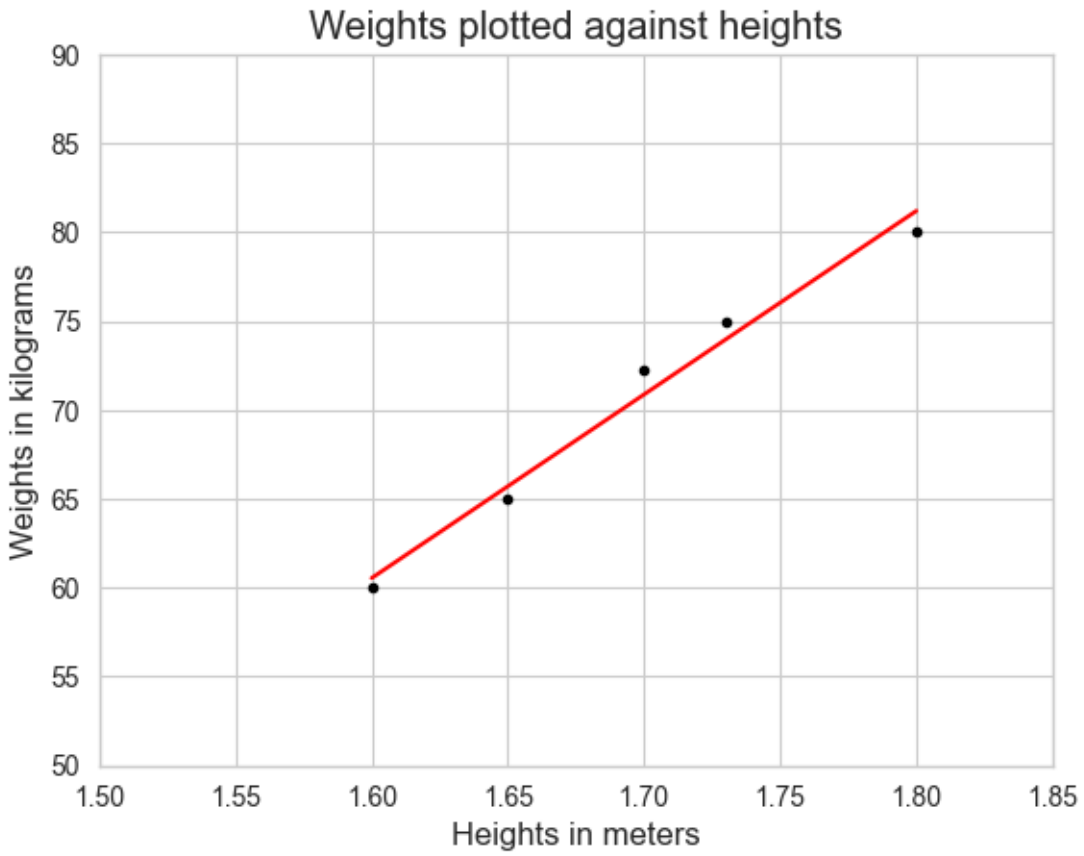


[31]:
```
model = LinearRegression()
model.fit(X=heights, y=weights)

weight = model.predict([[1.75]])[0][0]
print(f'Predicted weight for height 1.75 m: {round(weight,2)} kg')
```

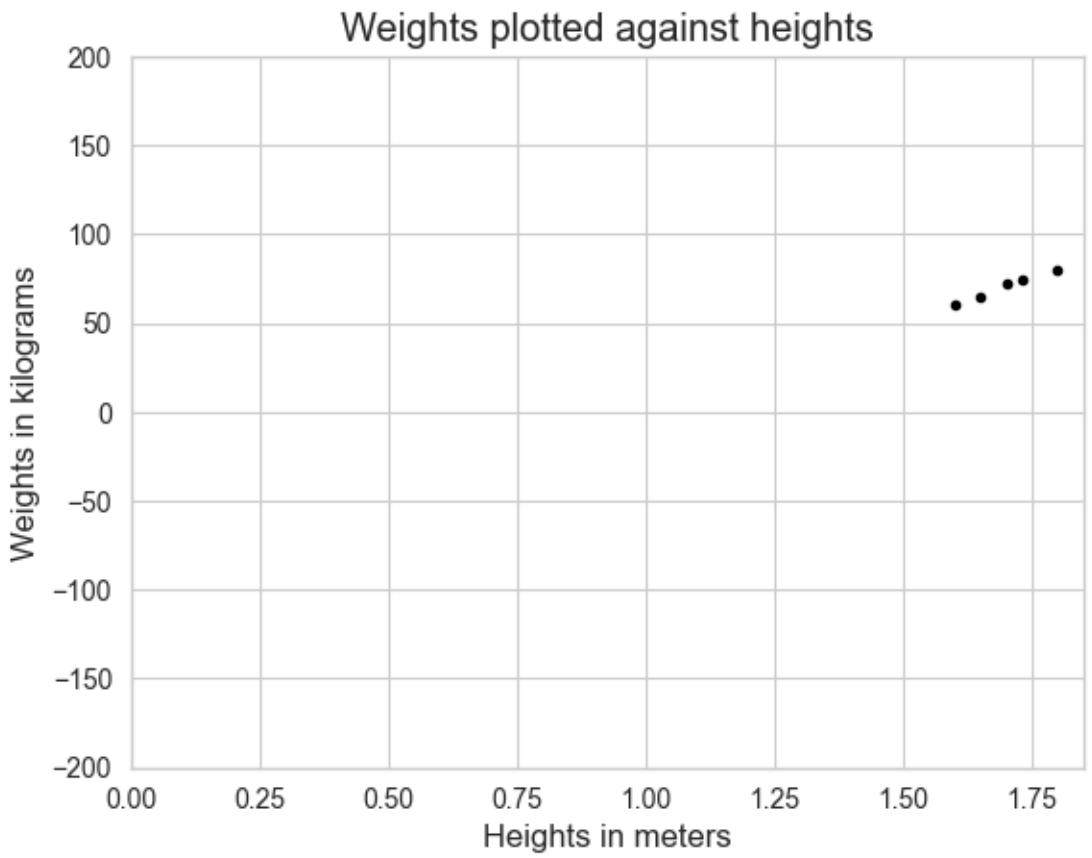
```
Predicted weight for height 1.75 m: 76.04 kg
```

[32]:
```
import matplotlib.pyplot as plt

heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]
weights = [[60], [65], [72.3], [75], [80]]

plt.title('Weights plotted against heights')
plt.xlabel('Heights in meters')
plt.ylabel('Weights in kilograms')
plt.plot(heights, weights, 'k.')

plt.axis([1.5, 1.85, 50, 90])
plt.grid(True)

# plot the regression line
plt.plot(heights, model.predict(heights), color='r')

round(model.predict([[0]])[0][0],2) # -104.75

print(round(model.intercept_[0],2)) # -104.75

print(round(model.coef_[0][0],2)) # 103.31
```

```
-104.75
103.31
```



[33]:
```
plt.title('Weights plotted against heights')
plt.xlabel('Heights in meters')
plt.ylabel('Weights in kilograms')

plt.plot(heights, weights, 'k.')

plt.axis([0, 1.85, -200, 200])
plt.grid(True)
```



[34]:
```
import numpy as np

print('Residual sum of squares: %.2f' %
      np.sum((weights - model.predict(heights)) ** 2))
```

```
Residual sum of squares: 5.34
```

[35]:
```
# test data
heights_test = [[1.58], [1.62], [1.69], [1.76], [1.82]]
weights_test = [[58], [63], [72], [73], [85]]
```

[36]:
```
# Total Sum of Squares (TSS)
weights_test_mean = np.mean(np.ravel(weights_test))
TSS = np.sum((np.ravel(weights_test) -
              weights_test_mean) ** 2)
print("TSS: %.2f" % TSS)

# Residual Sum of Squares (RSS)
RSS = np.sum((np.ravel(weights_test) -
              np.ravel(model.predict(heights_test)))
```

```
                  ** 2)
print("RSS: %.2f" % RSS)

# R_squared
R_squared = 1 - (RSS / TSS)
print("R-squared: %.2f" % R_squared)

# using scikit-learn to calculate r-squared
print('R-squared: %.4f' % model.score(heights_test,
                                      weights_test))
```

```
TSS: 430.80
RSS: 24.62
R-squared: 0.94
R-squared: 0.9429
```

[37]:
```
import pickle

# save the model to disk
filename = './data/HeightsAndWeights_model.sav'
# write to the file using write and binary mode
pickle.dump(model, open(filename, 'wb'))
```

[38]:
```
# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(heights_test,
                            weights_test)
result
```

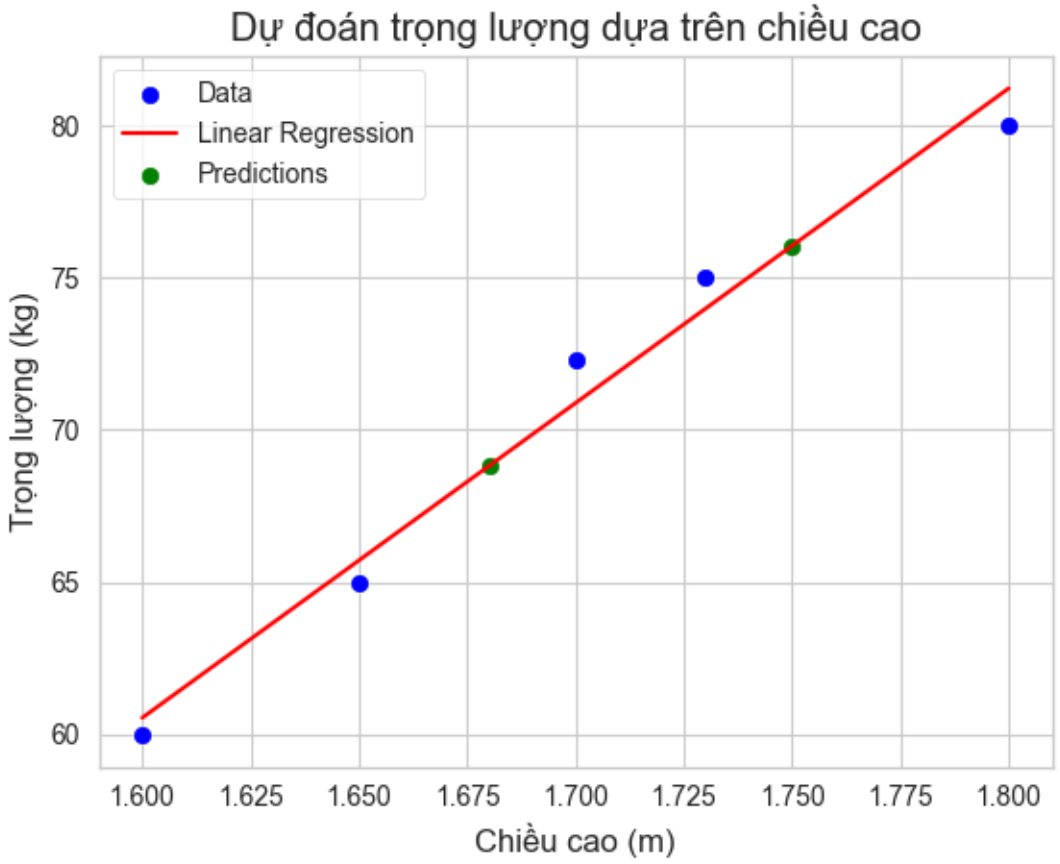
[38]:
```
0.9428592885995254
```

1.8.1 Personal records

[39]:
```
heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]

weights = [[60], [65], [72.3], [75], [80]]

model = LinearRegression()
model.fit(heights, weights)

# Predict weights for new heights
new_heights = np.array([[1.68], [1.75]])
predicted_weights = model.predict(new_heights)

# Display predictions
for height, weight in zip(new_heights, predicted_weights):
    print(f"Chiu cao: {height[0]} m, Trng lng d đoán: {weight[0]:.2f} kg")
```

```
Chiu cao: 1.68 m, Trng lng d đoán: 68.81 kg
Chiu cao: 1.75 m, Trng lng d đoán: 76.04 kg
```

[40]:
```
# Visualization
plt.scatter(heights, weights, color='blue', label='Data')
plt.plot(heights, model.predict(heights), color='red', label='Linear Regression')
plt.scatter(new_heights, predicted_weights, color='green', label='Predictions')
plt.xlabel('Chiu cao (m)')
plt.ylabel('Trng lng (kg)')
plt.title('D đoán trng lng da trên chiu cao')
plt.legend()
plt.show()
```



1.9 1.14

[41]:
```
dataset = fetch_openml(name='boston')
dataset.data
```

```
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:303:
UserWarning: Multiple active versions of the dataset matching the name boston
```

```
exist. Versions may be fundamentally different, returning version 1.
  warn(
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:1002:
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
```

[41]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	
..	...	...	...	...	...	...	...	...	..	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	

	PTRATIO	B	LSTAT
0	15.3	396.90	4.98
1	17.8	396.90	9.14
2	17.8	392.83	4.03
3	18.7	394.63	2.94
4	18.7	396.90	5.33
..	...	...	...
501	21.0	391.99	9.67
502	21.0	396.90	9.08
503	21.0	396.90	5.64
504	21.0	393.45	6.48
505	21.0	396.90	7.88

[506 rows x 13 columns]

[42]:
```
dataset.feature_names
```

[42]:
```
['CRIM',
 'ZN',
 'INDUS',
 'CHAS',
 'NOX',
 'RM',
 'AGE',
 'DIS',
```

```
 'RAD',
 'TAX',
 'PTRATIO',
 'B',
 'LSTAT']
```

[43]:
```
dataset.DESCR
```

[43]:
```
"**Author**:   \n**Source**: Unknown - Date unknown  \n**Please cite**:
\n\nThe Boston house-price data of Harrison, D. and Rubinfeld, D.L.
'Hedonic\nprices and the demand for clean air', J. Environ. Economics &
Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression
diagnostics\n...', Wiley, 1980.   N.B. Various transformations are used in the
table on\npages 244-261 of the latter.\nVariables in order:\nCRIM     per capita
crime rate by town\nZN       proportion of residential land zoned for lots over
25,000 sq.ft.\nINDUS    proportion of non-retail business acres per town\nCHAS
Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\nNOX
nitric oxides concentration (parts per 10 million)\nRM       average number of
rooms per dwelling\nAGE      proportion of owner-occupied units built prior to
1940\nDIS      weighted distances to five Boston employment centres\nRAD
index of accessibility to radial highways\nTAX      full-value property-tax rate
per $10,000\nPTRATIO  pupil-teacher ratio by town\nB        1000(Bk - 0.63)^2
where Bk is the proportion of blacks by town\nLSTAT    % lower status of the
population\nMEDV     Median value of owner-occupied homes in
$1000's\n\n\nInformation about the dataset\nCLASSTYPE: numeric\nCLASSINDEX:
last\n\nDownloaded from openml.org."
```

[44]:
```
dataset.target
```

[44]:
```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

[45]:
```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df.head()
```

[45]:

	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
0  396.90   4.98
1  396.90   9.14
2  392.83   4.03
3  394.63   2.94
4  396.90   5.33
```

[46]:
```
df['MEDV']=dataset.target
df.head()
```

[46]:
```
      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
```

[47]:
```
df.info()
```

```
<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
```

[48]:
```
print(df.isnull().sum())
```

```
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
```

[49]:
```
corr = df.corr()
corr
```

[49]:

	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
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

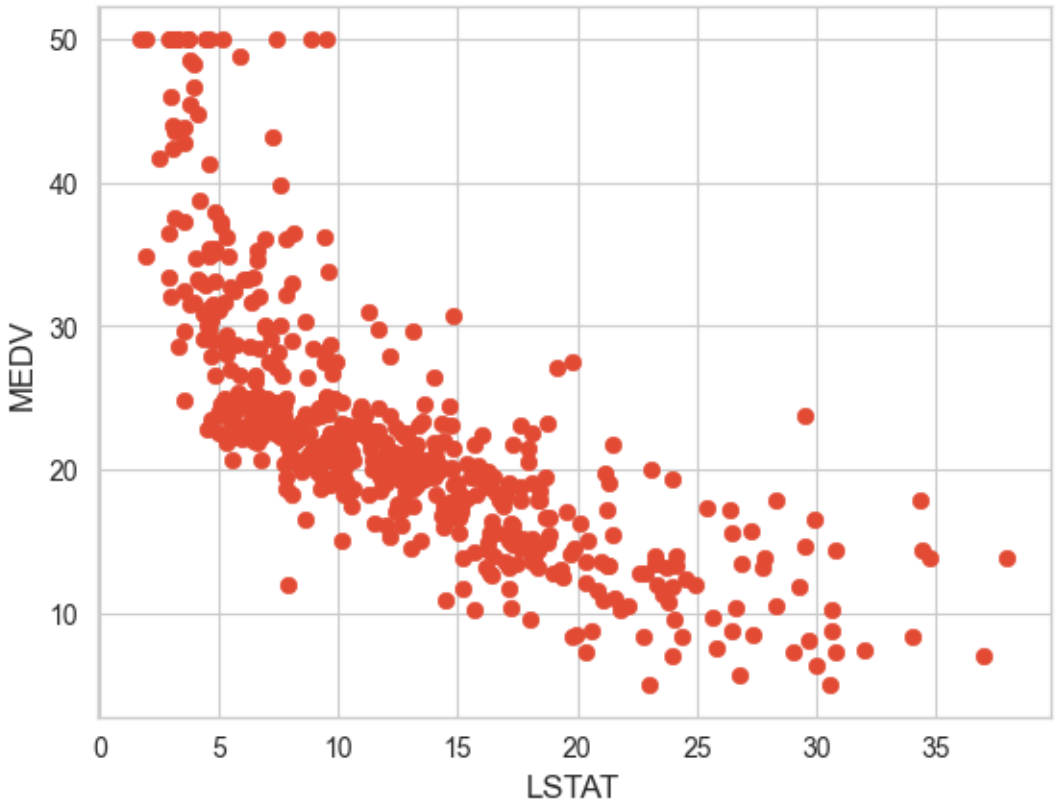
```
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -0.376955
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -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
```

[50]:
```
print(corr.abs().nlargest(3, 'MEDV').index)

print(corr.abs().nlargest(3, 'MEDV').values[:,13])
```

```
Index(['MEDV', 'LSTAT', 'RM'], dtype='object')
[1.         0.73766273 0.69535995]
```

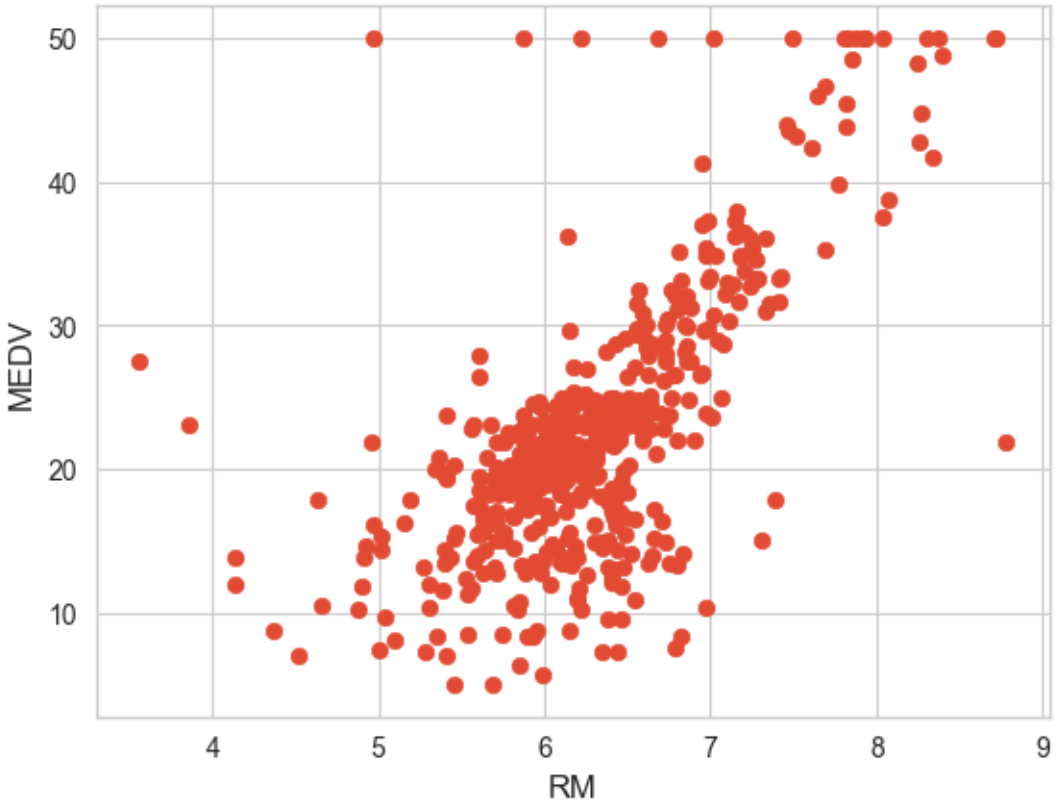
[51]:
```
plt.scatter(df['LSTAT'], df['MEDV'], marker='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
```

[51]:
```
Text(0, 0.5, 'MEDV')
```

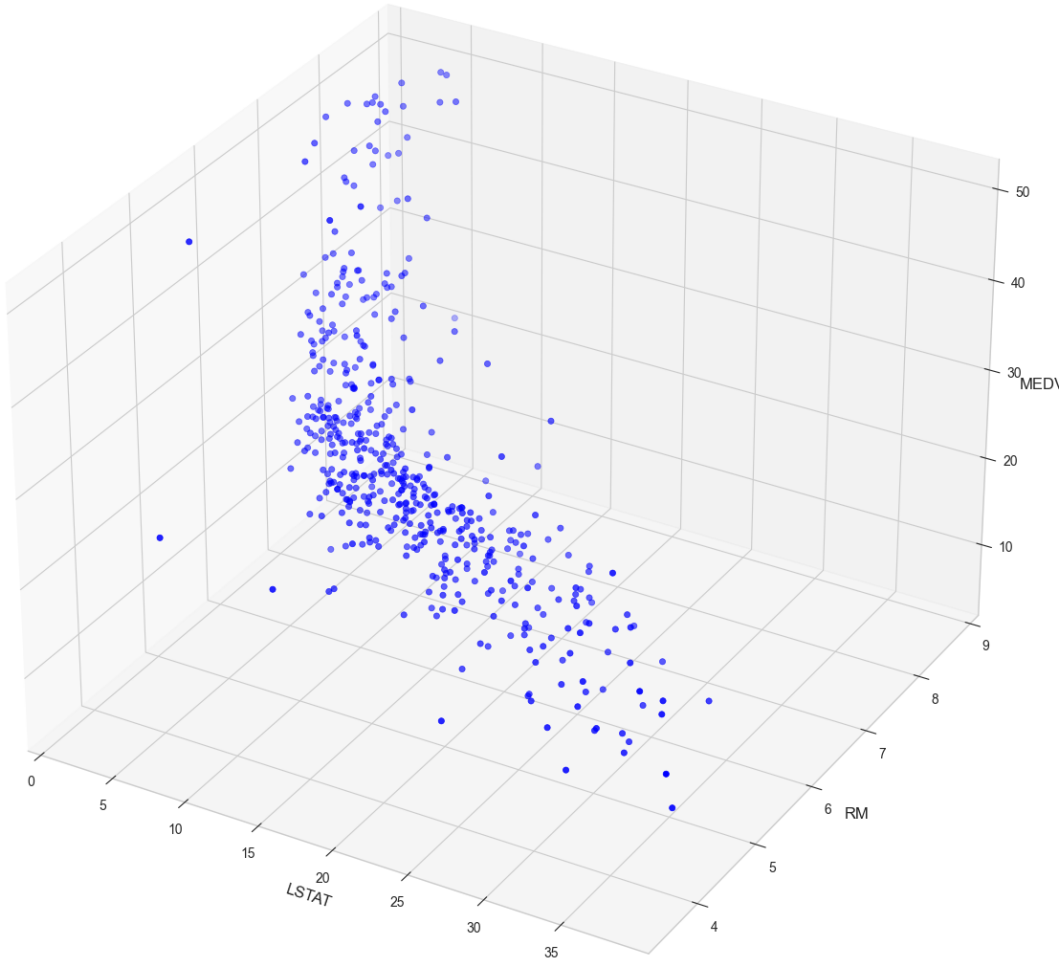


[52]:
```
plt.scatter(df['RM'], df['MEDV'], marker='o')
plt.xlabel('RM')
plt.ylabel('MEDV')
```

[52]:
```
Text(0, 0.5, 'MEDV')
```



[53]:
```
fig = plt.figure(figsize=(18,15))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['LSTAT'],
           df['RM'],
           df['MEDV'],
           c='b')

ax.set_xlabel("LSTAT")
ax.set_ylabel("RM")
ax.set_zlabel("MEDV")
plt.show()
```



[54]:
```
x = pd.DataFrame(np.c_[df['LSTAT'], df['RM']], columns = ['LSTAT','RM'])
Y = df['MEDV']
```

[55]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size = 0.3,␣
↪random_state=5)
```

[56]:
```
print(x_train.shape)
print(Y_train.shape)
```

```
(354, 2)
(354,)
```

[57]:
```
print(x_test.shape)
print(Y_test.shape)
```

```
(152, 2)
(152,)
```

[58]:
```
model = LinearRegression()
model.fit(x_train, Y_train)
price_prediction = model.predict(x_test)
```

[59]:
```
print('R-Squared: %.4f' % model.score(x_test,Y_test))
```

```
R-Squared: 0.6162
```

[60]:
```
mse = mean_squared_error(Y_test, price_prediction)
mse
```

[60]:
```
36.49422110915324
```

[61]:
```
plt.scatter(Y_test, price_prediction)
plt.xlabel("Actual price")
plt.ylabel("Predicted prices")
plt.title("Actual prices vs Predicted prices")
```

[61]:
```
Text(0.5, 1.0, 'Actual prices vs Predicted prices')
```

[62]:
```
print(model.intercept_)
print(model.coef_)
```

```
0.38437936780346504
[-0.65957972  4.83197581]
```

[63]:
```
print(model.predict([[30,5]]))
```

```
[4.75686695]
```

```
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

1.9.1 Plotting the 3D Hyperlane

[64]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
```

```
from sklearn.datasets import fetch_openml

dataset = fetch_openml(name='boston')

df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df['MEDV'] = dataset.target

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

fig = plt.figure(figsize=(18,15))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(x['LSTAT'],
           x['RM'],
           Y,
           c='b')

ax.set_xlabel("LSTAT")
ax.set_ylabel("RM")
ax.set_zlabel("MEDV")

#---create a meshgrid of all the values for LSTAT and RM---
x_surf = np.arange(0, 40, 1)   #---for LSTAT---
y_surf = np.arange(0, 10, 1)   #---for RM---
x_surf, y_surf = np.meshgrid(x_surf, y_surf)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, Y)

#---calculate z(MEDC) based on the model---
z = lambda x,y: (model.intercept_ + model.coef_[0] * x + model.coef_[1] * y)

ax.plot_surface(x_surf, y_surf, z(x_surf,y_surf),
                rstride=1,
                cstride=1,
                color='None',
                alpha = 0.4)

plt.show()
```

```
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:303:
UserWarning: Multiple active versions of the dataset matching the name boston
exist. Versions may be fundamentally different, returning version 1.
  warn(
c:\Users\tien2\miniconda3\lib\site-packages\sklearn\datasets\_openml.py:1002:
```

```
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
the Notes Section in fetch_openml's API doc for details.
  warn(
```