Họ và tên: Trần Tiến Nam

MSV: B20DCCN460

1.1

Trí tuệ nhân tạo (AI) là một lĩnh vực của khoa học máy tính tập trung vào việc tạo ra các hệ thống có khả năng suy nghĩ và hành động giống con người. AI đã có những tiến bộ đáng kể trong những năm gần đây và đang được sử dụng ngày càng rộng rãi trong nhiều lĩnh vực khác nhau, từ chăm sóc sức khỏe, tài chính, đến sản xuất và logistics.

Máy móc vẫn chưa tiếp quản. Ít nhất là chưa. Tuy nhiên, chúng đang xâm nhập vào cuộc sống của chúng ta, ảnh hưởng đến cách chúng ta sống, làm việc và giải trí. Từ trợ lý cá nhân được hỗ trợ bằng giọng nói như Siri và Alexa, đến các công nghệ cơ bản và cơ bản hơn như thuật toán hành vi, tìm kiếm gợi ý và xe tự lái chạy bằng năng lượng tự động có khả năng dự đoán mạnh mẽ, có một số ví dụ và ứng dụng về trí tuệ nhân tạo đang được sử dụng ngày nay.

Nhìn vào những hệ thống trợ giúp thông minh, chúng ta thấy rõ sự tiến bộ của trí tuệ nhân tạo. Những trợ lý cá nhân như Siri, Alexa không chỉ hiểu ngôn ngữ con người mà còn có khả năng đưa ra gợi ý hữu ích. Các thuật toán học máy dự đoán hành vi của chúng ta dựa trên dữ liệu lịch sử, từ đó tối ưu hóa trải nghiệm tìm kiếm. Ngay cả xe tự lái, sự kết hợp giữa trí tuệ nhân tạo và tự động hóa, đang từng bước thử nghiệm với khả năng tự đoán mạnh mẽ.

Tuy nhiên, cần nhấn mạnh rằng chúng ta vẫn đứng ở giai đoạn sơ khai của sự phát triển này. Trí tuệ nhân tạo còn nhiều hạn chế và vấn đề thách thức cần được giải quyết. Ví dụ, việc đảm bảo an toàn tuyệt đối cho các hệ thống xe tự lái vẫn còn là một thách thức lớn. Đồng thời, cảnh báo về việc trí tuệ nhân tạo có thể

thay thế nhiều công việc truyền thống cũng đang dấy lên những lo ngại về thất nghiệp.

Hứa hẹn của A.I.: Khả năng học tự nhiên của các hệ thống như Google's DeepMind. Điều này đặt nền tảng cho sự phát triển đáng kể trong tương lai khi A.I. có khả năng cải thiện và nâng cao khả năng của mình.

- Trí tuệ nhân tạo: Lợi ích và rủi ro

AI có tiềm năng mang lại nhiều lợi ích cho xã hội, bao gồm:

- Cải thiện hiệu quả: AI có thể được sử dụng để tự động hóa các nhiệm vụ và quy trình, giúp giảm chi phí và thời gian. Ví dụ, AI đang được sử dụng để tự động hóa các quy trình sản xuất, phân tích dữ liệu và cung cấp dịch vụ khách hàng.
- AI có thể được sử dụng để cải thiện hiệu quả trong nhiều lĩnh vực, chẳng hạn như chăm sóc sức khỏe, tài chính, sản xuất và logistics. Ví dụ, AI đang được sử dụng để phát triển các phương pháp điều trị mới, phân tích dữ liệu lớn và tự động hóa các quy trình sản xuất.
- Nâng cao chất lượng cuộc sống: AI có thể được sử dụng để cải thiện chất lượng cuộc sống của con người theo nhiều cách, chẳng hạn như phát triển các phương pháp điều trị mới, tạo ra các sản phẩm và dịch vụ mới và cá nhân hóa trải nghiệm của người dùng.
- Giải quyết các vấn đề lớn: AI có tiềm năng giải quyết một số vấn đề lớn nhất của thế giới, chẳng hạn như nghèo đói, bệnh tật và biến đổi khí hậu. Ví dụ, AI đang được sử dụng để phát triển các công nghệ mới giúp tăng năng suất nông nghiệp, phát hiện sớm bệnh tật và giảm thiểu tác động của biến đổi khí hậu.

Tuy nhiên, AI cũng có thể gây ra một số rủi ro, bao gồm:

- Mất việc làm: AI có thể tự động hóa nhiều công việc hiện đang được thực hiện bởi con người. Điều này có thể dẫn đến thất nghiệp và thay đổi cơ cấu việc làm.
- Phân biệt đối xử: AI có thể bị thiên vị bởi dữ liệu mà nó được đào tạo. Điều này có thể dẫn đến việc AI tạo ra kết quả phân biệt đối xử đối với các nhóm người khác nhau.
- Lạm dụng: AI có thể bị sử dụng cho các mục đích xấu, chẳng hạn như phát triển vũ khí tự động hoặc tấn công mạng.

Để giải quyết những rủi ro của AI, chúng ta cần:

- Tăng cường giáo dục và đào tạo về AI: Chúng ta cần giáo dục mọi người về AI để họ hiểu được tiềm năng và rủi ro của công nghệ này.
- Phát triển các tiêu chuẩn và quy định về AI: Chúng ta cần phát triển các tiêu chuẩn và quy định để đảm bảo rằng AI được phát triển và sử dụng một cách an toàn và có trách nhiệm.
- Tăng cường sự minh bạch và trách nhiệm giải trình: Chúng ta cần tăng cường sự minh bạch và trách nhiệm giải trình trong việc phát triển và sử dụng AI.

Trong tương lai, AI có thể thúc đẩy sự tiến bộ không tưởng, từ việc giải quyết các vấn đề phức tạp cho đến mở ra những khả năng mới trong y học, khoa học và nhiều lĩnh vực khác. Tuy nhiên, để đảm bảo rằng chúng ta hướng đến một tương lai tốt đẹp hơn với AI, chúng ta cần đối mặt với những thách thức này một cách trách nhiệm và tôn trọng đạo đức, đồng thời luôn đặt con người vào tâm điểm của mọi quyết định và hành động.

1.2

Định nghĩa về Hệ thống thông minh tạo ấn tượng mạnh nhất với tôi:

Hệ thống thông minh là một hệ thống có thể học hỏi, suy luận và hành động một cách tự chủ để đạt được các mục tiêu của nó. Hệ thống thông minh khác với các hệ thống truyền thống ở chỗ chúng có thể học hỏi và thích ứng với các tình huống mới. Điều này cho phép chúng giải quyết các vấn đề phức tạp một cách hiệu quả hơn.

Trong cuộc cách mạng công nghiệp 4.0 đầy biến đổi và tiến bộ, khái niệm về hệ thống thông minh đang từng bước cất cánh, mang theo vô vàn triển vọng và tiềm năng đối với sự tiến bộ của con người. Hệ thống thông minh không chỉ đơn thuần là những công cụ hỗ trợ, mà còn là những "người bạn đồng hành" có khả năng học hỏi, suy luận và hành động tự chủ để đạt được các mục tiêu đề ra. Với khả năng thích nghi và học từ những tình huống mới, hệ thống thông minh đang dẫn dắt chúng ta vào tương lai đầy tiềm năng và thách thức.

- Tính Chất Cơ Bản của Hệ Thống Thông Minh

Hệ thống thông minh khác biệt so với các hệ thống truyền thống bởi khả năng học hỏi và thích nghi của chúng. Ba đặc điểm chính của hệ thống thông minh gồm:

1. Khả năng Học Hỏi: Hệ thống thông minh có khả năng học hỏi từ dữ liệu và kinh nghiệm để cải thiện hiệu suất của mình. Học máy, học sâu và học tăng cường là những kỹ thuật quan trọng để hệ thống thông minh tiếp tục phát triển và hoàn thiện.
2. Khả năng Suy Luận:** Hệ thống thông minh có khả năng sử dụng kiến thức của mình để giải quyết vấn đề và đưa ra quyết định. Lý luận luận cứ, lý luận luận lý và lý luận Bayesian là những công cụ hỗ trợ hệ thống thông minh trong việc suy luận và đưa ra quyết định.
3. Khả năng Hành Động:** Hệ thống thông minh không chỉ biết cách suy nghĩ, mà còn biết cách hành động để đạt được mục tiêu. Điều khiển, lập

kế hoạch và điều khiển tự động là những cách mà hệ thống thông minh thể hiện khả năng hành động tự chủ.

- Phân Loại Các Loại Hệ Thống Thông Minh

Thế giới của hệ thống thông minh đa dạng và phong phú với nhiều loại khác nhau, từ hệ thống học máy đến hệ thống trí tuệ nhân tạo và hệ thống tự động hóa. Mỗi loại hệ thống mang đến những ứng dụng và tiềm năng riêng.

1. Hệ Thống Học Máy: Các hệ thống học máy có khả năng học từ dữ liệu để cải thiện hiệu suất. Chúng có thể phân loại hình ảnh, dịch ngôn ngữ và phát hiện gian lận, giúp tối ưu hóa nhiều quy trình trong cuộc sống hàng ngày.
2. Hệ Thống Trí Tuệ Nhân Tạo:** Hệ thống trí tuệ nhân tạo có khả năng suy luận và hành động tự chủ. Chẳng hạn như hệ thống lái xe tự động, chơi cờ, và chăm sóc người cao tuổi, chúng mang lại sự tiện ích và sự tự động hóa đáng kể.
3. Hệ Thống Tự Động Hóa:** Các hệ thống tự động hóa có khả năng thực hiện nhiệm vụ mà không cần sự tham gia của con người. Chúng được ứng dụng trong quản lý kho bãi, sản xuất và các trung tâm dữ liệu để tối ưu hóa hiệu suất và độ chính xác.

- Ứng Dụng Đa Dạng trong Cuộc Sống

Hệ thống thông minh đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Chăm sóc sức khỏe, quản lý logistics, tài chính, giáo dục và giải trí đều trở thành nơi thể hiện tiềm năng của hệ thống thông minh. Chúng giúp cải thiện dịch vụ, tối ưu hóa quy trình và mang lại lợi ích rõ rệt cho con người.

- Hệ Thống Thông Minh và Tương Lai

Sự phát triển không ngừng của hệ thống thông minh mang lại hy vọng cho một tương lai tươi sáng và bền vững. Hệ thống thông minh có thể đóng góp vào việc

giảm nghèo, chống lại bệnh tật và giải quyết biến đổi khí hậu. Tuy nhiên, cũng cần nhớ rằng hệ thống thông minh cũng đi kèm với những rủi ro, chẳng hạn như tạo ra sự bất bình đẳng và đe dọa quyền riêng tư.

- Kết Luận

Hệ thống thông minh là một hệ quả không thể tránh trong sự tiến bộ của xã hội và khoa học kỹ thuật. Khả năng học hỏi, suy luận và hành động tự chủ của chúng mở ra một tương lai đầy triển vọng và cơ hội. Để đảm bảo rằng sự phát triển của hệ thống thông minh mang lại lợi ích cho tất cả mọi người, chúng ta cần thực hiện và sử dụng công nghệ này một cách có trách nhiệm và tạo ra các chương trình kiểm soát thích hợp để đảm bảo tính bảo mật và công bằng.

1.3

Công nghệ trí tuệ nhân tạo (AI) đang ngày càng trở nên phổ biến và có tác động sâu sắc đến nhiều ngành công nghiệp khác nhau. Một số ví dụ về ứng dụng AI trong các ngành công nghiệp chính, bao gồm sản xuất, tài chính, chăm sóc sức khỏe, bán lẻ và vận tải. Những ứng dụng này không chỉ mang lại lợi ích cho các doanh nghiệp mà còn tạo ra những trải nghiệm tốt hơn cho người tiêu dùng. Tuy nhiên, cũng cần phải đối mặt với những thách thức và rủi ro của công nghệ AI.

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.

- Công nghiệp sản xuất:

Trong ngành sản xuất, AI đang được sử dụng để tự động hóa các quy trình sản xuất, tối ưu hóa hiệu quả sản xuất và cải thiện chất lượng sản phẩm. Các hệ thống thông minh có khả năng điều khiển máy móc, theo dõi quy trình sản xuất và đưa ra các cải tiến thiết kế dựa trên dữ liệu. Ví dụ, AI có thể phân tích dữ liệu từ các cảm biến để điều khiển máy móc và máy robot trong quy trình sản xuất. Nó cũng có thể kiểm tra chất lượng sản phẩm bằng cách so sánh dữ liệu từ sản phẩm thực tế với các tiêu chuẩn chất lượng.

- Công nghiệp tài chính:

Trong lĩnh vực tài chính, AI đang có sự ảnh hưởng lớn đến cách giao dịch và quản lý tài sản. Các hệ thống thông minh được sử dụng để phân tích dữ liệu lớn và dự đoán xu hướng thị trường. Ví dụ, các thuật toán AI có khả năng xác định các mô hình phức tạp trong dữ liệu tài chính và dự đoán giá cổ phiếu hoặc biến động thị trường. AI cũng được sử dụng để phát hiện các hoạt động gian lận trong giao dịch tài chính, giúp bảo vệ doanh nghiệp và người tiêu dùng khỏi các mất mát về tài chính.

- Công nghiệp chăm sóc sức khỏe:

Sự phát triển của AI đang định hình lại cách chúng ta chăm sóc sức khỏe. Các hệ thống thông minh được sử dụng để chẩn đoán bệnh, phát triển các phương pháp điều trị mới và nâng cao chất lượng dịch vụ chăm sóc sức khỏe. Trong việc chẩn đoán bệnh, AI có thể phân tích dữ liệu y tế và hình ảnh y khoa để đưa ra các kết quả chẩn đoán chính xác và sớm hơn. Các hệ thống thông minh cũng có thể cung cấp dịch vụ chăm sóc từ xa, theo dõi tình trạng sức khỏe của bệnh nhân và cung cấp hỗ trợ y tế từ xa.

- Công nghiệp bán lẻ:

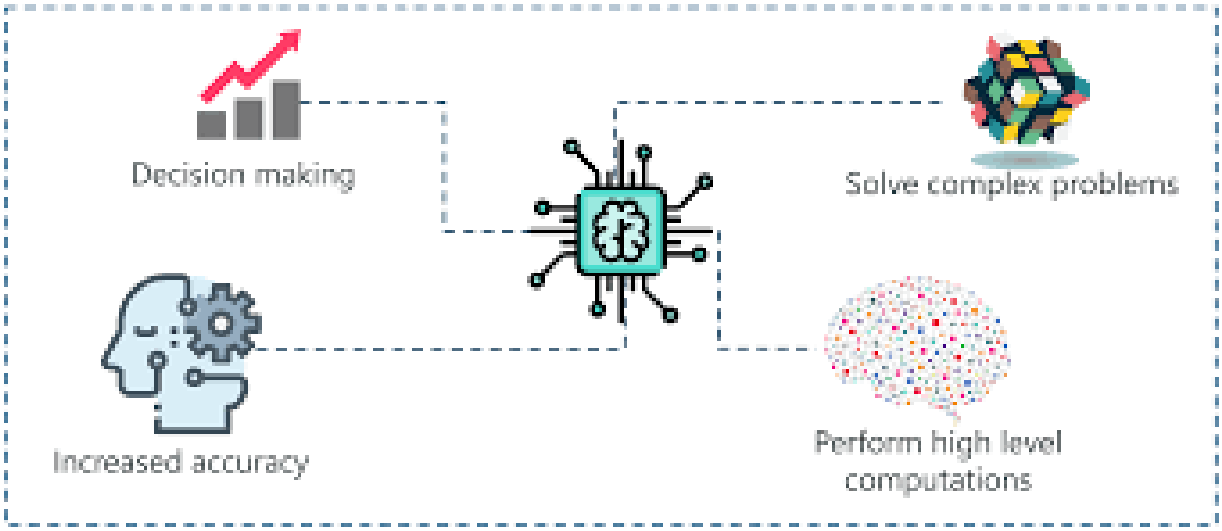
AI cũng đang thay đổi cách mua sắm và quản lý bán lẻ. Trong lĩnh vực bán lẻ, hệ thống thông minh được sử dụng để cá nhân hóa trải nghiệm mua sắm, tối ưu hóa chiến dịch tiếp thị và quản lý chuỗi cung ứng. Ví dụ, AI có khả năng theo dõi hành vi mua sắm của khách hàng trên các nền tảng trực tuyến và đề xuất sản phẩm tương thích với sở thích của họ. Các hệ thống này cũng có thể dự đoán nhu cầu sản phẩm và tối ưu hóa quản lý hàng tồn kho.

- Công nghiệp vận tải:

Trong lĩnh vực vận tải, AI đang giúp cải thiện an toàn và hiệu quả của các phương tiện giao thông. Các hệ thống tự động hóa và xe tự lái được phát triển nhằm giảm nguy cơ tai nạn và tối ưu hóa việc di chuyển. AI cũng được sử dụng để quản lý giao thông đô thị, dự đoán tình trạng giao thông và tối ưu hóa các tuyến đường. Ví dụ, hệ thống thông minh có thể theo dõi dữ liệu lưu lượng giao thông và đề xuất tuyến đường tốt nhất để giảm ùn tắc.

1.4

Trên hành trình không ngừng phát triển của trí tuệ nhân tạo (AI), việc phân loại các loại hệ thống thông minh đóng một vai trò quan trọng để hiểu rõ hơn về sự phức tạp và đa dạng của công nghệ này. Một trong những cách phân loại phổ biến nhất là dựa trên mức độ tự chủ của hệ thống, được chia thành ba loại chính: hệ thống thông minh thụ động, hệ thống thông minh chủ động và hệ thống thông minh tự học.



1. Hệ thống thông minh thụ động:

Hệ thống thông minh thụ động là những hệ thống không có khả năng hành động độc lập. Chúng chỉ có thể phản ứng với các tác nhân kích thích từ môi trường. Ví dụ, một hệ thống phát hiện chuyển động là một hệ thống thông minh thụ động. Nhiệm vụ của hệ thống này là theo dõi môi trường và phát hiện chuyển động, sau đó gửi tín hiệu cảnh báo khi có sự thay đổi.

2. Hệ thống thông minh chủ động:

Hệ thống thông minh chủ động là những hệ thống có khả năng hành động độc lập mà không cần sự can thiệp của con người. Chúng có khả năng tự động ra quyết định và thực hiện hành động dựa trên thông tin từ môi trường. Một ví dụ điển hình là robot tự lái. Robot này có khả năng tự động điều hướng, tránh vật cản và thậm chí tìm kiếm đường đến đích mà không cần sự điều khiển từ con người.

3. Hệ thống thông minh tự học:

Hệ thống thông minh tự học là loại cao cấp nhất, có khả năng học hỏi và thích nghi với môi trường. Chúng có khả năng tự động thu thập dữ liệu, phân tích dữ

liệu và từ đó đề xuất ra các quy tắc mới để cải thiện hiệu suất. Một ví dụ điển hình là hệ thống nhận dạng khuôn mặt. Ban đầu, hệ thống có thể nhận dạng một số khuôn mặt cố định. Tuy nhiên, khi tiếp tục nhận diện và so sánh dữ liệu mới, hệ thống có thể tự học cách nhận dạng các khuôn mặt mới với độ chính xác cao hơn theo thời gian.

Ngoài cách phân loại theo mức độ tự chủ, hệ thống thông minh còn có thể được phân loại theo chức năng, mức độ phức tạp và lĩnh vực ứng dụng.

- Theo chức năng:

Hệ thống thông minh nhận thức: Các hệ thống này có khả năng hiểu và tương tác với thông tin từ môi trường. Ví dụ, hệ thống nhận dạng tiếng nói có thể hiểu và phản hồi theo yêu cầu của người dùng.

Hệ thống thông minh lập kế hoạch: Các hệ thống này có khả năng lập kế hoạch và quản lý tác vụ. Ví dụ, hệ thống quản lý lịch là một loại hệ thống thông minh lập kế hoạch.

Hệ thống thông minh ra quyết định: Các hệ thống này có khả năng ra quyết định dựa trên dữ liệu và thông tin. Ví dụ, hệ thống tư vấn đầu tư có thể đưa ra các lời khuyên về việc đầu tư dựa trên phân tích dữ liệu tài chính.

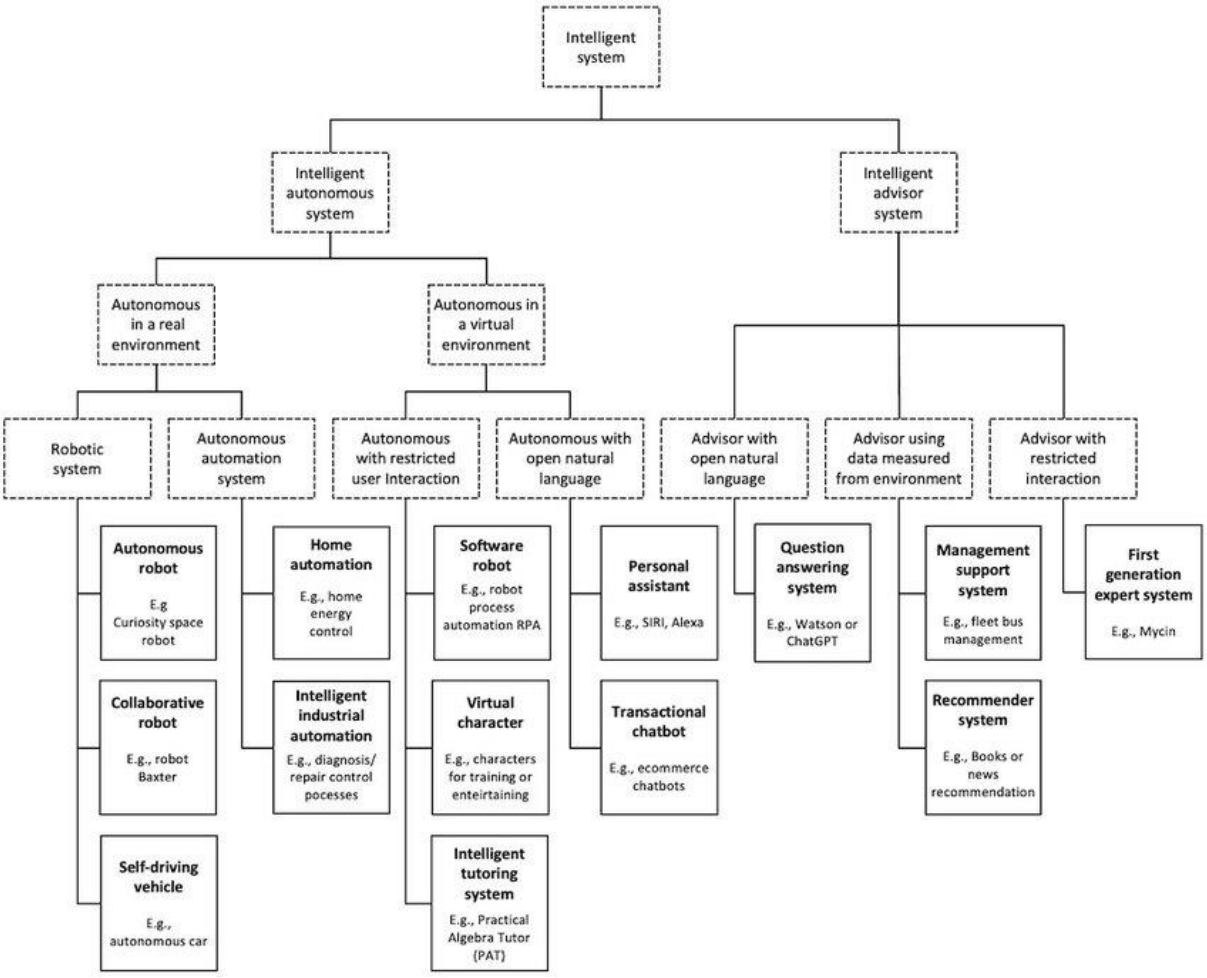
- Theo mức độ phức tạp:

Hệ thống thông minh cũng có thể được phân loại thành đơn giản và phức tạp, dựa vào mức độ khả năng thích ứng và học hỏi của chúng.

- Theo lĩnh vực ứng dụng:

Hệ thống thông minh có thể được phân loại dựa vào lĩnh vực ứng dụng như sản xuất, dịch vụ, y tế, tài chính và nhiều lĩnh vực khác.

1.5



Hình 7. Examples of intelligent systems

Hình 7 minh họa sự đa dạng của các loại hệ thống thông minh, được tổ chức theo một cấu trúc phân cấp (mặc dù cấu trúc này không nhằm mục đích là chính xác và toàn diện mà chỉ mang tính minh họa cho các loại hệ thống tồn tại). Các phần tiếp theo sẽ đi vào chi tiết hơn về một số trong những danh mục này.

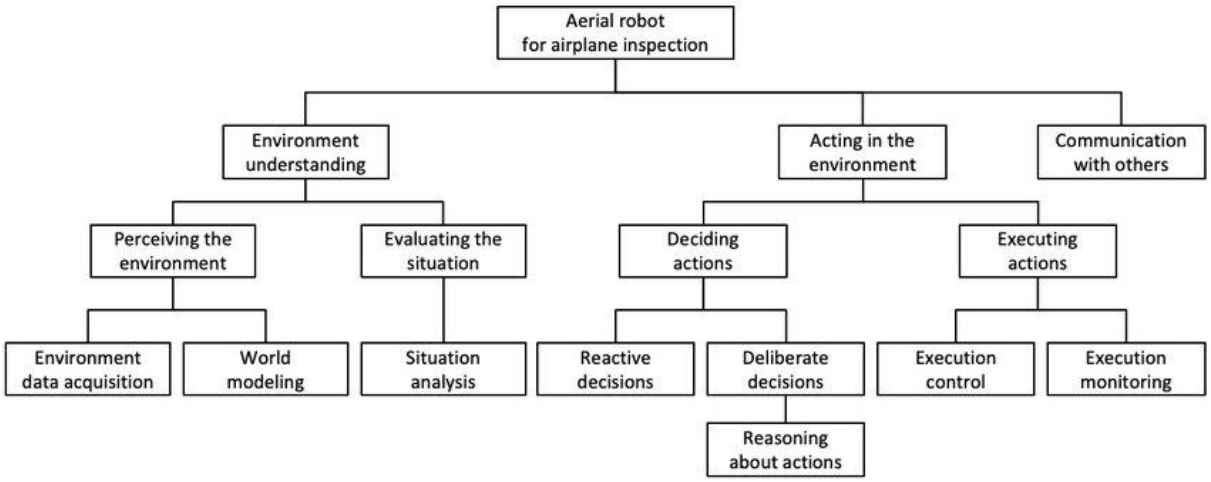
Một trong những ví dụ tiêu biểu về hệ thống thông minh là "robot tự động" - một loại hệ thống thông minh có thể bao gồm phần lớn các chức năng được xem xét trong tài liệu. Trong danh mục này, có các hệ thống có độ phức tạp khác nhau như robot hợp tác công nghiệp (ví dụ như Baxter), xe tự lái (như xe tự lái,

tàu thám hiểm sao Hỏa Curiosity), robot hỗ trợ, robot gia đình (như robot dọn nhà Roomba), robot giải trí (ví dụ như Sony Aibo, Furby), và nhiều loại khác.

Một ví dụ cụ thể về loại hệ thống này là robot bay dùng cho việc kiểm tra máy bay, phát triển dưới dạng một nguyên mẫu trình diễn vào năm 2019 cho công ty Airbus (Bavle 2019). Mục tiêu của robot này là giúp các kỹ sư bảo dưỡng kiểm tra bề mặt của máy bay để xác định các khuyết điểm có thể có. Robot có khả năng bay gần bề mặt của máy bay và xử lý hình ảnh được thu thập bằng cách sử dụng các máy ảnh được gắn trên máy bay. Hình 8 trình bày tổ chức chức năng tương ứng với hệ thống này, theo cách tiếp cận được trình bày trong tài liệu, bao gồm các chức năng cụ thể sau:

- Thu thập dữ liệu môi trường: Robot bay được trang bị nhiều cảm biến như (1) đơn vị đo iner (IMU) để cung cấp thông tin về hướng, vận tốc góc và gia tốc tuyến tính, (2) máy ảnh để tính toán vận tốc bằng cách sử dụng quỹ đạo thị giác, (3) cảm biến lidar tạo ra một điểm đám mây 3D, và (4) máy ảnh phía trước có độ phân giải cao.
- Mô hình thế giới: Hệ thống đại diện cho thế giới bằng một không gian 3D và xác định vị trí của mình thông qua việc kết hợp dữ liệu từ nhiều cảm biến bằng cách sử dụng bộ lọc Kalman mở rộng (EKF).
- Phân tích tình huống: Máy ảnh có độ phân giải cao được sử dụng để nhận biết các khuyết điểm trên bề mặt của máy bay bằng cách sử dụng thị giác máy tính, các khuyết điểm này được xác định không gian theo vị trí của robot bay.
- Quyết định phản xạ: Hệ thống bao gồm một cơ chế phản xạ ngăn chặn xe để tránh các chướng ngại vật bất ngờ được phát hiện bằng cách sử dụng cảm biến lidar. Nền tảng bay cũng bao gồm một cơ chế an toàn để hạ cánh ngay lập tức khi lượng năng lượng còn lại trong pin quá thấp.
- Ra quyết định về hành động: Hệ thống tạo ra một kế hoạch kiểm tra (bằng cách lập kế hoạch đường đi) để bao phủ bề mặt của máy bay.

- Kiểm soát thực hiện: Hệ thống sử dụng cơ chế kiểm soát chuyển động để thực hiện kế hoạch điều hướng. Nó sử dụng bộ điều khiển chuyển động với thuật toán theo dõi đường đi và kiểm soát PID.
- Giám sát thực hiện: Kiểm soát chuyển động được giám sát để phát hiện khi các mục tiêu phần được hoàn thành. Hệ thống cũng bao gồm một số cơ chế để phát hiện việc thực hiện đúng (mặc dù phần này chưa được phát triển hoàn chỉnh vì robot chỉ là một nguyên mẫu trình diễn).
- Giao tiếp với người khác: Người điều hành cung cấp hình học của máy bay với các vị trí liên quan để kiểm tra. Người điều hành có thể dừng việc thực hiện nhiệm vụ nếu có tình huống khẩn cấp.
- Trong tóm tắt trên, chúng ta đã xem xét một ví dụ về hệ thống thông minh tự động - robot bay kiểm tra máy bay. Từ việc thu thập dữ liệu môi trường đến việc ra quyết định phản xạ và giao tiếp với con người, hệ thống này thể hiện sự tích hợp của nhiều chức năng thông minh để đạt được mục tiêu kiểm tra và duyệt qua bề mặt máy bay.



1.6

1. Thư viện NumPy:
   - Đặc trưng:

NumPy là một thư viện quan trọng trong Python dành cho tính toán khoa học và toán học trên mảng đa chiều. NumPy cung cấp các cấu trúc dữ liệu mảng mạnh mẽ, cho phép thực hiện các phép toán số học và logic trên mảng một cách hiệu quả.

- Mục đích:

Xử lý dữ liệu mảng đa chiều: NumPy cho phép tạo, xử lý và thực hiện các phép toán trên các mảng đa chiều nhanh chóng.

Tính toán khoa học: NumPy cung cấp các hàm toán học, thống kê và đại số tuyến tính để thực hiện tính toán phức tạp trên dữ liệu.

Tích hợp với các thư viện khác: NumPy là nền tảng cho nhiều thư viện khoa học dữ liệu khác như pandas, scikit-learn và matplotlib.

- Ví dụ:

```
import numpy as np          // Import thư viện

arr = np.array([1, 2, 3, 4, 5])      // Tạo mảng gồm 5 phần tử

sum_arr = np.sum(arr)               // Tính tổng các phần tử trong mảng

matrix = np.array([[1, 2, 3], [4, 5, 6]])          // Tạo mảng 2 chiều ( ma trận )

matrix.T = np.array([[4, 5, 6], [1, 2, 3]])

matrix_product = np.dot(matrix, matrix.T)              // Tính tích hai ma trận
```

2. Thư viện Pandas:

- Đặc trưng:

Pandas là một thư viện mạnh mẽ dành cho xử lý và phân tích dữ liệu dạng bảng. Nó cung cấp các cấu trúc dữ liệu như DataFrame cho phép làm việc với dữ liệu có cấu trúc và không có cấu trúc một cách dễ dàng.

- Mục đích:

Xử lý và phân tích dữ liệu bảng: Pandas cho phép đọc, ghi, lọc, xử lý và thực hiện các phép tính trên dữ liệu bảng.

Chuẩn hóa và làm sạch dữ liệu: Pandas cung cấp các công cụ để xử lý dữ liệu thiếu, trùng lặp và không chính xác.

Thống kê và trực quan hóa: Pandas giúp thực hiện các phân tích thống kê cơ bản và tạo biểu đồ từ dữ liệu.

- Ví dụ:

```
import pandas as pd          // Import thư viện

# Tạo DataFrame từ dictionary

data = {'Name': ['Alice', 'Bob', 'Charlie'],

        'Age': [25, 30, 22]}

df = pd.DataFrame(data)

# Đọc dữ liệu từ file CSV

csv_data = pd.read_csv('data.csv')
```

```
# Lọc dữ liệu theo điều kiện

filtered_data = df[df['Age'] > 25]

# Tính trung bình tuổi

average_age = df['Age'].mean()

# Vẽ biểu đồ cột

df.plot(kind='bar', x='Name', y='Age')
```

3. Thư viện Matplotlib:
   - Đặc trưng:

Matplotlib là một thư viện trực quan hóa mạnh mẽ trong Python, cho phép tạo ra nhiều loại biểu đồ và đồ thị chất lượng cao.

- Mục đích:

Trực quan hóa dữ liệu: Matplotlib cho phép tạo biểu đồ cột, biểu đồ đường, biểu đồ phân tán và nhiều loại biểu đồ khác để trực quan hóa dữ liệu.

Hiển thị thông tin: Matplotlib giúp hiển thị thông tin một cách trực quan và dễ hiểu thông qua các biểu đồ và đồ thị.

Tùy chỉnh giao diện: Matplotlib cho phép tùy chỉnh các yếu tố trong biểu đồ như màu sắc, kích thước, tiêu đề, chú thích, vị trí trục, v.v.

- Ví dụ:

```
import matplotlib.pyplot as plt

# Tạo biểu đồ đường

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

y = [10, 15, 7, 12, 9]

plt.plot(x, y)

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Line Chart')

plt.show()

# Tạo biểu đồ cột

categories = ['A', 'B', 'C']

values = [25, 40, 30]

plt.bar(categories, values)

plt.xlabel('Categories')

plt.ylabel('Values')

plt.title('Bar Chart')

plt.show()
```

4. Thư viện scikit-learn:

- Đặc trưng:

Scikit-learn là một thư viện học máy phổ biến trong Python, cung cấp các công cụ cho việc xây dựng mô hình học máy và khai thác dữ liệu.

- Mục đích:

Xây dựng mô hình học máy: Scikit-learn cung cấp các thuật toán học máy phổ biến như hồi quy, phân loại, gom cụm, và giảm chiều dữ liệu.

Đánh giá mô hình: Scikit-learn cho phép đánh giá hiệu suất của các mô hình học máy thông qua các phép đo như chính xác, độ chính xác cân bằng và F1-score.

Tiền xử lý dữ liệu: Scikit-learn hỗ trợ tiền xử lý dữ liệu như chuẩn hóa, mã hóa và xử lý dữ liệu thiếu.

- Ví dụ:

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

# Tải dữ liệu iris

iris = load_iris()

X = iris.data
```

```
y = iris.target

# Chia dữ liệu thành tập huấn luyện và kiểm tra

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

# Xây dựng mô hình Logistic Regression

model = LogisticRegression()

model.fit(X_train, y_train)

# Dự đoán và đánh giá mô hình

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print('Accuracy:', accuracy)
```

1-7

August 27, 2023

[6]:
```
import numpy as np
import pandas as pd
```

[2]:
```
names = np.array(['Ann', 'Joe', 'Mark'])
heights = np.array([1.5, 1.78, 1.6])
weights = np.array([65, 46, 59])
```

[3]:
```
bmi = weights / heights ** 2
```

[4]:
```
bmi
```

[4]:
```
array([28.88888889, 14.51836889, 23.046875  ])
```

[7]:
```
def check_bmi(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif bmi < 25:
        return 'Normal weight'
    elif bmi < 30:
        return 'Overweight'
```

[8]:
```
classify = np.vectorize(check_bmi)
classify(bmi)
```

[8]:
```
array(['Overweight', 'Underweight', 'Normal weight'], dtype='<U13')
```

[9]:
```
df = pd.DataFrame({
  'Name': names,
  'Height': heights,
  'Weight': weights,
  'BMI': bmi,
  'Classify': classify(bmi)})
```

[10]:
```
df
```

[10]:

	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
```

1-8

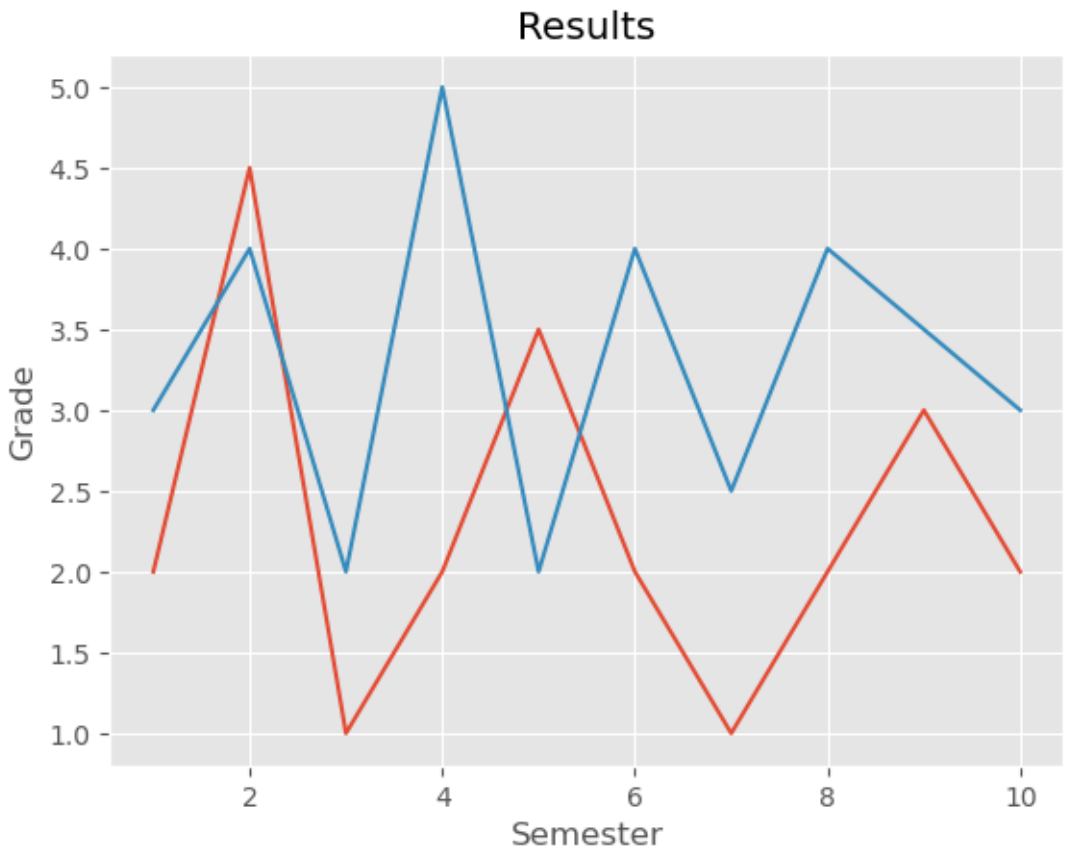
August 27, 2023

1 Plotting multiple Lines in the Same Chart

[7]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import pandas as pd
import random
```

[2]:
```
style.use('ggplot')

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')
```

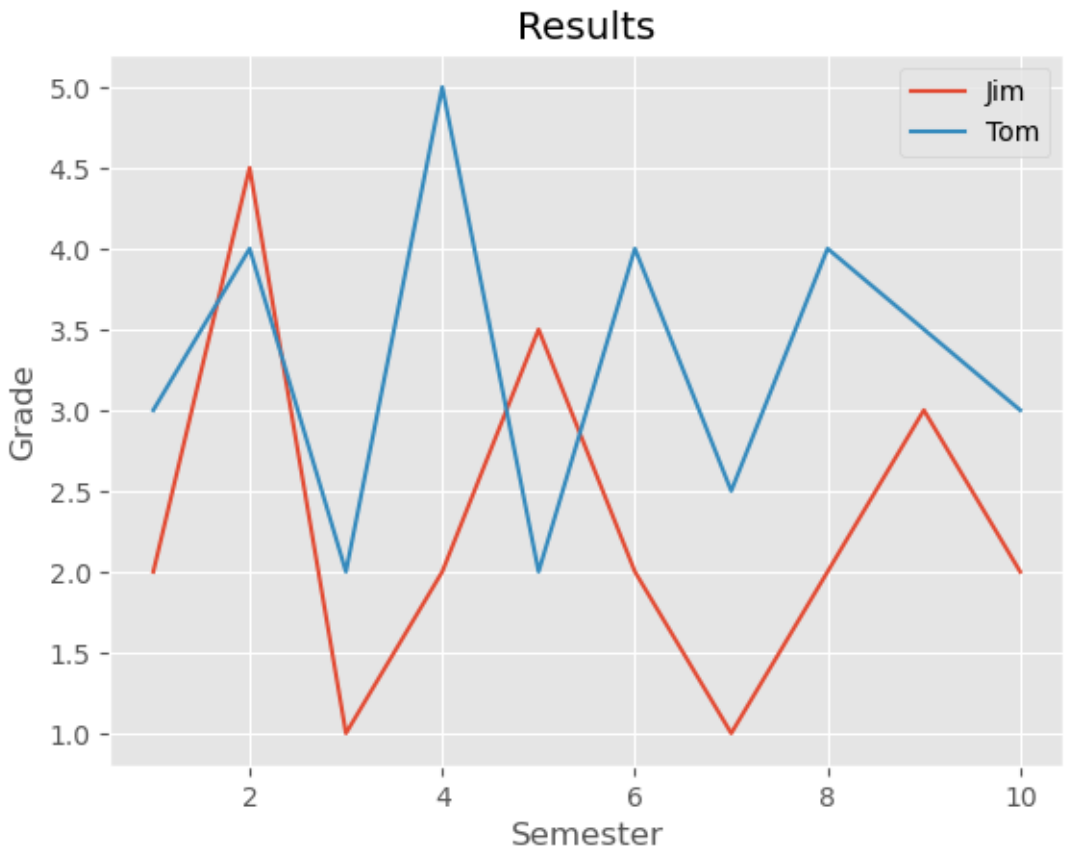
[2]:
```
Text(0, 0.5, 'Grade')
```



2 Adding a Legend

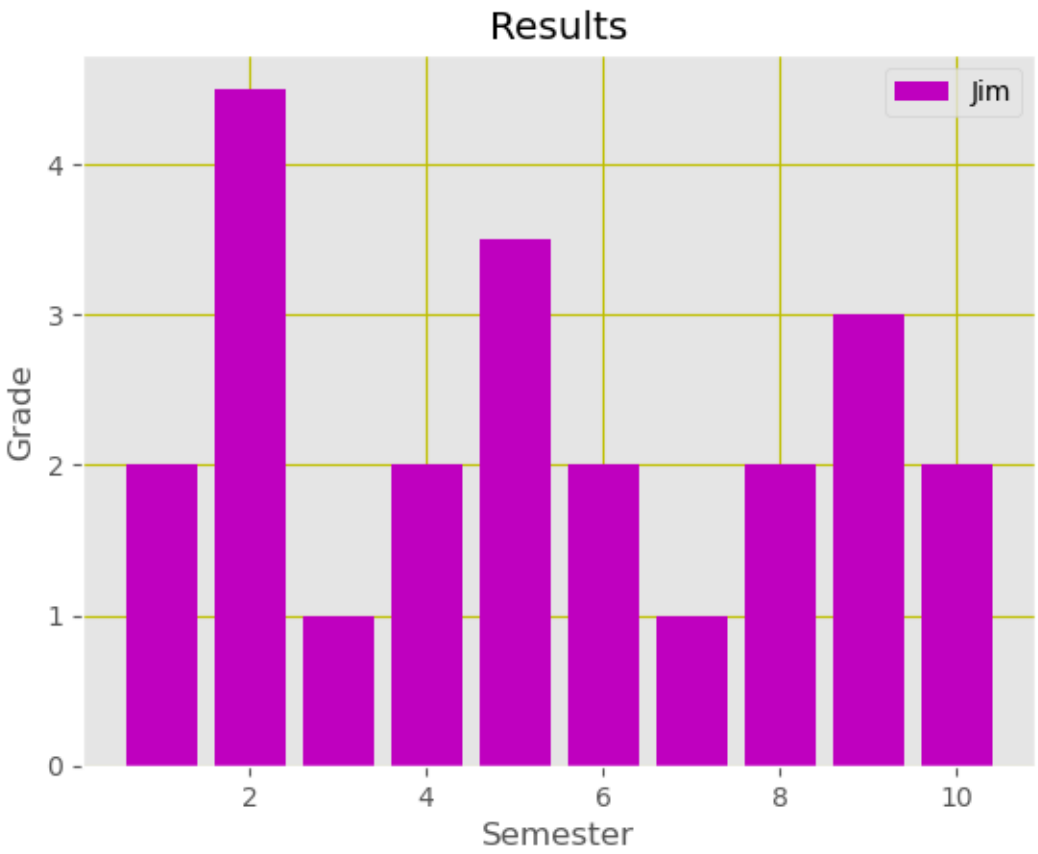
[4]:
```
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [2,4.5,1,2,3.5,2,1,2,3,2],
  label="Jim"
)
plt.plot(
  [1,2,3,4,5,6,7,8,9,10],
  [3,4,2,5,2,4,2.5,4,3.5,3],
  label="Tom"
)

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

[4]:
```
<matplotlib.legend.Legend at 0x28398e43e90>
```



3 Plotting bar charts

[6]:
```
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")
```



[11]:
```
df = pd.read_csv('./data/1_8.csv')
```

[13]:
```
df
```

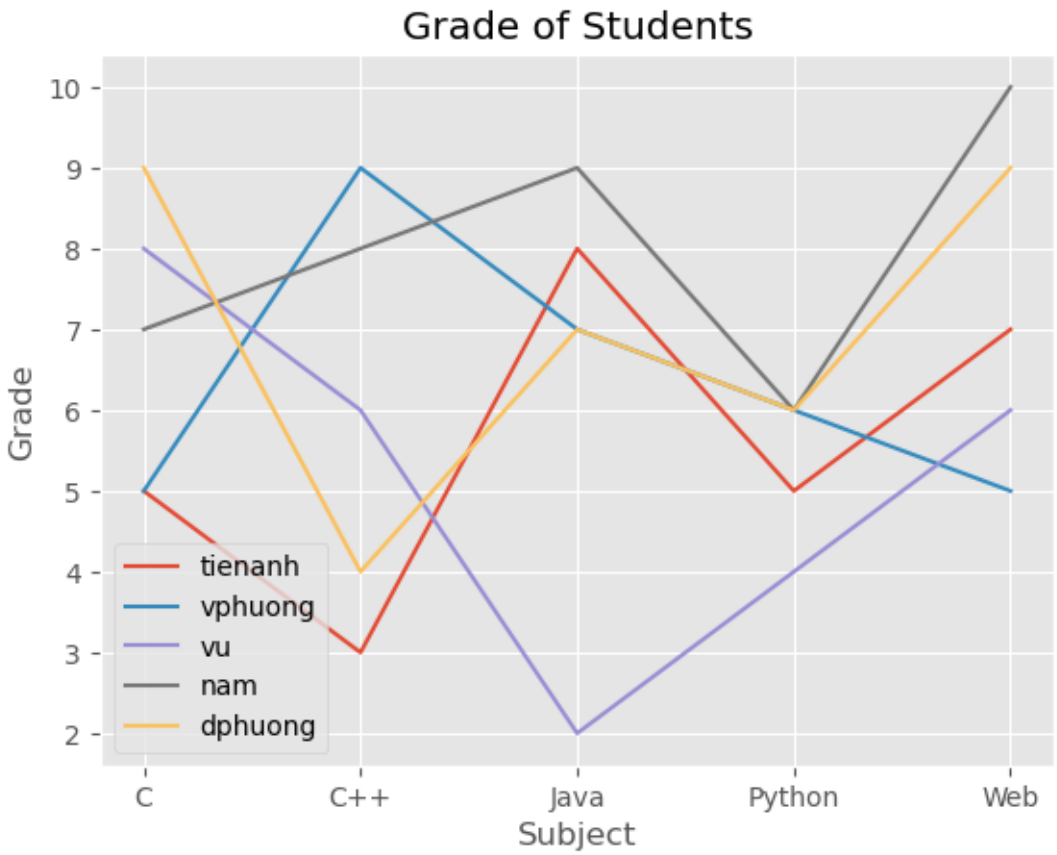
[13]:

	Subject	tienanh	vphuong	vu	nam	dphuong
0	C	5	5	8	7	9
1	C++	3	9	6	8	4
2	Java	8	7	2	9	7
3	Python	5	6	4	6	6
4	Web	7	5	6	10	9

[15]:
```
subjects = df["Subject"]
tienanh_grade = df["tienanh"]
vphuong_grade = df["vphuong"]
vu_grade = df["vu"]
nam_grade = df["nam"]
dphuong_grade = df["dphuong"]
```

[17]:
```
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()
```

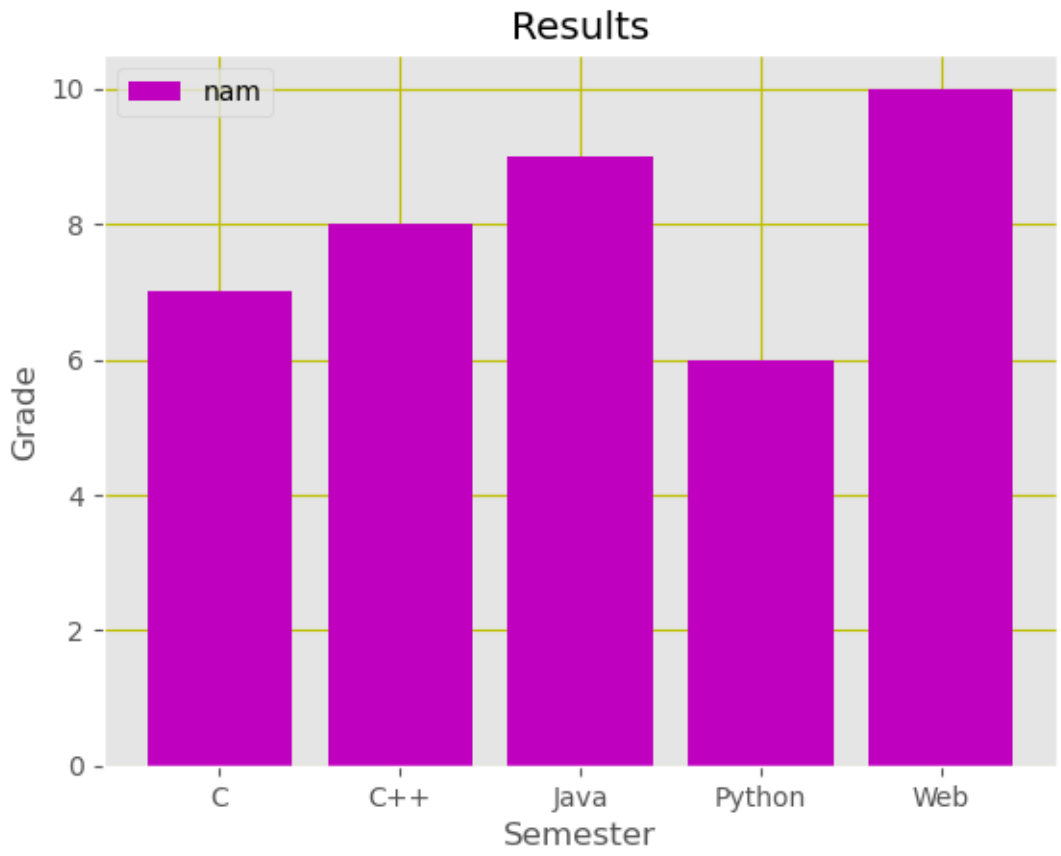
[17]:
```
<matplotlib.legend.Legend at 0x2839e1a7790>
```



[18]:
```
subjects = subjects.tolist()
subjects
```

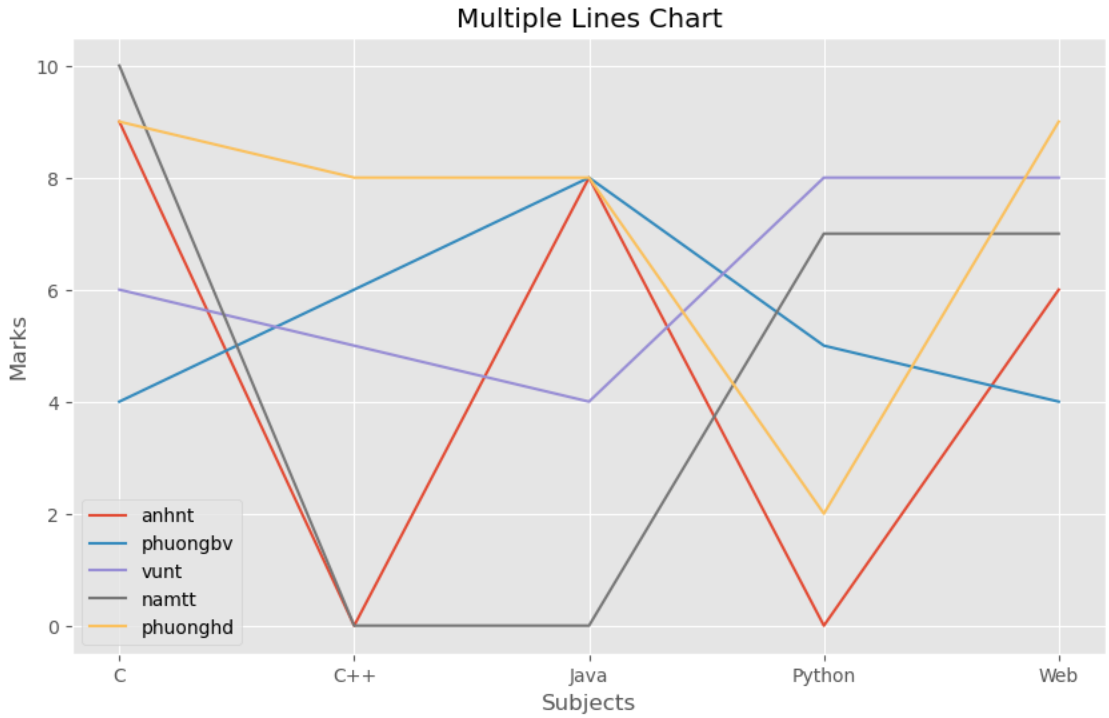
[18]:
```
['C', 'C++', 'Java', 'Python', 'Web']
```

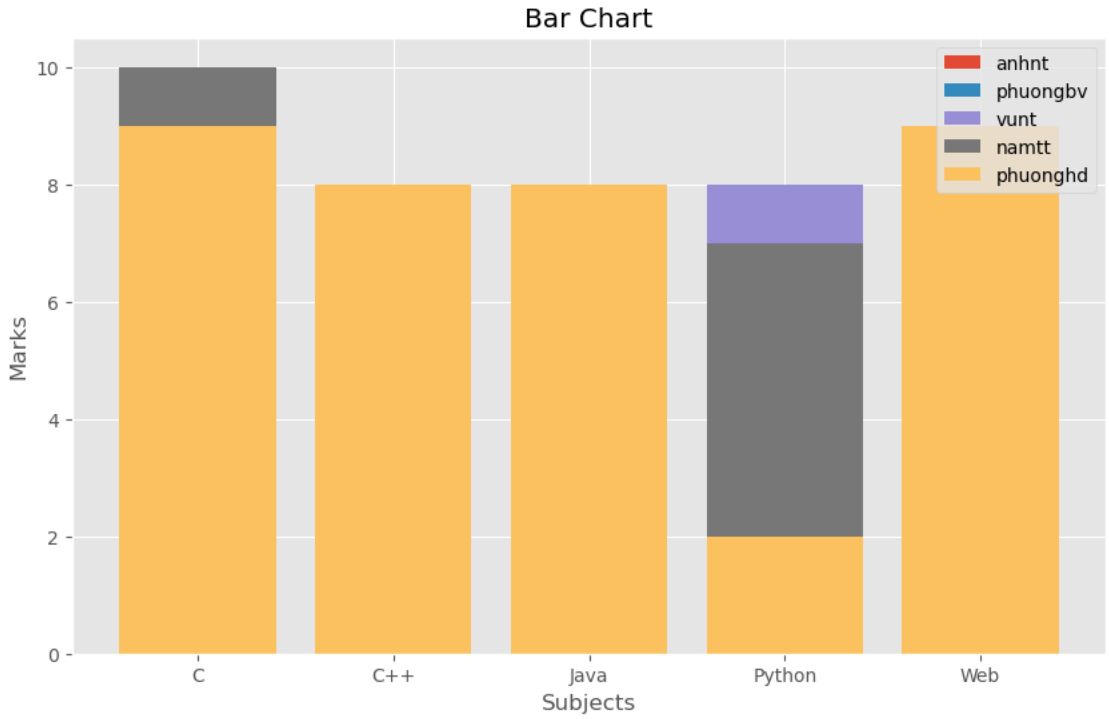
[19]:
```
plt.bar(
  subjects,
  nam_grade,
  label="nam",
  color = "m",
  align = "center"
  )

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

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



[20]:
```
student_names = ['anhnt', 'phuongbv', 'vunt', 'namtt', 'phuonghd']
subjects = ['C', 'C++', 'Java', 'Python', 'Web']
data = []

for student in student_names:
    marks = [random.randint(0, 10) for _ in subjects]
    student_data = {'student_name': student, 'subjects': subjects, 'marks':␣
 ↪marks}
    data.append(student_data)

plt.figure(figsize=(10, 6))
for student_data in data:
    plt.plot(student_data['subjects'], student_data['marks'],␣
 ↪label=student_data['student_name'])

plt.xlabel('Subjects')
plt.ylabel('Marks')
plt.title('Multiple Lines Chart')
plt.legend()
```

```
plt.show()

plt.figure(figsize=(10, 6))
for i, student_data in enumerate(data):
    plt.bar(student_data['subjects'], student_data['marks'],␣
 ↪label=student_data['student_name'])

plt.xlabel('Subjects')
plt.ylabel('Marks')
plt.title('Bar Chart')
plt.legend()
plt.show()

for student_data in data:
    print("{:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format(
        student_data['student_name'],
        student_data['marks'][0],
        student_data['marks'][1],
        student_data['marks'][2],
        student_data['marks'][3],
        student_data['marks'][4]
    ))
```





anhnt	9	0	8	0
6				
phuongbv	4	6	8	5
4				
vunt	6	5	4	8
8				
namtt	10	0	0	7
7				
phuonghd	9	8	8	2
9				

[ ]:

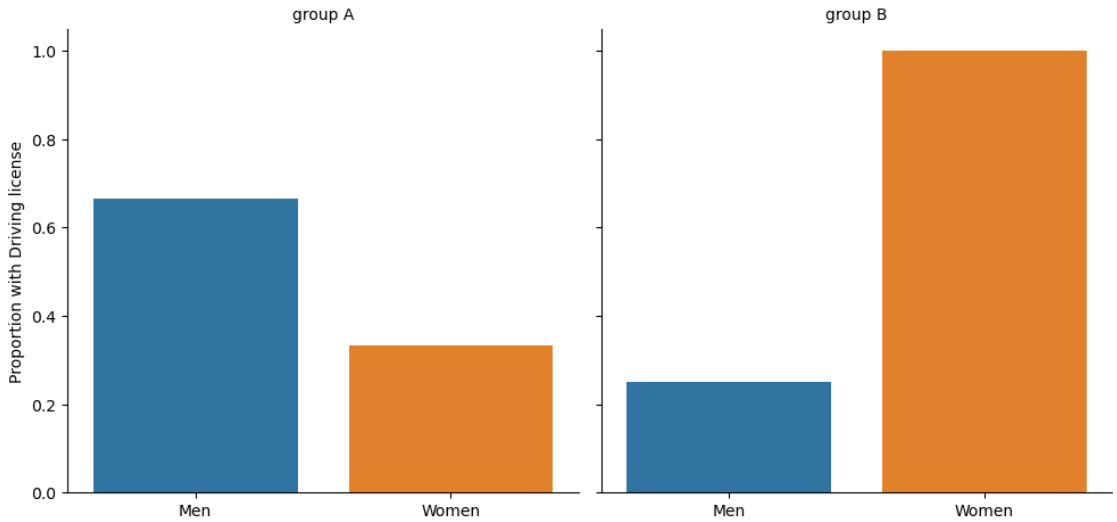
1-9

August 27, 2023

[1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

[7]:
```
data = pd.read_csv('data/drivinglicense.csv')
```

[17]:
```
#--- 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")
g.set_xticklabels(["Men", "Women"])
g.set_titles("{col_var} {col_name}")

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



[ ]:
```

```

1-10

August 27, 2023

[1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

[3]:
```
df = pd.read_csv('data/titanic.csv')
```

[6]:
```
df
```

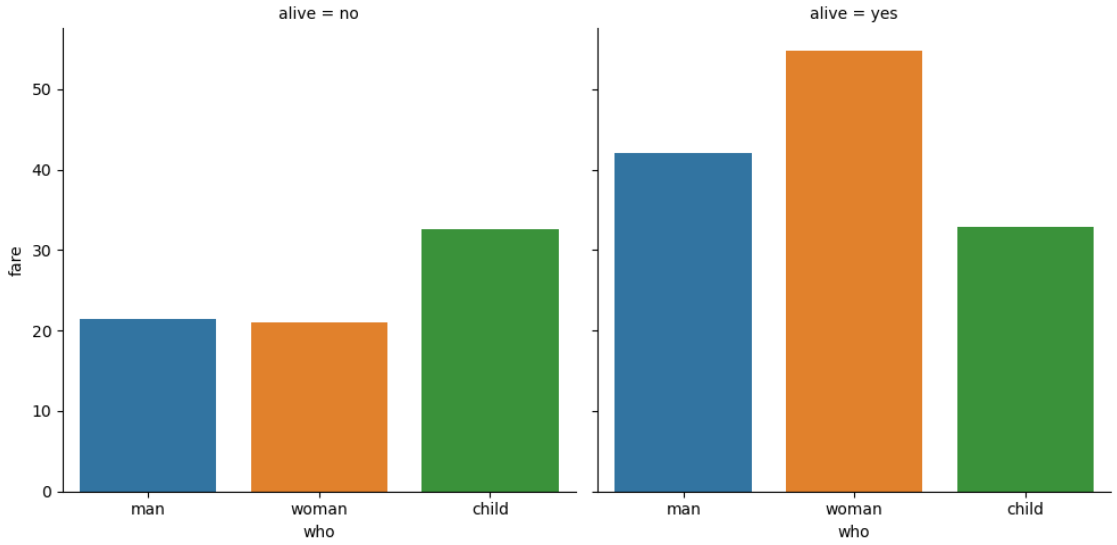
[6]:

	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]

[4]:
```
g = sns.catplot(x="who", y="fare", col="alive",
                data=df, kind="bar", errorbar=None, aspect=1.0)
```



1-11

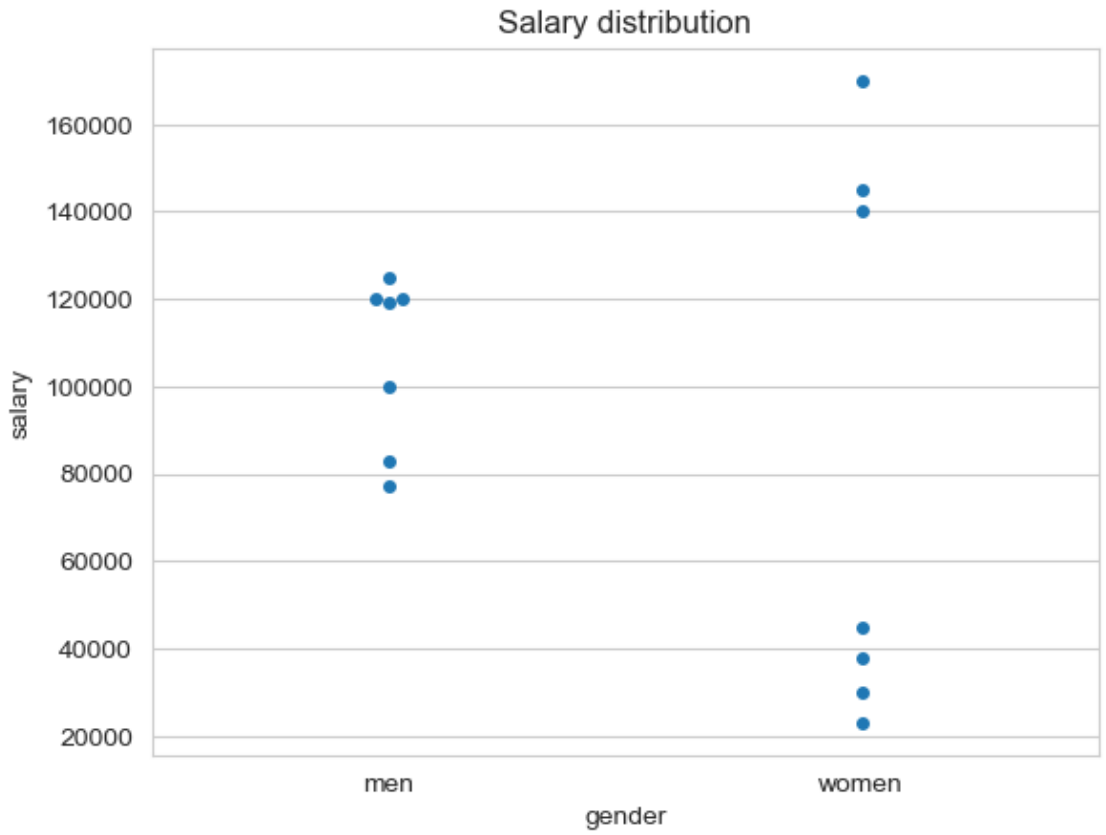
August 27, 2023

[2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

[3]:
```
sns.set_style("whitegrid")
```

[4]:
```
data = pd.read_csv('data/salary.csv')
```

[8]:
```
sns.swarmplot(x="gender", y="salary", data=data)

ax = plt.gca()
ax.set_title("Salary distribution")

plt.show()
```


Salary distribution
160000
140000
120000
100000
80000
60000
40000
20000
salary
men
women
gender

1-12

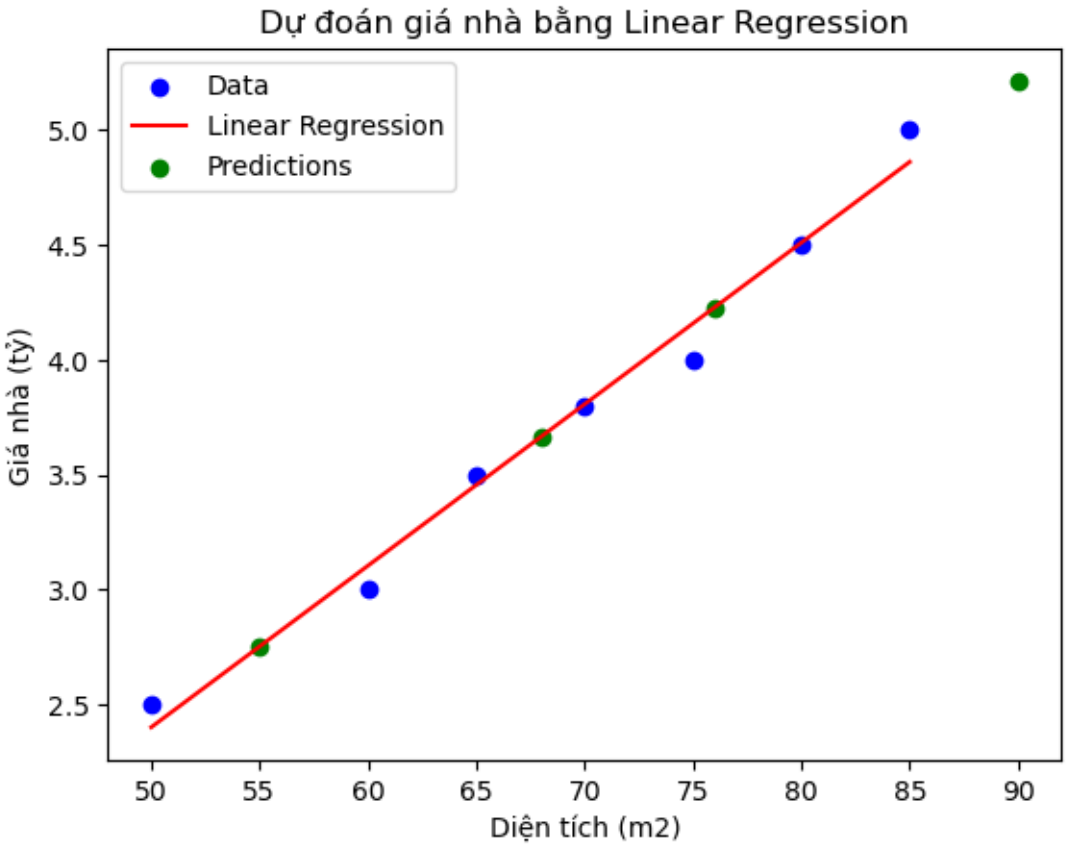
August 27, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

[2]:
```
data = np.array([(50, 2.5), (60, 3), (65, 3.5), (70, 3.8), (75, 4), (80, 4.5),␣
 ↪(85, 5)])
data
```

[2]:
```
array([[50. ,  2.5],
       [60. ,  3. ],
       [65. ,  3.5],
       [70. ,  3.8],
       [75. ,  4. ],
       [80. ,  4.5],
       [85. ,  5. ]])
```

[3]:
```
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)

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('Diện tích (m2)')
plt.ylabel('Giá nhà (tỷ)')
plt.title('Dự đoán giá nhà bằng Linear Regression')
plt.legend()
plt.show()
```


Dự đoán giá nhà bằng Linear Regression
Data
Linear Regression
Predictions
Giá nhà (tỷ)
Diện tích (m2)
2.5
3.0
3.5
4.0
4.5
5.0
50
55
60
65
70
75
80
85
90

1-13

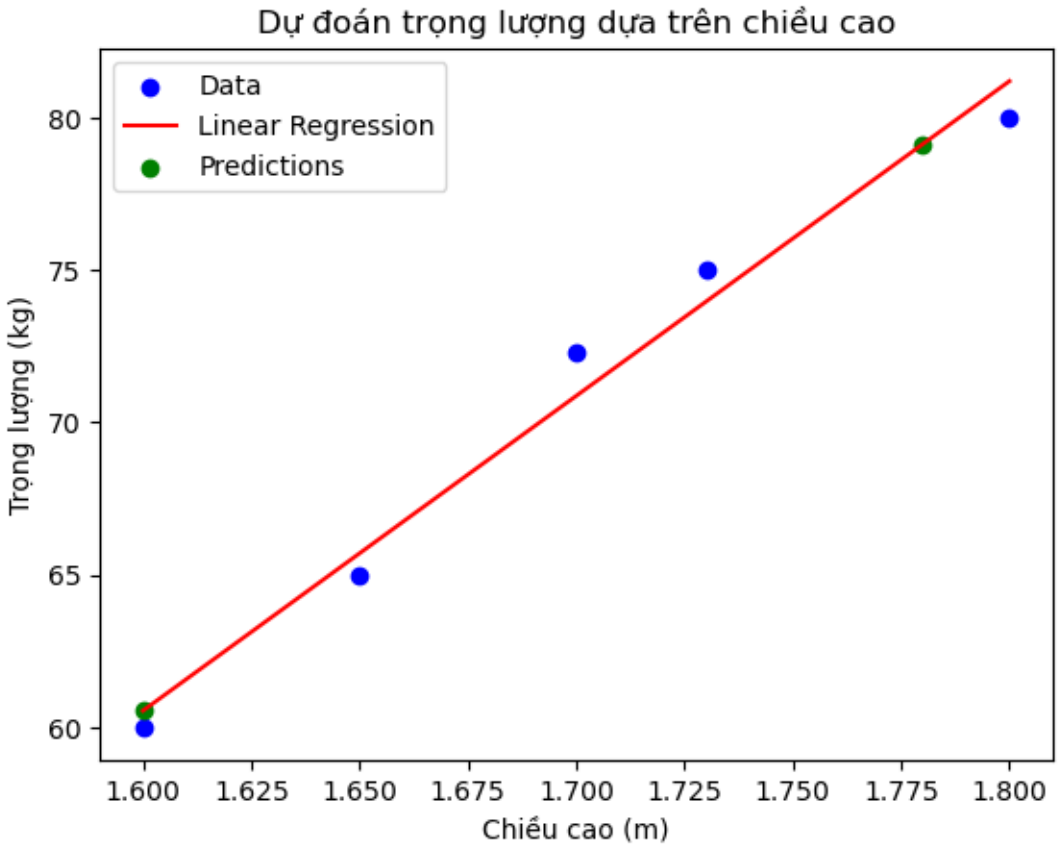
August 27, 2023

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
```

[10]:
```
heights = [[1.6], [1.65], [1.7], [1.73], [1.8]]

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

[22]:
```
new_heights = np.array([[1.6], [1.78]])
predicted_weights = model.predict(new_heights)
```

[23]:
```
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('Chiều cao (m)')
plt.ylabel('Trọng lượng (kg)')
plt.title('Dự đoán trọng lượng dựa trên chiều cao')
plt.legend()
plt.show()
```


Dự đoán trọng lượng dựa trên chiều cao
Data
Linear Regression
Predictions
Trọng lượng (kg)
80
75
70
65
60
1.600
1.625
1.650
1.675
1.700
1.725
1.750
1.775
1.800
Chiều cao (m)

1-14

August 27, 2023

[1]:
```
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
```

[2]:
```
dataset = fetch_openml(name='boston')
dataset.data
```

```
C:\Users\Admin\anaconda3\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\Admin\anaconda3\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(
```

[2]:

	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]
```

[3]:
```
dataset.feature_names
```

[3]:
```
['CRIM',
 'ZN',
 'INDUS',
 'CHAS',
 'NOX',
 'RM',
 'AGE',
 'DIS',
 'RAD',
 'TAX',
 'PTRATIO',
 'B',
 'LSTAT']
```

[4]:
```
dataset.DESCR
```

[4]:
```
"**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."
```

[5]:
```
dataset.target
```

[5]:
```
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
```

[6]:
```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df.head()
```

[6]:

	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

[7]:
```
df['MEDV']=dataset.target
df.head()
```

[7]:

	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
```

[8]:
```
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
```

[9]:
```
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
```

[12]:
```
corr = df.corr()
corr
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_13576\2438084875.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  corr = df.corr()
```

[12]:

	CRIM	ZN	INDUS	NOX	RM	AGE	DIS
CRIM	1.000000	-0.200469	0.406583	0.420972	-0.219247	0.352734	-0.379670
ZN	-0.200469	1.000000	-0.533828	-0.516604	0.311991	-0.569537	0.664408
INDUS	0.406583	-0.533828	1.000000	0.763651	-0.391676	0.644779	-0.708027
NOX	0.420972	-0.516604	0.763651	1.000000	-0.302188	0.731470	-0.769230
RM	-0.219247	0.311991	-0.391676	-0.302188	1.000000	-0.240265	0.205246
AGE	0.352734	-0.569537	0.644779	0.731470	-0.240265	1.000000	-0.747881
DIS	-0.379670	0.664408	-0.708027	-0.769230	0.205246	-0.747881	1.000000
TAX	0.582764	-0.314563	0.720760	0.668023	-0.292048	0.506456	-0.534432
PTRATIO	0.289946	-0.391679	0.383248	0.188933	-0.355501	0.261515	-0.232471
B	-0.385064	0.175520	-0.356977	-0.380051	0.128069	-0.273534	0.291512
LSTAT	0.455621	-0.412995	0.603800	0.590879	-0.613808	0.602339	-0.496996
MEDV	-0.388305	0.360445	-0.483725	-0.427321	0.695360	-0.376955	0.249929

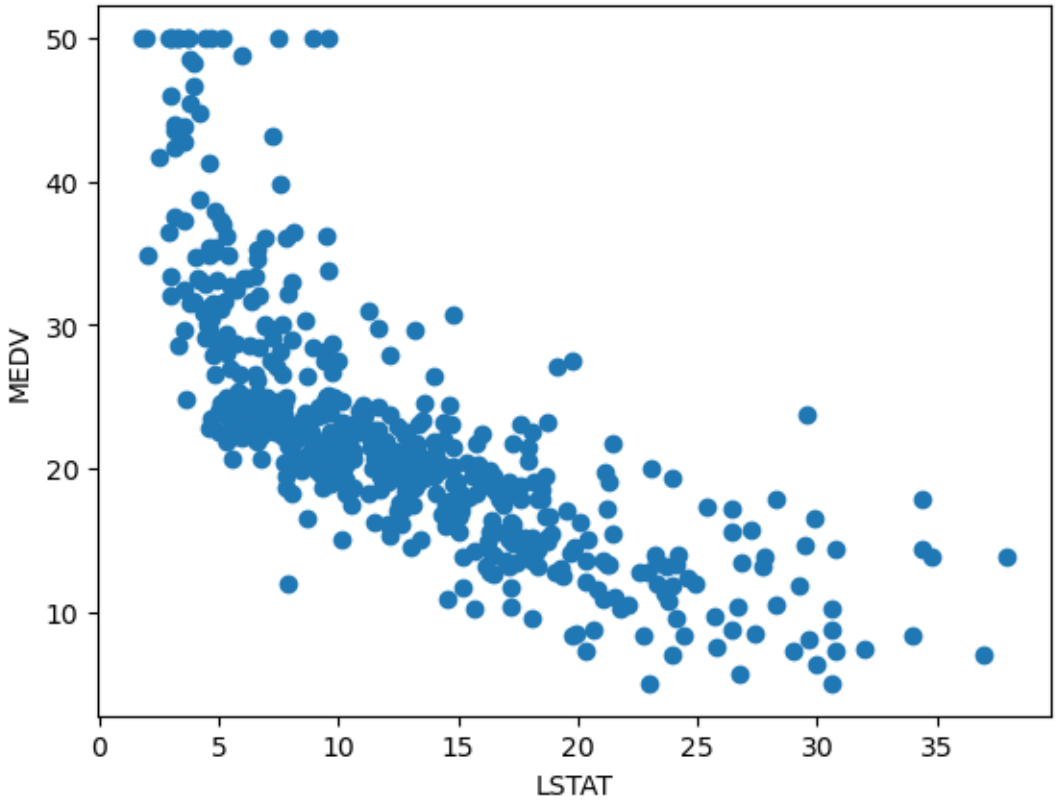
	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	0.582764	0.289946	-0.385064	0.455621	-0.388305
ZN	-0.314563	-0.391679	0.175520	-0.412995	0.360445
INDUS	0.720760	0.383248	-0.356977	0.603800	-0.483725
NOX	0.668023	0.188933	-0.380051	0.590879	-0.427321
RM	-0.292048	-0.355501	0.128069	-0.613808	0.695360
AGE	0.506456	0.261515	-0.273534	0.602339	-0.376955
DIS	-0.534432	-0.232471	0.291512	-0.496996	0.249929
TAX	1.000000	0.460853	-0.441808	0.543993	-0.468536
PTRATIO	0.460853	1.000000	-0.177383	0.374044	-0.507787
B	-0.441808	-0.177383	1.000000	-0.366087	0.333461
LSTAT	0.543993	0.374044	-0.366087	1.000000	-0.737663
MEDV	-0.468536	-0.507787	0.333461	-0.737663	1.000000

[16]:
```
print(corr.abs().nlargest(3, 'MEDV').index)

print(corr.abs().nlargest(3, 'MEDV').values[:,11])
```

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

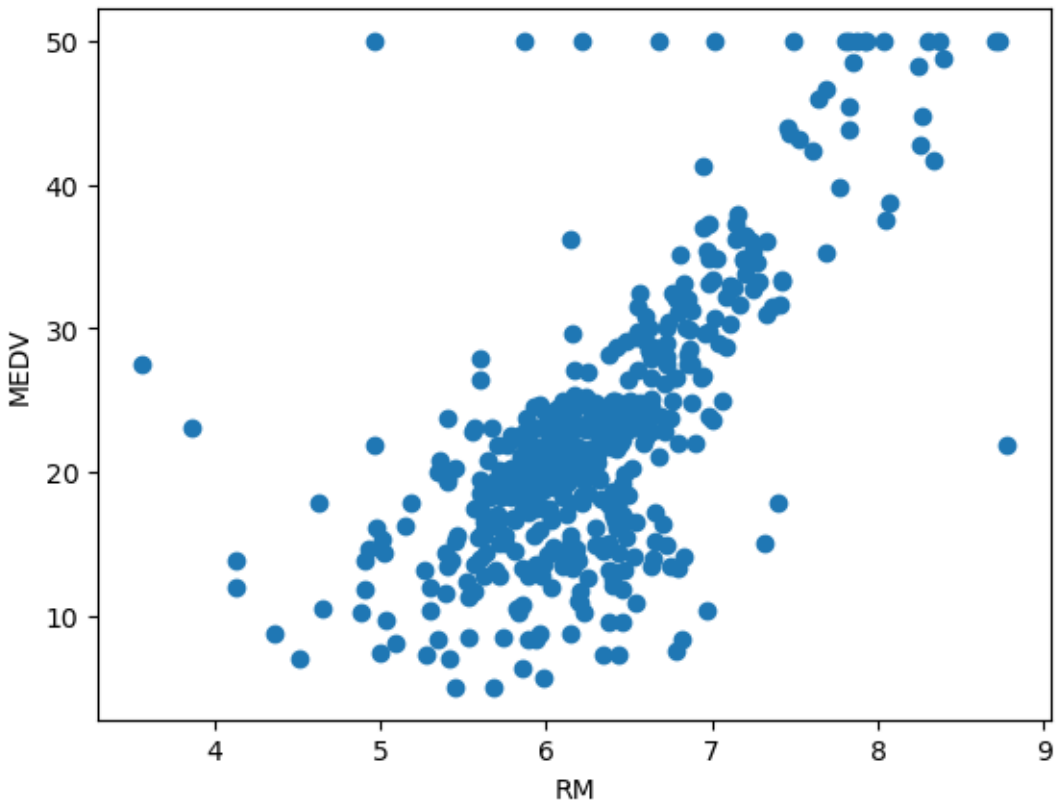
[17]:
```
plt.scatter(df['LSTAT'], df['MEDV'], marker='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
```

[17]: Text(0, 0.5, 'MEDV')

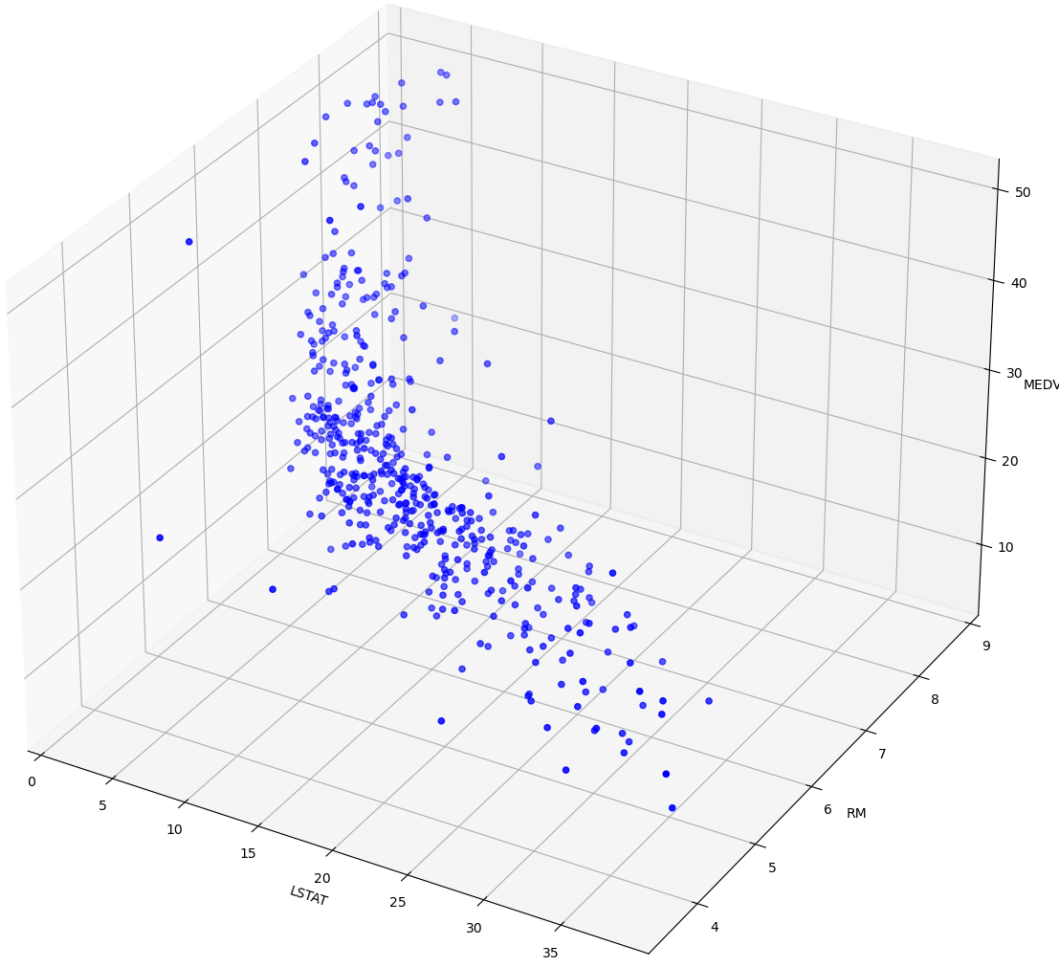


[18]:
```
plt.scatter(df['RM'], df['MEDV'], marker='o')
plt.xlabel('RM')
plt.ylabel('MEDV')
```

[18]: Text(0, 0.5, 'MEDV')



[19]:
```
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()
```



[20]:
```
x = pd.DataFrame(np.c_[df['LSTAT'], df['RM']], columns = ['LSTAT','RM'])
Y = df['MEDV']
```

[21]:
```
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)
```

[22]:
```
print(x_train.shape)
print(Y_train.shape)
```

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

[23]:
```
print(x_test.shape)
print(Y_test.shape)
```

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

[24]:
```
model = LinearRegression()
model.fit(x_train, Y_train)
price_prediction = model.predict(x_test)
```

[25]:
```
print('R-Squared: %.4f' % model.score(x_test,Y_test))
```

```
R-Squared: 0.6162
```

[26]:
```
mse = mean_squared_error(Y_test, price_prediction)
mse
```

[26]:
```
36.49422110915324
```

[27]:
```
plt.scatter(Y_test, price_prediction)
plt.xlabel("Actual price")
plt.ylabel("Predicted prices")
plt.title("Actual prices vs Predicted prices")
```

[27]:
```
Text(0.5, 1.0, 'Actual prices vs Predicted prices')
```

[28]:
```
print(model.intercept_)
print(model.coef_)
```

```
0.3843793678034615
[-0.65957972  4.83197581]
```

[31]:
```
print(model.predict([[30,5]]))
```

```
[4.75686695]
```

```
C:\Users\Admin\anaconda3\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 Plotting the 3D Hyperlane

[32]:
```
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\Admin\anaconda3\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\Admin\anaconda3\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(
```