

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN TRÍ TUỆ NHÂN TẠO

Đề tài: Tìm hiểu ID3 và ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ

Giảng viên hướng dẫn: ThS Lê Thị Thủy
Lớp học phần: 20221IT6046011

Nhóm 15:

- 1. Kiều Đức Anh - 2020601000**
- 2. Phan Tiến Phương - 2020602782**
- 3. Thân Ngọc Thiện - 2021602775**

Hà Nội, năm 2022

MỤC LỤC

Chương 1. Tổng quan về trí tuệ nhân tạo	4
1.1. Khái niệm về trí tuệ nhân tạo	4
1.2. Vai trò của trí tuệ nhân tạo	7
1.3. Kỹ thuật trong TTNT là gì và một số kỹ thuật cơ bản trong TTNT	12
1.4. Lịch sử phát triển của trí tuệ nhân tạo	13
1.5. Các thành phần trong hệ thống của trí tuệ nhân tạo	15
1.6. Các lĩnh vực nghiên cứu và ứng dụng cơ bản	15
Chương 2. Tìm hiểu về giải thuật ID3	17
2.1. Giới thiệu về cây quyết định	17
2.2. Tìm hiểu về thuật toán ID3	18
2.3. Lập trình Python cho ID3	28
Chương 3. Ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ	30
3.1. Bộ dữ liệu	30
3.2. Mô hình bài toán	32
3.3. Đánh giá	42
3.4. Xây dựng cây quyết định	42
Tài liệu tham khảo	44

Lời mở đầu

Ngày nay, Trí tuệ nhân tạo (AI) dần đi vào cuộc sống của chúng ta một cách mạnh mẽ và được xác định là lĩnh vực mũi nhọn trong ngành kinh tế các nước. Cách mạng công nghiệp 4.0 mở ra một thời kỳ mới với việc ứng dụng trí tuệ nhân tạo trong hầu hết các lĩnh vực trong đời sống, mang lại những thay đổi lớn trong xã hội, đặc biệt là trong kinh tế và khoa học ứng dụng.

Nội dung cuốn báo cáo này cung cấp cho người đọc những kiến thức chung nhất về trí tuệ nhân tạo, có một cái nhìn tổng quát về việc áp dụng thuật toán ID3 và Ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ. Các vấn đề cụ thể trình bày trong cuốn báo cáo được chia thành 3 chương:

Chương 1: Tổng quan về trí tuệ nhân tạo

Chương 2: Tìm hiểu về ID3

Chương 3: Ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ

Nhóm em xin gửi lời cảm ơn đến cô Lê Thị Thủy - Giảng viên Trường ĐHCNHN, thầy đã tận tình hướng dẫn và góp ý để hoàn thành bài báo cáo. Mặc dù đã rất cố gắng xong báo cáo này không tránh khỏi những thiếu sót, nhóm em mong nhận những đóng góp ý kiến của thầy và bạn đọc để bài báo cáo được hoàn thiện hơn.

Xin chân thành cảm ơn!

Nhóm sinh viên thực hiện

Chương 1. Tổng quan về trí tuệ nhân tạo

1.1. Khái niệm về trí tuệ nhân tạo

Theo như cha đẻ của trí tuệ nhân tạo, John McCarthy thì nó là "Khoa học và kỹ thuật của việc tạo ra những máy thông minh, đặc biệt là chương trình máy tính thông minh".

Trí tuệ nhân tạo là hướng đi của việc tạo ra máy tính, người máy điều khiển bằng máy tính hay là những phần mềm suy nghĩ thông minh hơn, tương tự như suy nghĩ thông minh của con người.

Trí tuệ nhân tạo được học như bộ não con người, như cách mà con người học, quyết định và làm việc khi giải quyết một vấn đề, và sau đó sử dụng kết quả của quá trình học đó như là nền tảng của việc phát triển phần mềm và hệ thống thông minh.

Ở thời điểm hiện tại, Thuật ngữ này thường dùng để nói đến các MÁY TÍNH có mục đích không nhất định và ngành khoa học nghiên cứu về các lý thuyết và ứng dụng của trí tuệ nhân tạo. Tức là mỗi loại trí tuệ nhân tạo hiện nay đang dừng lại ở mức độ những máy tính hoặc siêu máy tính dùng để xử lý một loại công việc nào đó như điều khiển một ngôi nhà, nghiên cứu nhận diện hình ảnh, xử lý dữ liệu của bệnh nhân để đưa ra phác đồ điều trị, xử lý dữ liệu để tự học hỏi, khả năng trả lời các câu hỏi về chẩn đoán bệnh, trả lời khách hàng về các sản phẩm của một công ty,...

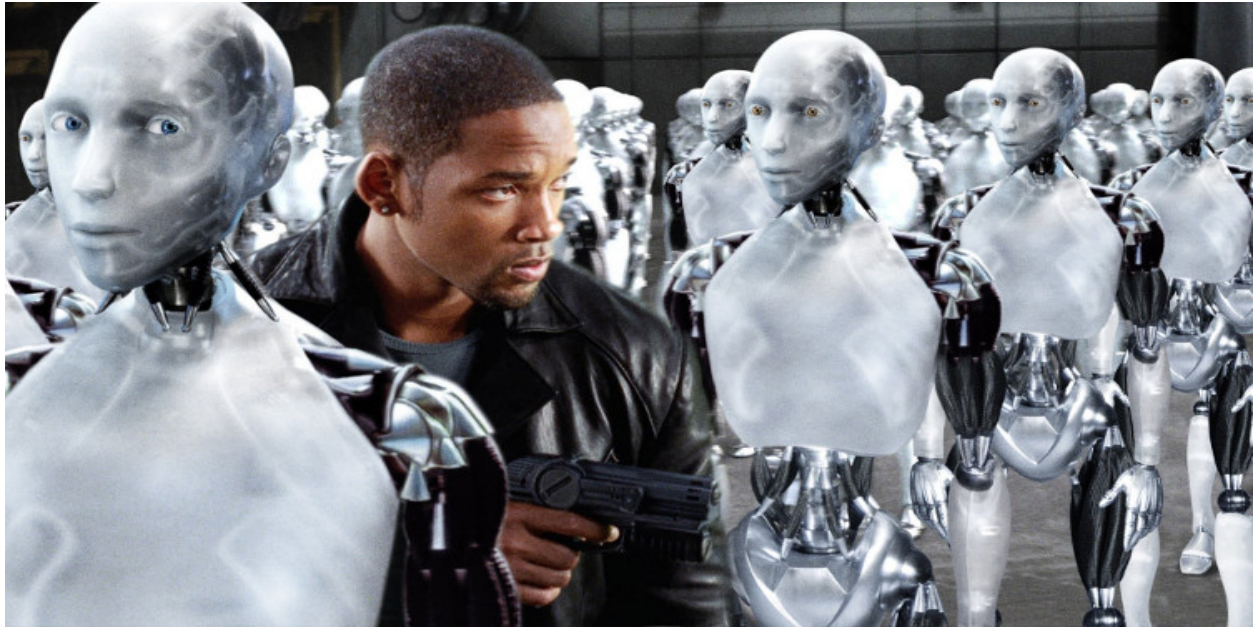


Hình 1.1. AI là một bộ phận của khoa học máy tính

Nói nôm na cho dễ hiểu: đó là trí tuệ của máy móc được tạo ra bởi con người. Trí tuệ này có thể tư duy, suy nghĩ, học hỏi,... như trí tuệ con người. Xử lý dữ liệu ở mức rộng lớn hơn, quy mô hơn, hệ thống, khoa học và nhanh hơn so với con người.

Rất nhiều hãng công nghệ nổi tiếng có tham vọng tạo ra được những AI (trí tuệ nhân tạo) vì giá trị của chúng là vô cùng lớn, giải quyết được rất nhiều vấn đề của con người mà loài người đang chưa giải quyết được.

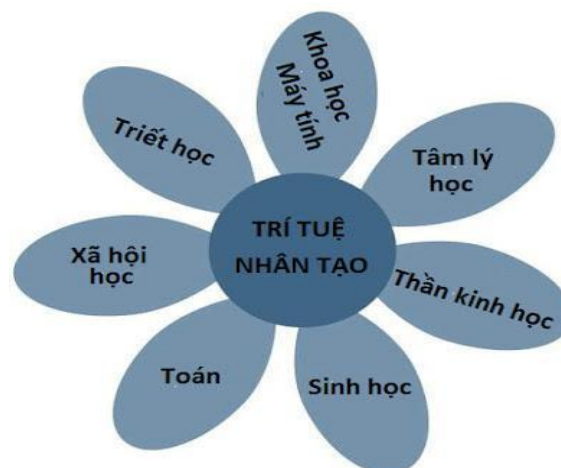
Trí tuệ nhân tạo mang lại rất nhiều giá trị cho cuộc sống loài người, nhưng cũng tiềm ẩn những nguy cơ. Rất nhiều chuyên gia lo lắng rằng khi trí tuệ nhân tạo đạt tới 1 ngưỡng tiến hóa nào đó thì đó cũng là thời điểm loài người bị tận diệt. Rất nhiều các bộ phim đã khai thác đề tài này với nhiều góc nhìn, nhưng qua đó đều muốn cảnh báo loài người về mối nguy đặc biệt này.



Hình 1.2. 1 cảnh trong bộ phim "I, Robot" nói về một AI đã tiến hóa

Trí tuệ nhân tạo là một ngành khoa học và công nghệ dựa trên nền tảng của Khoa học máy tính, Sinh học, Triết học, Ngôn ngữ học, Toán học và Kỹ thuật. Một chuyên ngành chính của Trí tuệ nhân tạo là phát triển chức năng của máy tính kết hợp với sự thông minh của con người, chẳng hạn như suy luận, học hỏi và giải quyết vấn đề.

Trong những lĩnh vực dưới đây, một hoặc nhiều lĩnh vực có thể góp thành để xây dựng hệ thống thông minh.



1.2. Vai trò của trí tuệ nhân tạo

Vai trò của AI là vô tận đối với cuộc sống của chúng ta. AI có thể tiếp cận với con người thông qua nhiều lĩnh vực, ngành nghề khác nhau. Ưu điểm của trí tuệ nhân tạo AI là khả năng xử lý dữ liệu khoa học hơn, nhanh hơn, hệ thống hơn so với con người. Việc phát triển và đưa các sản phẩm AI tới tay người dùng đúng cách sẽ thúc đẩy mạnh mẽ sự phát triển của toàn nhân loại. Mở ra một thế giới hoàn toàn mới cùng các giải pháp bù đắp cho những vấn đề mà con người không thể giải quyết.

Vai trò của trí tuệ nhân tạo trong y học



Hình 1.3. Áp dụng AI trong y khoa

Công nghệ AI đã mở ra một trang mới cho nền y học thế giới, đặc biệt là nền y học nước nhà. Nó mang đến cho con người những giá trị đáng kinh ngạc trong việc bảo vệ sức khỏe và điều trị bệnh tật. Tại lĩnh vực này, trí tuệ nhân tạo có vai trò quan trọng trong việc hỗ trợ điều trị y tế như định lượng thuốc, các phương pháp điều trị khác nhau cho bệnh nhân và quy trình phẫu thuật trong phòng mổ. Chúng sử dụng những thuật toán phân tích để hỗ trợ bệnh nhân theo dõi kết quả điều trị 24/7.

Vai trò của AI trong tài chính



Hình 1.4. Áp dụng AI trong lĩnh vực tài chính

Ngoài việc hỗ trợ con người chăm sóc sức khỏe, AI còn có vai trò quan trọng trong ngành tài chính ngân hàng. AI là công cụ giúp con người xử lý các hoạt động trong ngân hàng như xử lý giao dịch, theo dõi số dư, quản lý tài sản và các tài khoản tiền gửi lớn một cách nhanh chóng và chính xác nhất. Trí tuệ nhân tạo không những giúp các ngân hàng hợp lý hóa giao dịch mà còn có thể ước tính cung, cầu và định giá chứng khoán một cách dễ dàng hơn.

Vai trò của AI trong trò chơi và công nghệ

Hiện nay, những tập đoàn lớn đang ngày càng thúc đẩy việc sử dụng máy móc thông minh vào dây chuyền sản xuất. AI được sử dụng như các robot có thể thay thế một phần công việc của con người. Khối lượng công việc và thời gian hoàn thành sẽ nhanh chóng và nhẹ nhàng hơn dưới sự hoạt động của máy móc tích hợp trí tuệ nhân tạo. Tiêu biểu là với các sản phẩm như ô tô tự lái và trò chơi điện tử. Trong trò chơi điện tử, trí tuệ nhân tạo AI sẽ tự phân tích các hành vi và đưa ra những đáp án không kém cạnh với trí tuệ con người. Với ô tô tự lái, hệ thống AI tính toán tất cả các dữ liệu bên trong động cơ, tìm hiểu cách đi và ngăn chặn va chạm bởi chướng ngại vật

Sự kết hợp hoàn hảo của AI và robot hút bụi



Hình 1.5. Áp dụng AI để sản xuất robot hút bụi

Khi mọi người nghe đến trí tuệ nhân tạo, điều đầu tiên họ thường nghĩ đến là robot. Đối với lĩnh vực dọn dẹp tự động hóa gia đình, AI là điều không thể thiếu. Kết hợp các công nghệ tiên tiến cùng công nghệ AI siêu thông minh, các dòng máy robot hút bụi tự động liên tục được ra mắt trên thị trường. Tiêu biểu là dòng robot hút bụi Roomba của iRobot. Các sản phẩm tích hợp AI thường là những công cụ cao cấp nhất, đem lại hiệu quả cực lớn trong việc làm sạch sàn nhà của các hộ gia đình.

Với thời đại công nghệ 4.0 hiện nay, việc ứng dụng AI không còn xa lạ gì với cuộc sống của chúng ta. Trí tuệ nhân tạo có mặt trong mọi lĩnh vực đời sống từ giải trí cho đến y tế, xã hội. Đây chính là chìa khóa để mở ra một thế hệ mới đầy văn minh, thúc đẩy sự phát triển to lớn của loài người.

So sánh giữa lập trình không có TTNT và lập trình có TTNT

Lập trình không có TTNT	Lập trình có TTNT
Chương trình máy tính mà không có Trí tuệ nhân tạo thì chỉ có thể trả lời những câu hỏi xác định được quy định sẵn để giải quyết vấn đề.	Chương trình máy tính mà có Trí tuệ nhân tạo thì có thể trả lời những câu hỏi chung, cùng loại để giải quyết vấn đề.
Chỉnh sửa chương trình dẫn đến thay đổi trong cấu trúc của nó.	Chương trình TTNT có thể tiếp thu sự cập nhật cái mới bằng cách đề cao tính độc lập của những thông tin với nhau. Vì vậy bạn có thể sửa đổi một phần thông tin trong chương trình mà không làm ảnh hưởng đến cấu trúc của nó.
Việc chỉnh sửa thường không nhanh và không dễ dàng. Nó có thể dẫn đến việc ảnh hưởng chương trình của bạn.	Chỉnh sửa chương trình nhanh và dễ dàng.

Những tác động của TTNT đến sản xuất trong nền công nghiệp 4.0 như sau:

- **Chất lượng – Năng suất dự đoán :** Vai trò của trí tuệ nhân tạo đầu tiên là giảm thiểu các hao tổn trong sản xuất và ngăn ngừa các quy trình sản xuất kém hiệu quả. Khi nhu cầu ngày càng tăng để đáp ứng sự cạnh tranh thì trí tuệ nhân tạo là điều vô cùng cần thiết.
- **Bảo trì dự đoán :** Một trong những lợi ích của trí tuệ nhân tạo nữa là bảo trì dự đoán. Thay vì việc bảo trì theo lịch trình định trước thì bảo trì dự đoán sẽ sử dụng thuật toán để dự đoán lỗi tiếp theo của một bộ phận/máy móc/hệ thống. Nhờ đó có thể cảnh báo nhân viên thực hiện các quy trình bảo trì tập trung để ngăn chặn sự cố. Bảo trì dự đoán có ưu điểm là giảm đáng kể chi phí trong khi loại bỏ nhu cầu về thời gian ngừng hoạt động theo kế hoạch

trong nhiều trường hợp. Ngoài ra, nhờ nó mà Tuổi thọ hữu dụng còn lại của máy móc và thiết bị lâu hơn.

- Kết hợp giữa robot và con người



Hình 1.6. Robot thay thế con người trong một số công việc

Tính đến năm 2020, ước tính có khoảng 1,64 triệu robot công nghiệp đang hoạt động trên toàn thế giới. Robot sản xuất được chấp thuận làm việc cùng với con người để tăng năng suất công việc.

Khi áp dụng robot ngày càng nhiều thì AI sẽ đóng một vai trò quan trọng trong việc đảm bảo an toàn cho con người. Đồng thời trao cho robot nhiều trách nhiệm hơn trong việc đưa ra các quyết định có thể tối ưu hóa các quy trình dựa trên dữ liệu thời gian thực được thu thập từ sản xuất.

- Thiết kế sáng tạo : Nhà sản xuất có thể tận dụng trí tuệ nhân tạo vào giai đoạn thiết kế. Khi có bản tóm tắt thiết kế được xác định rõ ràng làm đầu vào

thì các nhà kỹ sư, thiết kế có thể sử dụng thuật toán AI. Mục đích để khám phá tất cả các cấu hình có thể có của một giải pháp.

- Nhu cầu cung ứng thị trường : Vai trò của trí tuệ nhân tạo cuối cùng mà chúng tôi muốn nhắc đến là cung ứng thị trường. Hiện nay trí tuệ nhân tạo đang hiện hữu ở mọi nơi trong hệ sinh thái công nghiệp 4.0. Nhà sản xuất có thể sử dụng các thuật toán AI để tối ưu hóa chuỗi cung ứng của các hoạt động sản xuất. Đồng thời giúp họ phản ứng và dự đoán tốt hơn những thay đổi trên thị trường.

1.3. Kỹ thuật trong TTNT là gì và một số kỹ thuật cơ bản trong TTNT

Trong thế giới thực, Tri thức có một vài thuộc tính như sau:

- Dung lượng đồ sộ, phi thường.
- Tổ chức tốt, định dạng tốt.
- Luôn luôn cập nhật sự thay đổi.

Kỹ thuật Trí tuệ nhân tạo là một cách để tổ chức và sử dụng tri thức có hiệu quả trong những cách sau đây:

- Có thể nhận thức được người đã cung cấp cho nó.
- Có thể sửa đổi dễ dàng để sửa lỗi.
- Nó có thể hữu ích trong một số tình huống dù nó chưa hoàn thiện hoặc chưa chính xác lắm.

Kỹ thuật Trí tuệ nhân tạo nâng cao tốc độ thực thi của những chương trình phức tạp.

Một số kỹ thuật Trí tuệ nhân tạo cơ bản :

- Lý thuyết giải bài toán và suy diễn thông minh
- Lý thuyết tìm kiếm may rủi
- Các ngôn ngữ về TTNT
- Lý thuyết thể hiện tri thức và hệ chuyên gia
- Lý thuyết nhận dạng và xử lý tiếng nói
- Người máy
- Tâm lý học xử lý thông tin
- Xử lý danh sách, kỹ thuật đệ quy, kỹ thuật quay lui và xử lý cú pháp hình thức

1.4. Lịch sử phát triển của trí tuệ nhân tạo

Đây là lịch sử của Trí tuệ nhân tạo trong suốt thế kỷ XX.

Năm	Cột mốc / Phát minh
1923	Vở kịch khoa học viễn tưởng của Karel Capek tên là "Rossum's Universal Robots" (RUR) diễn ra tại Luân Đôn (nước Anh). Lần đầu tiên sử dụng từ "robot" trong tiếng Anh.
1943	Nền tảng của mạng thần kinh được đặt nền móng.
1945	Isaac Asimov, một cựu sinh viên trường Đại học Columbia, đưa ra thuật ngữ "Robotics"
1950	Alan Turing giới thiệu Bài kiểm tra Turing để đánh giá sự thông minh và công bố Máy thông minh và Sự thông minh. Claude Shannon công bố "Phân tích chi tiết của việc chơi cờ".
1956	John McCarthy đưa ra thuật ngữ Trí tuệ nhân tạo. Biểu diễn chạy chương trình trí tuệ nhân tạo đầu tiên tại trường Đại học Carnegie Mellon.
1958	John McCarthy sáng tạo ra LISP, ngôn ngữ lập trình cho trí tuệ nhân tạo.
1964	Bài luận văn của Danny Bobrow tại MIT cho thấy máy tính có thể hiểu được ngôn ngữ tự nhiên của con người.
1965	Joseph Weizenbaum tại MIT đã xây dựng ELIZA, một vấn đề tương tác được mang trong đoạn đối thoại Tiếng Anh.

1969	Cá nhà khoa học tại Viện nghiên cứu Stanford đã phát triển Shakey, một robot, được trang bị sự vận động, nhận thức, và giải quyết vấn đề.
1973	Các nhóm hội về người máy tại Đại học Edinburgh đã xây dựng Freddy. Một người máy Scotland nổi tiếng, có khả năng sử dụng thị giác để định vị và lắp ráp mô hình.
1979	Xe tự quản được điều khiển bằng máy tính đầu tiên được xây dựng. Đó là Stanford Cart.
1985	Harold Cohen tạo và trình diễn chương trình đồ họa mang tên Aaron.
1990	Những chuyên đề nâng cao trong tất cả các lĩnh vực của Trí tuệ nhân tạo là: <ul style="list-style-type: none"> • Có tính chất quan trọng trong "học máy". • Suy luận theo tình huống • Lên lịch trình • Khai thác dữ liệu, thu thập web • Hiểu và dịch ngôn ngữ tự nhiên của con người • Thị giác và thực tế ảo • Ứng dụng trong trò chơi
1997	Chương trình "Deep Blue Chess" đánh bại nhà vô địch cờ thế giới, Garry Kasparov.
2000	Những robot thú cưng có sự tương tác đã được thương mại hóa. MIT đã trình diễn <i>Kismet</i> - một robot có khuôn mặt có thể biểu lộ cảm xúc. robot <i>Nomad</i> khám phá những vùng xa xôi hẻo lánh của Nam Cực và xác định thiên thạch.

1.5. Các thành phần trong hệ thống của trí tuệ nhân tạo

Hệ thống trí tuệ nhân tạo bao gồm hai thành phần cơ bản đó là biểu diễn tri thức và tìm kiếm tri thức trong miền biểu diễn:

$$\text{TTNT} = \text{Tri thức} + \text{Suy diễn}$$

Tri thức của bài toán có thể được phân ra làm ba loại cơ bản đó là tri thức mô tả, tri thức thủ tục và tri thức điều khiển.

Để biểu diễn tri thức người ta sử dụng các phương pháp sau đây:

- Phương pháp biểu diễn nhờ luật
- Phương pháp biểu diễn nhờ mạng ngữ nghĩa

- Phương pháp biểu diễn nhờ bộ ba liên hợp OAV
- Phương pháp biểu diễn nhờ Frame
- Phương pháp biểu diễn nhờ logic vị tư

Sau khi tri thức của bài toán đã được biểu diễn, kỹ thuật trong lĩnh vực trí tuệ nhân tạo là các phương pháp tìm kiếm trong miền đặc trưng tri thức về bài toán đó. Với mỗi cách biểu diễn sẽ có các giải pháp tương ứng. Các vấn đề này sẽ được đề cập trong chương 3.

1.6. Các lĩnh vực nghiên cứu và ứng dụng cơ bản

Trí tuệ nhân tạo có những ảnh hưởng vượt trội trong nhiều lĩnh vực như:

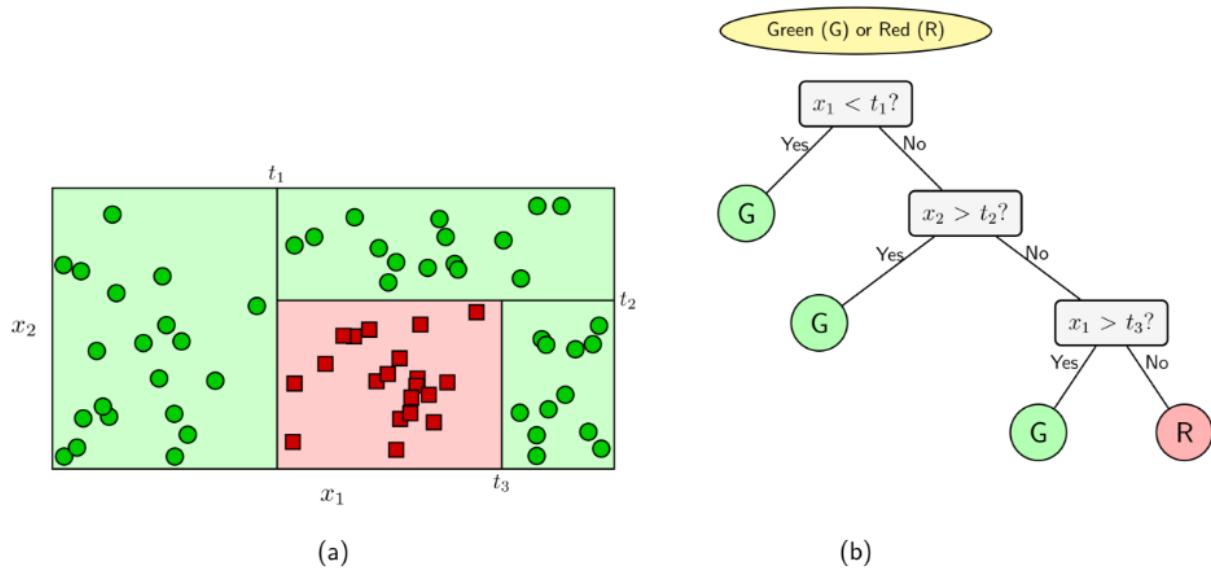
- Game - Trí tuệ nhân tạo đóng vai trò cốt yếu trong những game chiến lược như cờ, đánh bài, tic-tac-toe (như cờ caro), ... nơi mà máy móc có thể suy nghĩ số lớn những trường hợp có khả năng xảy ra dựa trên tri thức.
- Xử lý ngôn ngữ tự nhiên - Nó có khả năng tương tác với máy tính, hiểu ngôn ngữ tự nhiên mà con người nói.
- Hệ thống chuyên môn hóa - Có một vài ứng dụng mà các máy móc thông minh, phần mềm và những thông tin đặc biệt để suy luận. Nó giải thích và đưa ra lời khuyên cho người dùng hệ thống đó.
- Hệ thống thị giác - Hệ thống có thể hiểu, phân tích và tiếp thu dữ liệu vào thuộc về thị giác ngay trên máy tính. Ví dụ như:
 - Những máy bay do thám chụp lại hình ảnh, sau đó sử dụng kỹ thuật này để mô hình hóa những thông tin không gian hay bản đồ của khu vực.
 - Bác sỹ sử dụng hệ thống buồng bệnh chuyên môn để chẩn đoán cho bệnh nhân.
 - Cảnh sát có thể sử dụng phần mềm máy tính để nhận diện khuôn mặt của tội phạm từ những hình chân dung được vẽ lại bởi những họa sĩ pháp y.

- Nhận diện lời nói - Một vài hệ thống thông minh có khả năng nghe và tiếp thu ngôn ngữ trong cấu trúc và nghĩa của câu trong khi con người nói. Nó có thể nắm bắt được độ nhấn mạnh khác nhau, từ lóng, tiếng ồn phía sau, sự thay đổi trong âm thanh của con người do trời lạnh, ...
- Nhận diện chữ viết tay - Phần mềm nhận diện chữ viết tay đọc văn bản được viết trên giấy bằng bút hoặc viết trên màn hình bằng bút cảm ứng. Nó nhận dạng được hình dạng của chữ và chuyển nó thành văn bản có thể chỉnh sửa được.
- Người máy thông minh - Người máy có khả năng thực hiện nhiệm vụ mà con người giao cho. Nó có các cảm biến để nhận dạng các dữ liệu vật lý trong thế giới thực như ánh sáng, hơi nóng, nhiệt độ, sự di chuyển, âm thanh, sự va chạm và áp lực. Nó được trang bị bộ xử lý hiệu quả, đa cảm biến và bộ nhớ lớn để thể hiện sự thông minh. Hơn thế nữa, nó có khả năng học từ lỗi sai của nó và thích nghi với môi trường mới.

Chương 2. Tìm hiểu về giải thuật ID3

2.1. Giới thiệu về cây quyết định

Cây quyết định được dùng để đưa ra tập luật if – then nhằm mục đích dự báo, giúp con người nhận biết về tập dữ liệu. Cây quyết định cho phép phân loại đối tượng tùy thuộc vào các điều kiện tại các nút trong cây, bắt đầu từ gốc cây tới các nút sát lá-Nút xác định phân loại đối tượng. Mỗi nút trong của cây xác định điều kiện đối với thuộc tính mô tả của đối tượng. Mỗi nhánh tương ứng với điều kiện: Nút (thuộc tính) bằng giá trị nào đó. Đối tượng được phân loại nhờ tích hợp các điều kiện bắt đầu từ nút gốc của cây và các thuộc tính mô tả với giá trị của thuộc tính đối tượng.



Hình 2.1 Ví dụ về bài toán phân lớp sử dụng cây quyết định.

Ưu/nhược điểm của thuật toán cây quyết định

Ưu điểm

Cây quyết định là một thuật toán đơn giản và phổ biến. Thuật toán này được sử dụng rộng rãi bởi những lợi ích của nó:

- Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.
- Dữ liệu đầu vào có thể là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả
- Có thể làm việc với cả dữ liệu số và dữ liệu phân loại
- Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê
- Có khả năng làm việc với dữ liệu lớn

Nhược điểm

Kèm với đó, cây quyết định cũng có những nhược điểm cụ thể:

- Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thậm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.
- Cây quyết định hay gặp vấn đề overfitting.

2.2. Tìm hiểu về thuật toán ID3

2.2.1 Ý Tưởng

ID3 là một thuật toán decision tree được áp dụng cho các bài toán classification mà tất cả các thuộc tính đều ở dạng categorical.

Trong ID3, chúng ta cần xác định thứ tự của thuộc tính cần được xem xét tại mỗi bước. Với các bài toán có nhiều thuộc tính và mỗi thuộc tính có nhiều giá trị khác nhau, việc tìm được nghiệm tối ưu thường là không khả thi. Thay vào đó, một phương pháp đơn giản thường được sử dụng là tại mỗi bước, một thuộc tính tốt nhất sẽ được chọn ra dựa trên một tiêu chuẩn nào đó (chúng ta sẽ bàn sớm). Với mỗi thuộc tính được chọn, ta chia dữ liệu vào các child node tương ứng với các giá trị của thuộc tính đó rồi tiếp tục áp dụng phương pháp này cho mỗi child node. Việc chọn ra thuộc tính tốt nhất ở mỗi bước như thế này được gọi là cách chọn greedy (tham lam). Cách chọn này có thể không phải là tối ưu, nhưng trực giác cho chúng ta thấy rằng cách làm này sẽ gần với cách làm tối ưu. Ngoài ra, cách làm này khiến cho bài toán cần giải quyết trở nên đơn giản hơn.

Sau mỗi câu hỏi, dữ liệu được phân chia vào từng child node tương ứng với các câu trả lời cho câu hỏi đó. Câu hỏi ở đây chính là một thuộc tính, câu trả lời chính là giá trị của thuộc tính đó. Để đánh giá chất lượng của một cách phân chia, chúng ta cần đi tìm một phép đo.

Trước hết, thế nào là một phép phân chia tốt? Bằng trực giác, một phép phân chia là tốt nhất nếu dữ liệu trong mỗi child node hoàn toàn thuộc vào một class—khi đó child node này có thể được coi là một leaf node, tức ta không cần phân chia thêm nữa. Nếu dữ liệu trong các child node vẫn lẫn vào nhau theo tỉ lệ lớn, ta coi

rằng phép phân chia đó chưa thực sự tốt. Từ nhận xét này, ta cần có một hàm số đo độ tinh khiết (purity), hoặc độ vẩn đục (impurity) của một phép phân chia. Hàm số này sẽ cho giá trị thấp nhất nếu dữ liệu trong mỗi child node nằm trong cùng một class (tinh khiết nhất), và cho giá trị cao nếu mỗi child node có chứa dữ liệu thuộc nhiều class khác nhau.

2.2.2 Hàm số entropy

Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau $x_1, x_2, x_3, \dots, x_n$. Giả sử rằng xác suất để x nhận các giá trị này là

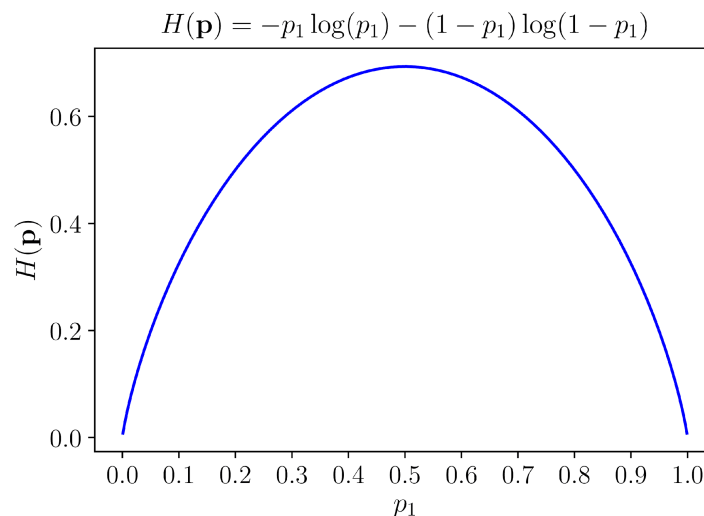
$p_i = p(x = x_i)$ với $0 \leq p_i \leq 1$, $\sum_{i=1}^n p_i = 1$. Ký hiệu phân phối này là

$p = (p_1, p_2, \dots, p_n)$. Entropy của phân phối này được định nghĩa là

$$H(p) = - \sum_{i=1}^n p_i \log \log(p_i) \quad (1)$$

Trong đó \log là logarit tự nhiên và quy ước $0 \log \log(0) = 0$.

Xét một ví dụ với $n=2$ được cho trên hình 2.2. Trong trường hợp \mathbf{p} là tinh khiết nhất, tức một trong hai giá trị của p_i bằng 1, giá trị kia bằng 0, entropy của phân phối này là $H(p) = 0$. Khi \mathbf{p} là vẩn đục nhất, tức cả hai giá trị $p_i = 0.5$, Hàm entropy đạt giá trị cao nhất.



Hình 2.2 Đồ thị của hàm entropy với $n = 2$

Tổng quát lên với $n > 2$, hàm entropy đạt giá trị nhỏ nhất nếu có một giá trị $p_i = 1$, đạt giá trị lớn nhất nếu tất cả các p_i bằng nhau. Những tính chất này của hàm entropy khiến nó được sử dụng trong việc đo độ vẩn đục của một phép phân chia của ID3. Vì lý do này, ID3 còn được gọi là entropy-based decision tree.

2.2.3 Thuật toán ID3

Trong ID3, tổng các trọng số của *entropy tại các leaf-node* sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Như đã đề cập, việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một *non-leaf node*. Chúng ta sẽ xây dựng phương pháp tính toán dựa trên mỗi node này.

Xét một bài toán với C class khác nhau. Giả sử ta đang làm việc với một *non-leaf node* với các điểm dữ liệu tạo thành một tập S với số phần tử là $|S| = N$. Giả sử thêm rằng trong số N điểm dữ liệu này, $N_c, c = 1, 2, 3, \dots, C$ điểm thuộc vào class

c . Xác suất để mỗi điểm dữ liệu rơi vào một class c được xấp xỉ bằng $\frac{N_c}{N}$

(*maximum likelihood estimation*). Như vậy, entropy tại node này được tính bởi:

$$H(S) = - \sum_{c=1}^C \frac{N_c}{N} \log \log \left(\frac{N_c}{N} \right) \quad (2)$$

Tiếp theo, giả sử thuộc tính được chọn là x . Dựa trên x , các điểm dữ liệu trong S được phân ra thành K child node S_1, S_2, \dots, S_K , với số điểm trong mỗi child node lần lượt là m_1, m_2, \dots, m_K . Ta định nghĩa

$$H(x, S) = - \sum_{k=1}^K \frac{m_k}{N} \log_2 \left(\frac{m_k}{N} \right) \quad (3)$$

là tổng có trọng số entropy của mỗi child node—được tính tương tự như (2). Việc lấy trọng số này là quan trọng vì các node thường có số lượng điểm khác nhau.

Tiếp theo, ta định nghĩa *information gain* dựa trên thuộc tính x :

$$G(x, S) = H(S) - H(x, S)$$

Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên:

$$x^* = \arg \max_x G(x, S) = \arg \max_x H(x, S)$$

tức thuộc tính khiến cho information gain đạt giá trị lớn nhất.

2.2.4 Ví dụ

Để mọi thứ được rõ ràng hơn, chúng ta cùng xem ví dụ với dữ liệu huấn luyện được cho trong Bảng dưới đây. Bảng dữ liệu này được lấy từ cuốn sách [Data Mining: Practical Machine Learning Tools and Techniques](#), trang 11. Đây là một bảng dữ liệu được sử dụng rất nhiều trong các bài giảng về decision tree. Bảng dữ liệu này mô tả mối quan hệ giữa thời tiết trong 14 ngày (bốn cột đầu, không tính cột id) và việc một đội bóng có chơi bóng hay không (cột cuối cùng). Nói cách khác, ta phải dự đoán giá trị ở cột cuối cùng nếu biết giá trị của bốn cột còn lại.

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

Có bốn thuộc tính thời tiết:

1. *Outlook* nhận một trong ba giá trị: sunny, overcast, rainy.
2. *Temperature* nhận một trong ba giá trị: hot, cool, mild.
3. *Humidity* nhận một trong hai giá trị: high, normal.
4. *Wind* nhận một trong hai giá trị: weak, strong.

(Tổng cộng có $3 \times 3 \times 2 \times 2 = 36$ loại thời tiết khác nhau, trong đó 14 loại được thể hiện trong bảng.)

Đây có thể được coi là một bài toán dự đoán liệu đội bóng có chơi bóng không dựa trên các quan sát thời tiết. Ở đây, các quan sát đều ở dạng categorical. Cách dự đoán dưới đây tương đối đơn giản và khá chính xác, có thể không phải là cách ra quyết định tốt nhất:

- Nếu *outlook* = sunny và *humidity* = high thì *play* = no.
- Nếu *outlook* = rainy và *windy* = true thì *play* = no.
- Nếu *outlook* = overcast thì *play* = yes.
- Ngoài ra, nếu *humidity* = normal thì *play* = yes.
- Ngoài ra, *play* = yes.

Chúng ta sẽ cùng tìm thứ tự các thuộc tính bằng thuật toán ID3.

Trong 14 giá trị đầu ra ở Bảng trên, có năm giá trị bằng no và chín giá trị bằng yes. Entropy tại *root node* của bài toán là:

$$H(S) = -\frac{5}{14} \log \log \left(\frac{5}{14} \right) - \frac{9}{14} \log \log \left(\frac{9}{14} \right) \approx 0.65$$

Tiếp theo, chúng ta tính tổng có trọng số entropy của các *child node* nếu chọn một trong các thuộc tính *outlook*, *temperature*, *humidity*, *wind*, *play* để phân chia dữ liệu.

Xét thuộc tính *outlook*. Thuộc tính này có thể nhận một trong ba giá trị *sunny*, *overcast*, *rainy*. Mỗi một giá trị sẽ tương ứng với một *child node*. Gọi tập hợp các điểm trong mỗi *child node* này lần lượt là S_s , S_o , S_r với tương ứng m_s , m_o , m_r phần tử. Sắp xếp lại Bảng ban đầu theo thuộc tính *outlook* ta đạt được ba Bảng nhỏ sau đây.

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
11	sunny	mild	normal	strong	yes

id	outlook	temperature	humidity	wind	play
3	overcast	hot	high	weak	yes
7	overcast	cool	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes

id	outlook	temperature	humidity	wind	play
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
10	rainy	mild	normal	weak	yes
14	rainy	mild	high	strong	no

Quan sát nhanh ta thấy rằng *child node* ứng với *outlook* = *overcast* sẽ có entropy bằng 0 vì tất cả $m_o = 4$ output đều là *yes*. Hai *child node* còn lại với

$m_s = m_r = 5$ có entropy khá cao vì tần suất output bằng *yes* hoặc *no* là xấp xỉ nhau. Tuy nhiên, hai *child node* này có thể được phân chia tiếp dựa trên hai thuộc tính *humidity* và *wind*.

Bạn đọc có thể kiểm tra được rằng

$$H(S_s) = -\frac{2}{5} \log \log \left(\frac{2}{5} \right) - \frac{3}{5} \log \log \left(\frac{3}{5} \right) \approx 0.673$$

$$H(S_o) = 0$$

$$H(S_r) = -\frac{2}{5} \log \log \left(\frac{2}{5} \right) - \frac{3}{5} \log \log \left(\frac{3}{5} \right) \approx 0.673 \quad H(\text{outlook}, S) = \frac{5}{14} H(S_s) + \frac{4}{14} H(S_o) + \frac{5}{14} H(S_r) \approx 0.48$$

Xét thuộc tính *temperature*, ta có phân chia như các Bảng dưới đây.

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
13	overcast	hot	normal	weak	yes

id	outlook	temperature	humidity	wind	play
4	rainy	mild	high	weak	yes
8	sunny	mild	high	weak	no
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
14	rainy	mild	high	strong	no

id	outlook	temperature	humidity	wind	play
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
9	sunny	cool	normal	weak	yes

Gọi S_h , S_m , S_c là ba tập con tương ứng với *temperature* bằng *hot*, *mild*, *cool*. Bạn

đọc có thể tính được

$$H(S_h) = -\frac{2}{4} \log \log \left(\frac{2}{4} \right) - \frac{2}{4} \log \log \left(\frac{2}{4} \right) \approx 0.693$$

$$H(S_m) = -\frac{4}{6} \log \log \left(\frac{4}{6} \right) - \frac{2}{6} \log \log \left(\frac{2}{6} \right) \approx 0.637$$

$$H(S_c) = -\frac{3}{4} \log \log \left(\frac{3}{4} \right) - \frac{1}{4} \log \log \left(\frac{1}{4} \right) \approx 0.562 \quad H(\text{temperature}, S) = \frac{4}{14} H(S_h) + \frac{6}{14} H(S_m) + \frac{4}{14} H(S_c) \approx 0.673$$

Việc tính toán với hai thuộc tính còn lại được dành cho bạn đọc. Nếu các kết quả là giống nhau, chúng sẽ bằng:

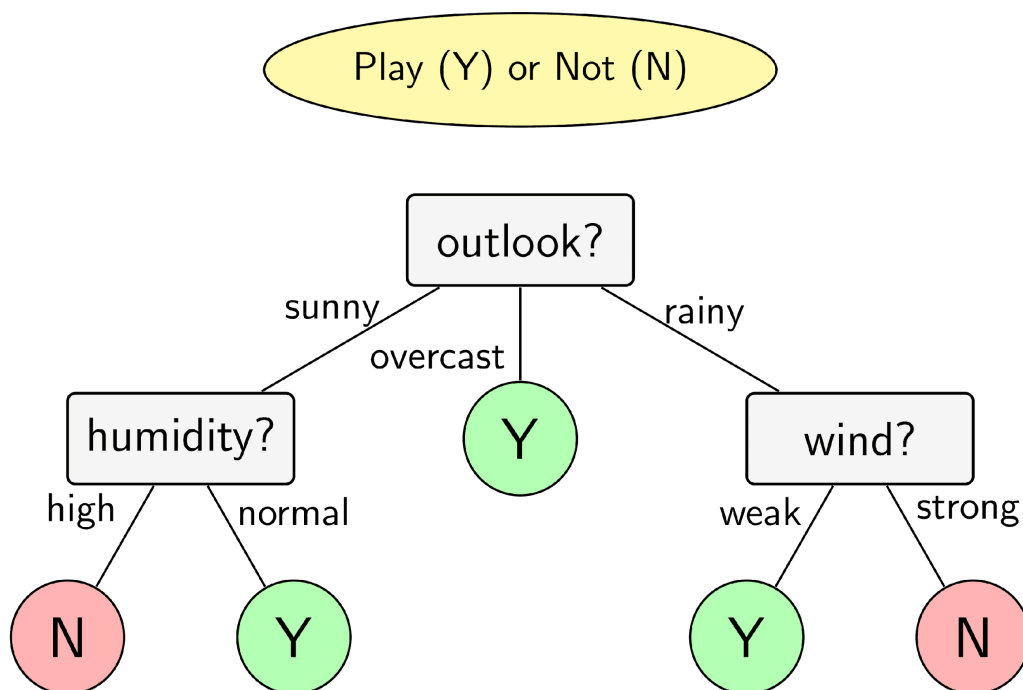
$$H(Humidity, S) \approx 0.547, H(Wind, S) \approx 0.618$$

Như vậy, thuộc tính cần chọn ở bước đầu tiên là *outlook* vì $H(Outlook, S)$ đạt giá trị nhỏ nhất (information gain là lớn nhất).

Sau bước phân chia đầu tiên này, ta nhận được ba child node với các phân tử như trong ba Bảng phân chia theo *outlook*. Child node thứ hai không cần phân chia tiếp vì nó đã *tinh khiết*. Với child node thứ nhất, ứng với *outlook = sunny*, kết quả tính được bằng ID3 sẽ cho chúng ta thuộc tính *humidity* vì tổng trọng số của entropy sau bước này sẽ bằng 0 với output bằng *yes* khi và chỉ khi *humidity = normal*.

Tương tự, child node ứng với *outlook = windy* sẽ được tiếp tục phân chia bởi thuộc tính *wind* với output bằng *yes* khi và chỉ khi *wind = weak*.

Như vậy, cây quyết định cho bài toán này dựa trên ID3 sẽ có dạng như trong Hình 2.3



Hình 2.3 Decision tree cho bài toán ví dụ sử dụng thuật toán ID3.

2.2.4 Điều kiện dừng

Trong các thuật toán decision tree nói chung và ID3 nói riêng, nếu ta tiếp tục phân chia các node *chưa tinh khiết*, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, tree có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng overfitting sẽ xảy ra.

Để tránh overfitting, một trong số các phương pháp sau có thể được sử dụng. Tại một node, nếu một trong số các điều kiện sau đây xảy ra, ta không tiếp tục phân chia node đó và coi nó là một leaf node:

- nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
- nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
- nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế *chiều sâu của tree* này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.
- nếu tổng số leaf node vượt quá một ngưỡng nào đó.
- nếu việc phân chia node đó không làm giảm entropy quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

Ngoài các phương pháp trên, một phương pháp phổ biến khác được sử dụng để tránh overfitting là *pruning*, tạm dịch là *cắt tỉa*.

2.2.5 Pruning

Pruning là một kỹ thuật regularization để tránh overfitting cho decision tree nói chung. Trong pruning, một decision tree sẽ được xây dựng tới khi mọi điểm trong training set đều được phân lớp đúng. Sau đó, các leaf node có chung một non-leaf node sẽ được *cắt tỉa* và non-leaf node đó trở thành một leaf-node, với class tương ứng với class chiếm đa số trong số mọi điểm được phân vào node đó. Việc cắt tỉa cây quyết định này có thể được xác định dựa vào các cách sau.

1. Dựa vào một validation set. Trước tiên, training set được tách ra thành một training set nhỏ hơn và một validation set. Decision tree được xây dựng trên training set cho tới khi mọi điểm trong training set được phân lớp đúng. Sau đó, đi ngược từ các leaf node, cắt tỉa các sibling node của nó và giữ lại node *bố mẹ* nếu độ chính xác trên validation set được cải thiện. Khi nào độ

chính xác trên validation set không được cải thiện nữa, quá trình pruning dừng lại. Phương pháp này còn được gọi là *reduced error pruning*.

2. Dựa vào toàn bộ data set. Trong phương pháp này, ta không tách tập training ban đầu ra mà sử dụng toàn bộ dữ liệu trong tập này cho việc xây dựng decision tree. Một ví dụ cho việc này là cộng thêm một đại lượng regularization vào hàm mất mát. Đại lượng regularization sẽ lớn nếu số leaf node là lớn. Cụ thể, giả sử decision tree cuối cùng có K leaf node, tập hợp các điểm huấn luyện rơi vào mỗi leaf node lần lượt là S_1, \dots, S_K . Khi đó, regularized loss của ID3 có thể được tính tương tự như (3):

$$L = \sum_{k=1}^K \frac{|S_k|}{|S|} H(S_k) + \lambda K \quad (5)$$

với $|S_k|$ ký hiệu số phần tử của tập hợp S_k và $H(S_k)$ chính là entropy của leaf node tương ứng với S_k , được tính tương tự như (2), và λ là một số thực

dương không quá lớn. Giá trị của hàm số này nhỏ nếu cả data loss—số hạng thứ nhất—nhỏ (entropy tại mỗi node là thấp) và regularization—số hạng thứ hai—cũng nhỏ (số leaf node là ít). Vì hàm mất mát trong (5) là một hàm rời rạc, rất khó để trực tiếp tối ưu hàm này. Việc tối ưu có thể được thực hiện thông qua pruning như sau. Trước hết, xây dựng một decision tree mà mọi điểm trong tập huấn luyện đều được phân loại đúng (toàn bộ các entropy của các node bằng 0). Lúc này data loss bằng 0 nhưng regularization có thể lớn, khiến cho \mathcal{L} lớn. Sau đó, ta có thể *tỉa* dần các leaf node sao cho \mathcal{L} giảm. Việc cắt tỉa được lặp lại đến khi \mathcal{L} không thể giảm được nữa.

2.3 Lập trình Python cho ID3

Module DecisionTree trong sklearn không thực hiện thuật toán ID3 mà là một thuật toán khác được đề cập trong bài tiếp theo. Phiên bản hiện tại trong sklearn chưa hỗ trợ các thuộc tính ở dạng categorical. Với dữ liệu có thuộc tính categorical, cách thường dùng là chuyển đổi các thuộc tính đó sang dạng numerical (1, 2, 3 cho mỗi giá trị). Chẳng hạn, các giá trị *hot*, *mild*, *cool* có thể lần lượt được thay bằng 1, 2, 3. Cách làm này có hạn chế vì trong cách chuyển đổi này, *mild* là trung bình cộng của *hot* và *cool*, nhưng nếu thứ tự các giá trị được đặt khác đi, việc chuyển đổi có thể ảnh hưởng lớn tới kết quả. Nhắc lại rằng các thuộc tính categorical, ví dụ màu sắc, thường không có tính thứ tự.

Xây dựng class TreeNode

```
from __future__ import print_function
import numpy as np
import pandas as pd

class TreeNode(object):
    def __init__(self, ids = None, children = [], entropy = 0, depth = 0):
        self.ids = ids          # index of data in this node
        self.entropy = entropy   # entropy, will fill later
        self.depth = depth      # distance to root node
        self.split_attribute = None # which attribute is chosen, it non-leaf
        self.children = children # list of its child nodes
        self.order = None       # order of values of split_attribute in children
        self.label = None       # label of node if it is a leaf

    def set_properties(self, split_attribute, order):
        self.split_attribute = split_attribute # split at which attribute
        self.order = order # order of this node's children

    def set_label(self, label):
        self.label = label # set label if the node is a leaf
```

Hàm tính entropy dựa trên tần suất

Trong hàm này, chúng ta phải chú ý bỏ các tần suất bằng 0 đi vì logarit tại đây không xác định.

```
def entropy(freq):
    # remove prob 0
    freq_0 = freq[np.array(freq).nonzero()[0]]
    prob_0 = freq_0/float(freq_0.sum())
    return -np.sum(prob_0*np.log(prob_0))
```

Dữ liệu trong ví dụ được lưu trong file weather.csv. Việc huấn luyện decision tree dựa trên ID3 cho tập dữ liệu này và đưa ra dự đoán cho training set được cho bởi

```
df = pd.DataFrame.from_csv('weather.csv')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
tree = DecisionTreeID3(max_depth = 3, min_samples_split = 2)
tree.fit(X, y)
print(tree.predict(X))
```

Kết quả

```
['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'yes']
```

Chương 3. Ứng dụng dự đoán nhu cầu mở tài khoản ký quỹ

3.1 Bộ dữ liệu:

- Dữ liệu được lấy trên trang kaggle.com - là một trong những trang uy tín, có danh tiếng để tìm dữ liệu phục vụ cho việc luyện tập Trí tuệ nhân tạo.

The screenshot shows the Kaggle website interface. On the left, there's a sidebar with a 'Create' button and links for 'Home', 'Competitions', and 'Datasets'. The main area features a search bar and a notebook titled 'Banking Dataset Classification' by 'RASHMI', updated 2 years ago. Below the title, it says 'Predicting if the client will subscribe to a term deposit'. There are buttons for 'New Notebook' and 'Download (418 kB)'. The page number '29' is visible in the center.

+ Mô tả dữ liệu:

Feature	Feature_Type	Description
age	numeric	age of a person
job	Categorical,nominal	type of job ('admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
marital	categorical,nominal	marital status ('divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)

education	categorical,nominal	('basic.4y','basic.6y','basic.9y', 'high.school','illiterate','professional.course', 'university.degree','unknown')
-----------	---------------------	---

default	categorical,nominal	has credit in default? ('no','yes','unknown')
housing	categorical,nominal	has housing loan? ('no','yes','unknown')
loan	categorical,nominal	has personal loan? ('no','yes','unknown')
contact	categorical,nominal	contact communication type ('cellular','telephone')

month	categorical,ordinal	last contact month of year ('jan', 'feb', 'mar', ..., 'nov', 'dec')
dayofweek	categorical,ordinal	last contact day of the week ('mon', 'tue', 'wed', 'thu', 'fri')

poutcome	categorical, nominal	'failure','nonexistent','success'
----------	----------------------	-----------------------------------

+ Kết quả đầu ra:

y	binary	'yes','no'
---	--------	------------

3.2 Mô hình bài toán

3.2.1 Tiền xử lý dữ liệu

- Khai báo một số thư viện cần thiết trong quá trình nhập và xử lý dữ liệu:

```
# import các thư viện cần thiết
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- Kết nối Google Colab với dữ liệu được lưu trong Google Driver. Điều này tránh cho việc trong quá trình training và sử dụng lại code phải tải lên lại dữ liệu

liệu lên Google Colab (do dữ liệu tập huấn tạm thời lưu trong thư mục sample data, sau một khoảng thời gian sẽ tự động bị xóa)

```
✓ [8] #kết nối colab với dữ liệu của Google Driver
2s from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
✓ [9] %cd /content/drive/MyDrive/ColabNotebooks/Datasets
0s
```

/content/drive/MyDrive/ColabNotebooks/Datasets

- Nhập dữ liệu và hiển thị thử 5 dòng đầu của dữ liệu đưa vào

```
✓ [10] df = pd.read_csv("new_train1.csv")
0s
```

```
✓ df.head()
0s
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	previous	outcome	y
0	49	blue-collar	married	basic.9y	unknown	no	no	cellular	nov	wed	227	0	nonexistent	no
1	37	entrepreneur	married	university.degree	no	no	no	telephone	nov	wed	202	1	failure	no
2	78	retired	married	basic.4y	no	no	no	cellular	jul	mon	1148	0	nonexistent	yes
3	36	admin.	married	university.degree	no	yes	no	telephone	may	mon	120	0	nonexistent	no
4	59	retired	divorced	university.degree	no	no	no	cellular	jun	tue	368	0	nonexistent	no

- Chuẩn hóa các dữ liệu về dạng số:

```
for i in range(0, len(df.age)):
    if 10 < df.age[i] < 20:
        df.age[i] = 0
    elif 20 < df.age[i] < 35:
        df.age[i] = 1
    elif 35 < df.age[i] < 45:
        df.age[i] = 2
    elif 40 < df.age[i] < 55:
        df.age[i] = 3
    elif 55 < df.age[i] < 65:
        df.age[i] = 4
    elif 65 < df.age[i] < 75:
        df.age[i] = 5
    elif 75 < df.age[i] < 85:
        df.age[i] = 6
    elif 75 < df.age[i] < 95:
```

```

df.age[i] = 7
else:
    df.age[i] = 8
df.age[i] += 1
print("age", i, " is changed to", df.age[i])

```

+ Bắt đầu với thuộc tính tuổi (age). Ở thuộc tính này ta chia dữ liệu thành 9 nhóm như sau:

- 10 đến 20 tuổi
- 20 đến 35 tuổi
- 35 đến 45 tuổi
- 45 đến 55 tuổi
- 55 đến 65 tuổi
- 65 đến 75 tuổi
- 75 đến 85 tuổi
- 85 đến 95 tuổi
- Và lớn hơn 95 tuổi

```

➞ age 32892  is changed to 9
   age 32893  is changed to 2
   age 32894  is changed to 4
   age 32895  is changed to 3
   age 32896  is changed to 2
   age 32897  is changed to 4
   age 32898  is changed to 4

```

Trong quá trình chuẩn hóa dữ liệu, ta in ra các bước mà chương trình đã thực hiện được để kiểm soát được chương trình đang làm công việc gì, kết quả ra sao.

+ Tương tự như thuộc tính ‘age’ ta tiếp tục chuẩn hóa các thuộc tính tiếp theo

```

['job',    'marital',    'education',    'default',    'housing',    'loan',
 'contact', 'month',    'day_of_week', 'poutcome']

//job
for i in range(0, len(df.job)):
    if df.job[i] == 'admin.':
        df.job[i] = 0

```

```

elif df.job[i] == 'blue-collar':
    df.job[i] = 1
elif df.job[i] == 'entrepreneur':
    df.job[i] = 2
elif df.job[i] == 'housemaid':
    df.job[i] = 3
elif df.job[i] == 'management':
    df.job[i] = 4
elif df.job[i] == 'retired':
    df.job[i] = 5
elif df.job[i] == 'self-employed':
    df.job[i] = 6
elif df.job[i] == 'services':
    df.job[i] = 7
elif df.job[i] == 'student':
    df.job[i] = 8
elif df.job[i] == 'technician':
    df.job[i] = 9
elif df.job[i] == 'unemployed':
    df.job[i] = 10
elif df.job[i] == 'unknown':
    df.job[i] = 11
//marital
for i in range(0, len(df.marital)):
    if df.marital[i] == 'divorced':
        df.marital[i] = 0
    elif df.marital[i] == 'married':
        df.marital[i] = 1
    elif df.marital[i] == 'single':
        df.marital[i] = 2
    elif df.marital[i] == 'unknown':
        df.marital[i] = 3
//education
for i in range(0, len(df.education)):
    if df.education[i] == 'basic.4y':
        df.education[i] = 0
    elif df.education[i] == 'basic.6y':
        df.education[i] = 1
    elif df.education[i] == 'basic.9y':
        df.education[i] = 2
    elif df.education[i] == 'high.school':
        df.education[i] = 3

```

```

elif df.education[i] == 'illiterate':
    df.education[i] = 4
elif df.education[i] == 'professional.course':
    df.education[i] = 5
elif df.education[i] == 'university.degree':
    df.education[i] = 6
elif df.education[i] == 'unknown':
    df.education[i] = 7
//default
for i in range(0, len(df.default)):
    if df.default[i] == 'no':
        df.default[i] = 0
    elif df.default[i] == 'yes':
        df.default[i] = 1
    elif df.default[i] == 'unknown':
        df.default[i] = 2
//housing
for i in range(0, len(df.housing)):
    if df.housing[i] == 'no':
        df.housing[i] = 0
    elif df.housing[i] == 'yes':
        df.housing[i] = 1
    elif df.housing[i] == 'unknown':
        df.housing[i] = 2
//loan
for i in range(0, len(df.loan)):
    if df.loan[i] == 'no':
        df.loan[i] = 0
    elif df.loan[i] == 'yes':
        df.loan[i] = 1
    elif df.loan[i] == 'unknown':
        df.loan[i] = 2
//contact
for i in range(0, len(df.contact)):
    if df.contact[i] == 'cellular':
        df.contact[i] = 0
    elif df.contact[i] == 'telephone':
        df.contact[i] = 1
//month
for i in range(0, len(df.month)):
    if df.month[i] == 'jan':
        df.month[i] = 0

```

```

elif df.month[i] == 'feb':
    df.month[i] = 1
elif df.month[i] == 'mar':
    df.month[i] = 2
elif df.month[i] == 'apr':
    df.month[i] = 3
elif df.month[i] == 'may':
    df.month[i] = 4
elif df.month[i] == 'jun':
    df.month[i] = 5
elif df.month[i] == 'jul':
    df.month[i] = 6
elif df.month[i] == 'aug':
    df.month[i] = 7
elif df.month[i] == 'sep':
    df.month[i] = 8
elif df.month[i] == 'oct':
    df.month[i] = 9
elif df.month[i] == 'nov':
    df.month[i] = 10
elif df.month[i] == 'dec':
    df.month[i] = 11
//day_of_week
for i in range(0, len(df.day_of_week)):
    if df.day_of_week[i] == 'mon':
        df.day_of_week[i] = 0
    elif df.day_of_week[i] == 'tue':
        df.day_of_week[i] = 1
    elif df.day_of_week[i] == 'wed':
        df.day_of_week[i] = 2
    elif df.day_of_week[i] == 'thu':
        df.day_of_week[i] = 3
    elif df.day_of_week[i] == 'fri':
        df.day_of_week[i] = 4
//duration
for i in range(0, len(df.duration)):
    if df.duration[i] == 0:
        df.duration[i] = 0
    else:
        df.duration[i] = 1
//poutcome
for i in range(0, len(df.poutcome)):


```

```

if df.poutcome[i] == 'failure':
    df.poutcome[i] = 0
elif df.poutcome[i] == 'nonexistent':
    df.poutcome[i] = 1
else:
    df.poutcome[i] = 2

```

- + Sau khi đã chuẩn hóa các thuộc tính của bộ dữ liệu xong ta thu được kết quả như sau:

 df.head()

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	previous	poutcome	y
0	4	1	1	2	2	0	0	0	10	2	1	0	1	0
1	3	2	1	6	0	0	0	1	10	2	1	1	0	0
2	7	5	1	0	0	0	0	0	6	0	1	0	1	1
3	3	0	1	6	0	1	0	1	4	0	1	0	1	0
4	5	5	0	6	0	0	0	0	5	1	1	0	1	0

3.2.2 Thiết lập mô hình

3.2.2.1 Thư viện sklearn

- Để xây dựng mô hình xử lý cho bài toán lần này, nhóm chúng em quyết định sử dụng thư viện Scikit-learn(sklearn) để thực hiện.
- Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.
- Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.
- Ví dụ (nguồn: vncoder.vn) **Classification and Regression Trees**
 - + Ở ví dụ sau, chúng ta sử dụng cây quyết định Decision tree phân loại để mô hình hóa bộ dữ liệu hoa Iris.

- + Bộ dữ liệu này được cung cấp dưới dạng tập dữ liệu mẫu với thư viện và được tải. Trình phân loại phù hợp với dữ liệu và sau đó dự đoán được thực hiện trên dữ liệu đào tạo.
- + Bộ dữ liệu này được cung cấp dưới dạng tập dữ liệu mẫu ngay trong thư viện sau đó được tải xuống. Thuật toán phân loại bắt đầu huấn luyện mô hình với bộ dữ liệu Iris ban đầu sau đó dự đoán lại các dữ liệu huấn luyện.
- + Cuối cùng, chúng ta đánh giá độ tốt của mô hình bằng quan sát accuracy và confusion matrix của 2 tập nhãn thực tế và nhãn dự đoán của mô hình.

```
# Sample Decision Tree Classifier

from sklearn import datasets

from sklearn import metrics

from sklearn.tree import DecisionTreeClassifier

# load the iris datasets

dataset = datasets.load_iris()

# fit a CART model to the data

model = DecisionTreeClassifier()

model.fit(dataset.data, dataset.target)

print(model)

# make predictions

expected = dataset.target

predicted = model.predict(dataset.data)

# summarize the fit of the model

print(metrics.classification_report(expected, predicted))

print(metrics.confusion_matrix(expected, predicted))
```

=> Như vậy có thể thấy, khi ta sử dụng thư viện sklearn code sẽ trở nên ngắn gọn và đơn giản hơn rất nhiều. Và vì được xây dựng trước nên cũng tránh được những sai sót trong quá trình thực hiện nếu tự xây dựng các class, hàm để tạo cây quyết định.

3.2.2.2 Thực hiện mô hình bài toán

- Chọn hệ số tương quan.

+ Code:

```
# chọn bằng hệ số tương quan  
  
cor = df.corr()
```

+ Đôi nét về hàm corr() :

corr() được sử dụng để tìm mối tương quan theo cặp của tất cả các cột trong trong Python. Mọi giá trị NaN sẽ tự động bị loại trừ. Bất kỳ loại hoặc cột dữ liệu không phải số nào trong Dataframe, nó sẽ bị bỏ qua.

- Chọn các cột có thuộc tính ảnh hưởng đến đầu ra của bài toán

+ Code:

```
cotdt = ['age', 'job', 'marital', 'education', 'default',  
         'housing', 'loan', 'contact', 'month', 'day_of_week',  
         'poutcome']  
  
x = df[cotdt]  
  
y = df['y'].astype('int')
```

- Chia tập dữ liệu ra làm hai phần ra train và test. Chia bộ dữ liệu train và test cân bằng nhau, sau đó ta trộn dữ liệu lên random_state = 26

+ Code:

```
//Thêm một số thành phần sử dụng của thư viện sklearn  
  
from sklearn.tree import DecisionTreeClassifier  
  
from sklearn.model_selection import train_test_split
```



```
//chia bộ dữ liệu làm hai phần
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y,  
test_size=0.5, random_state=26)
```

```
clf = DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

```
clf = clf.fit(X_train, y_train)
```

```
predictions = clf.predict(X_test)
```

- Thực hiện hàm hồi quy

- + Code:

```
import numpy as np
```

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

```
from sklearn.svm import SVC
```

```
svc = SVC()
```

```
svc.fit(X_train, y_train)
```

- Xây dựng hàm tính điểm chính xác giữa mô hình train và test

- + Code:

```
def compute_accuracy(Y_true, Y_pred):
```

```
    correctly_predicted = 0
```

```
    # lặp lại nhãn và kiểm tra độ chính xác
```

```
    for true_label, predicted in zip(Y_true, Y_pred):
```

```
        if true_label == predicted:
```

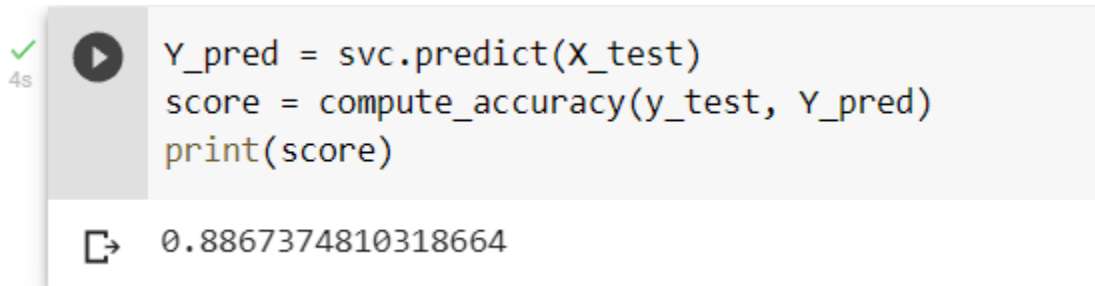
```
            correctly_predicted += 1
```

```
    # Tính toán điểm chính xác của việc so sánh
```

```
    accuracy_score = correctly_predicted / len(Y_true)
```

```
    return accuracy_score
```

- Hiển thị điểm chính xác của mô hình:



A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark and a play button icon, with '4s' indicating execution time. The code cell contains three lines of Python code: `Y_pred = svc.predict(X_test)`, `score = compute_accuracy(y_test, Y_pred)`, and `print(score)`. Below the code, the output is displayed as `0.8867374810318664`.

3.3 Đánh giá

- Sau khi thực hiện mô hình trên, nhóm nhận thấy điểm chính xác của mô hình là tương đối cao (0.8867374810318664)
- Tuy nhiên, dữ liệu huấn luyện còn hạn chế, nhóm cần tìm hiểu và thêm nhiều dữ liệu chính xác hơn để đảm bảo được tính đúng đắn của thuật toán đưa ra.
- Bên cạnh đó, trong quá trình thực hiện nhóm cũng chưa so sánh được độ chính xác giữa các thuật toán nhằm đưa ra tính hiệu quả của thuật toán ID3 cho bộ dữ liệu đang sử dụng.

3.4 Xây dựng cây quyết định

- Với bộ dữ liệu chuẩn bị quá lớn, khi dùng toàn bộ để vẽ cây quyết định sẽ dẫn đến một rừng cây và khó để biểu diễn thành hình ảnh nên nhóm quyết định lấy 25 dòng đầu tiên của bộ dữ liệu sử dụng cho phần này.
- Tiến hành vẽ cây:
 - + Thêm các thư viện cần thiết:

- Code:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px
```

+ Liên kết với bộ dataset (train_mini.csv):

- Code:

```
from google.colab import drive

drive.mount('/content/drive')

%cd /content/drive/MyDrive/ColabNotebooks/Datasets

df = pd.read_csv("train_mini.csv")
```

+ Tương tự như bộ dữ liệu đã dùng, ta chuẩn hóa các thuộc tính về dạng số.

+ Để tăng độ chính xác cũng như rút ngắn thời gian trong quá trình thực hiện, nhóm quyết định sử dụng thư viện `sklearn`, cụ thể là `sklearn.tree` để vẽ cây.

- Một số hàm cần dùng của thư viện:

```
import sklearn.tree as tr

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split
```

+ Ta chọn các thuộc tính ảnh hưởng đến đầu ra và dùng các hàm của `sklearn.tree` để tính toán:

- Code:

```
cotdt = ['age', 'job', 'marital', 'education', 'default',
'housing', 'loan', 'contact', 'month', 'day_of_week',
'poutcome']

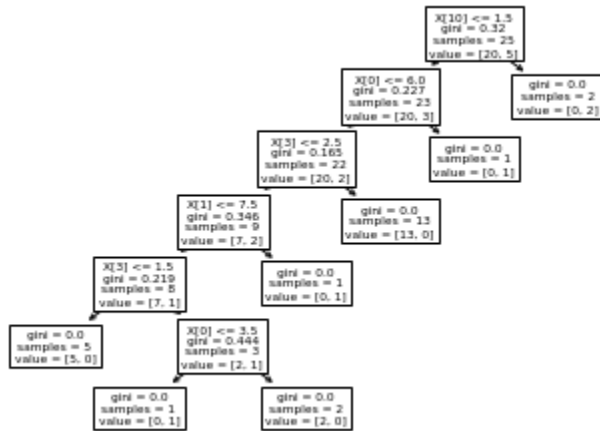
x = df[cotdt]

y = df['y'].astype('int')

dtree = tr.DecisionTreeClassifier()

dtree = dtree.fit(x, y)
```

+ Và cuối cùng là hiển thị thực thể ra cây một cách trực quan, ta sử dụng hàm `plot_tree()`, kết quả thu được một ảnh như sau:



TÀI LIỆU THAM KHẢO

- [1] Tổng quan về trí tuệ nhân tạo: lib.agu.edu.vn/TTNT.
- [2] PhD Vũ Hữu Tiếp, Machine Learning cơ bản: [Decision Trees \(1\): Iterative Dichotomiser 3](#).
- [3] Nguyễn Phương Nga, Giáo trình Trí tuệ nhân tạo, Trường Đại học Công nghiệp Hà Nội, [bản 2021](#).
- [4] <http://id3alg.altervista.org/>
- [5] J.R. Quinlan, C4.5 programs for machine learning Morgan Kaufmann Publishers, livres Google.
- [6] Nguyễn Nhật Quang, Tiền xử lý dữ liệu, [Khai phá dữ liệu](#).
- [7] https://www.javatpoint.com/accuracy_score-in-sklearn
- [8] Quách Xuân Nam, Trương Văn Thông, Nguyễn Đình Thanh: “[Decision Tree: ID3](#)”
- [9] Đỗ Mạnh Quang, Trần Thị Thảo Nguyên: [Decision-Tree-ID3](#)

[10] J.R. QUINLAN, Induction of Decision Trees, 1986, Machine Learning
1:81-106