**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: HỌC MÁY**

**ĐỀ TÀI: PHÁT HIỆN XÂM NHẬP BẰNG HỌC MÁY**

**Giáo viên hướng dẫn: Trần Hồng Việt**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** |
| **1** | **1671020010** | **Đặng Lê Hoàng Anh** | **23/02/2004** |
| **2** | **1671020228** | **Nguyễn Khôi Nguyên** | **29/12/2004** |

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: HỌC MÁY**

**ĐỀ TÀI: PHÁT HIỆN XÂM NHẬP BẰNG HỌC MÁY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | Điểm | |
| **Bằng Số** | **Bằng Chữ** |
| 1 | 1671020010 | Đặng Lê Hoàng Anh | **23/02/2004** |  |  |
| 2 | 1671020228 | Nguyễn Khôi Nguyên | **29/12/2004** |  |  |
| 3 |  |  |  |  |  |

### 

### CÁN BỘ CHẤM THI

**Hà Nội, 2024**

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ số hiện nay, việc bảo vệ an ninh mạng trở thành một trong những nhiệm vụ quan trọng hàng đầu của các tổ chức và cá nhân. Với sự gia tăng của các cuộc tấn công mạng, xâm nhập hệ thống đã trở thành mối đe dọa nghiêm trọng đối với dữ liệu và các dịch vụ quan trọng. Do đó, việc nghiên cứu và phát triển các hệ thống phát hiện xâm nhập (Intrusion Detection Systems - IDS) là điều cần thiết.

Trong bài tập lớn này, chúng tôi sẽ trình bày về một hệ thống phát hiện xâm nhập sử dụng các thuật toán học máy. Việc ứng dụng học máy trong lĩnh vực phát hiện xâm nhập không chỉ giúp tăng cường khả năng nhận diện các cuộc tấn công, mà còn cải thiện độ chính xác và hiệu quả của hệ thống. Học máy cho phép hệ thống tự động học hỏi từ dữ liệu, nhận diện các mẫu xâm nhập mới và phân biệt giữa các hoạt động hợp lệ và bất thường.

Mục tiêu của bài tập là phát triển một mô hình học máy có khả năng phát hiện các cuộc xâm nhập vào hệ thống thông qua việc phân tích dữ liệu mạng, từ đó xây dựng một hệ thống bảo mật chủ động và hiệu quả. Chúng tôi sẽ sử dụng các phương pháp học máy như phân loại, học sâu, và học tăng cường để đưa ra giải pháp cho vấn đề này.

Bài tập lớn này không chỉ mang ý nghĩa lý thuyết mà còn có tính ứng dụng cao, giúp sinh viên hiểu rõ hơn về an ninh mạng và vai trò quan trọng của học máy trong việc bảo vệ các hệ thống thông tin.

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 4](#_Toc179576052)

[**CHƯƠNG I. LÝ THUYẾT VỀ PHÁT HIỆN XÂM NHẬP BẰNG THUẬT TOÁN HỌC MÁY** 7](#_Toc179576053)

[**I. Giới thiệu:** 7](#_Toc179576054)

[1. Hệ thống phát hiện xâm nhập (IDS) 7](#_Toc179576055)

[2. Các phương pháp phát hiện xâm nhập 7](#_Toc179576056)

[3. Ứng dụng học máy trong IDS 8](#_Toc179576057)

[4. Các thách thức 9](#_Toc179576058)

[5. Các giải pháp cải tiến 10](#_Toc179576059)

[**CHƯƠNG II. XÂY DỰNG CHƯƠNG TRÌNH** 11](#_Toc179576060)

[**I. Phân tích và tiền xử lý dữ liệu** 11](#_Toc179576061)

[**II. Xây dựng và phân tích mô hình học máy:** 11](#_Toc179576062)

[**CHƯƠNG III. KẾT LUẬN** 24](#_Toc179576063)

# **CHƯƠNG I. LÝ THUYẾT VỀ PHÁT HIỆN XÂM NHẬP BẰNG THUẬT TOÁN HỌC MÁY**

## **I. Giới thiệu:**

Hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) là một thành phần quan trọng trong an ninh mạng, được thiết kế để giám sát và phân tích lưu lượng mạng nhằm phát hiện các hành vi bất thường hoặc độc hại có thể dẫn đến vi phạm bảo mật. Việc áp dụng thuật toán học máy (Machine Learning - ML) trong các hệ thống phát hiện xâm nhập đã trở nên phổ biến, mang lại nhiều ưu điểm trong việc phát hiện các cuộc tấn công mới và phức tạp hơn. Dưới đây là các lý thuyết cơ bản về hệ thống IDS sử dụng thuật toán học máy.

### 1. Hệ thống phát hiện xâm nhập (IDS)

Hệ thống IDS có hai dạng chính:

* **HIDS (Host-based IDS):** Giám sát hoạt động trên một máy tính cụ thể (host), bao gồm các tệp, quy trình và đăng nhập.
* **NIDS (Network-based IDS):** Giám sát lưu lượng mạng, tìm kiếm các mẫu hoặc hành vi bất thường có thể cho thấy xâm nhập.

## 2. Các phương pháp phát hiện xâm nhập

Có hai phương pháp chính để phát hiện xâm nhập:

* **Phát hiện dựa trên dấu hiệu (Signature-based Detection):** So sánh các sự kiện mạng với một tập hợp các dấu hiệu đã biết (mẫu tấn công). Phương pháp này hiệu quả cho các cuộc tấn công đã biết nhưng không có khả năng phát hiện các cuộc tấn công mới (zero-day attacks).
* **Phát hiện bất thường (Anomaly-based Detection):** Phát hiện các hành vi bất thường trong hệ thống, dựa trên mô hình hành vi chuẩn. Học máy được sử dụng phổ biến trong phương pháp này.

## 3. Ứng dụng học máy trong IDS

Thuật toán học máy giúp cải thiện khả năng phát hiện xâm nhập bằng cách tự động học và nhận diện các mẫu hành vi phức tạp và bất thường. Quá trình này bao gồm các bước sau:

#### 3.1. Thu thập dữ liệu

Dữ liệu được thu thập từ nhiều nguồn như lưu lượng mạng, các gói tin, hoặc hành vi người dùng. Dữ liệu thường bao gồm các đặc điểm như:

* Địa chỉ IP nguồn và đích.
* Cổng nguồn và cổng đích.
* Loại giao thức (TCP, UDP, ICMP).
* Kích thước gói tin.
* Tốc độ truyền dữ liệu, v.v.

#### 3.2. Tiền xử lý dữ liệu

Dữ liệu được tiền xử lý để làm sạch, loại bỏ nhiễu, và chuẩn hóa để phù hợp với thuật toán học máy. Các kỹ thuật như:

* **Xử lý giá trị thiếu:** Loại bỏ hoặc thay thế dữ liệu thiếu.
* **Biểu diễn đặc trưng:** Trích xuất và lựa chọn các đặc trưng quan trọng từ dữ liệu mạng.
* **Chuẩn hóa:** Chuẩn hóa giá trị để đưa các thuộc tính về cùng một thang đo.

#### 3.3. Thuật toán học máy cho IDS

Có nhiều thuật toán học máy được sử dụng trong IDS, bao gồm:

* **Học có giám sát (Supervised Learning):** Dựa vào tập dữ liệu đã gán nhãn (tấn công hoặc bình thường) để huấn luyện mô hình. Các thuật toán như: **Support Vector Machine (SVM), Random Forest, Naive Bayes, Decision Tree**.
* **Học không giám sát (Unsupervised Learning):** Phát hiện xâm nhập dựa trên việc tìm kiếm các cụm dữ liệu (clustering) hoặc phân phối bất thường mà không cần nhãn trước. Các thuật toán phổ biến: **K-means, DBSCAN, Isolation Forest**.
* **Học bán giám sát (Semi-supervised Learning):** Kết hợp dữ liệu có gán nhãn và không gán nhãn để huấn luyện mô hình.

#### 3.4. Huấn luyện và đánh giá mô hình

Mô hình học máy cần được huấn luyện và kiểm tra trên các tập dữ liệu có chứa cả hành vi bình thường và các loại tấn công khác nhau. Các chỉ số đánh giá mô hình phổ biến bao gồm:

* **Độ chính xác (Accuracy).**
* **Độ nhạy (Recall) và độ đặc hiệu (Specificity).**
* **F1-Score:** Là sự kết hợp giữa độ chính xác và độ nhạy.
* **Tỷ lệ phát hiện dương tính giả (False Positive Rate).**

#### 3.5. Phát hiện thời gian thực

Các mô hình học máy trong IDS có thể được triển khai để phát hiện xâm nhập trong thời gian thực bằng cách liên tục theo dõi lưu lượng mạng và so sánh với mô hình đã được huấn luyện. Khi phát hiện hành vi bất thường, hệ thống sẽ kích hoạt cảnh báo để quản trị viên xử lý.

## 4. Các thách thức

* **Dữ liệu mất cân bằng:** Các cuộc tấn công hiếm khi xảy ra so với hành vi bình thường, dẫn đến dữ liệu mất cân bằng, gây khó khăn cho việc huấn luyện mô hình.
* **Khả năng mở rộng:** IDS cần xử lý lượng lớn dữ liệu mạng trong thời gian thực, đòi hỏi hệ thống phải có khả năng mở rộng và hiệu suất cao.
* **Tính chính xác:** Phát hiện nhầm lẫn (false positive) có thể dẫn đến cảnh báo không cần thiết, làm gián đoạn hoạt động bình thường.

## 5. Các giải pháp cải tiến

Để nâng cao hiệu quả, một số giải pháp cải tiến bao gồm:

* **Deep Learning:** Sử dụng các mô hình học sâu như RNN, LSTM, và CNN để xử lý dữ liệu mạng phức tạp hơn.
* **Hybrid IDS:** Kết hợp giữa phát hiện dựa trên dấu hiệu và bất thường để tận dụng ưu điểm của cả hai phương pháp.

Hệ thống IDS kết hợp học máy mang lại tiềm năng lớn trong việc phát hiện các mối đe dọa mạng hiện đại, tuy nhiên cần liên tục cải tiến để bắt kịp các xu hướng tấn công ngày càng tinh vi.

# **CHƯƠNG II. XÂY DỰNG CHƯƠNG TRÌNH**

## **I. Phân tích và tiền xử lý dữ liệu**

**- Mục tiêu:**

* Xác định rõ các yêu cầu và mục tiêu của chương trình phát hiện xâm nhập.
* Xây dựng một hệ thống học máy có khả năng phát hiện xâm nhập vào mạng hoặc hệ thống dựa trên các hành vi bất thường hoặc dấu hiệu tấn công.
* Đánh giá hiệu suất và độ chính xác của mô hình phát hiện xâm nhập được triển khai.

**- Thu thập và tiền xử lý dữ liệu:**

* **Làm sạch dữ liệu:** Xử lý các giá trị bị thiếu, loại bỏ các dữ liệu nhiễu từ lưu lượng mạng hoặc nhật ký hệ thống.
* **Biến đổi dữ liệu:** Chuyển đổi dữ liệu mạng hoặc các thông tin log thành các dạng đặc trưng mà mô hình học máy có thể hiểu được (vector hóa các thuộc tính, chuẩn hóa dữ liệu).
* **Phân loại nhãn:** Đánh nhãn dữ liệu là "tấn công" hoặc "bình thường" để phục vụ cho quá trình huấn luyện mô hình phát hiện xâm nhập.

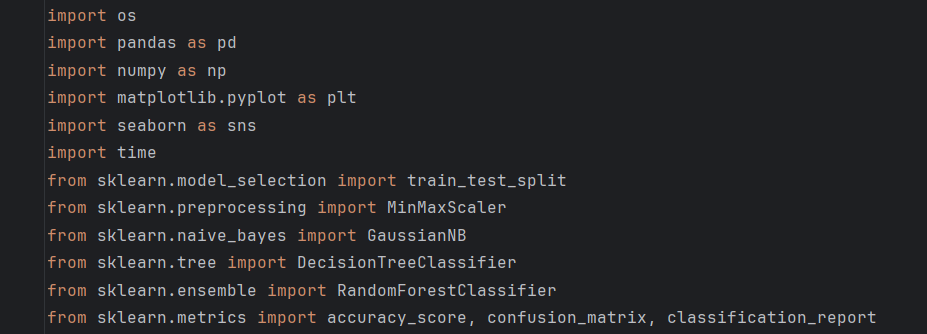
## **II. Xây dựng và phân tích mô hình học máy:**

**1. Lựa chọn và chuẩn bị dữ liệu:**

**1.1. Mô tả dữ liệu:**

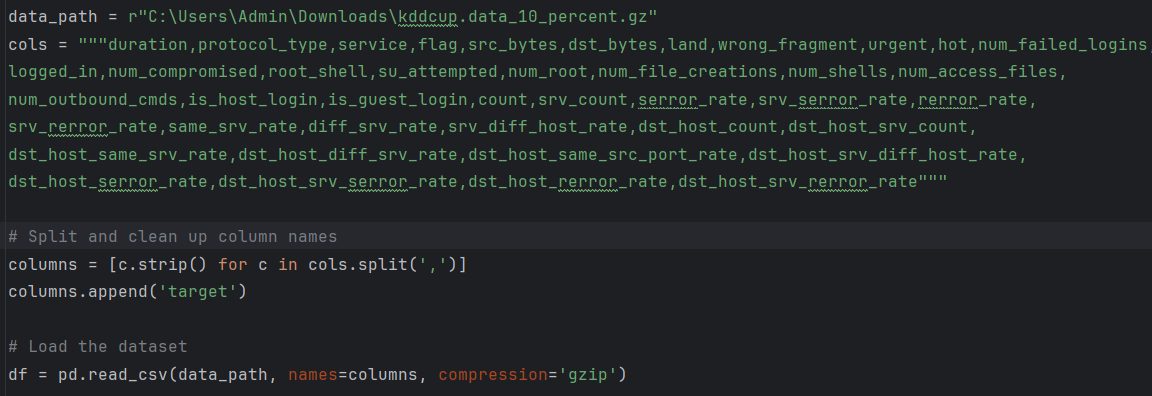
**1.1.1. Import các thư viện cần thiết:**

* Sử dụng các thư viện như pandas, numpy, matplotlib, seaborn, sklearn để phân tích dữ liệu và huấn luyện các mô hình.



**1.1.2. Xử lý dữ liệu:**

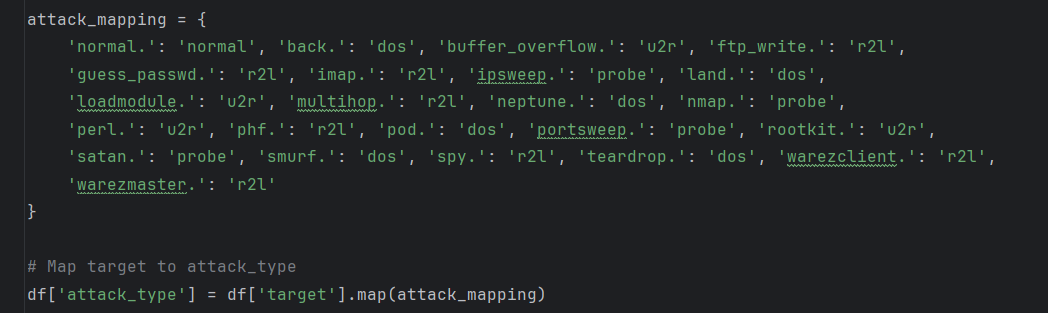
* **Tải dữ liệu** từ đường dẫn file và thêm tên cột cho dataset.
* **Code:**

****

* Ở đây, biến data\_path lưu trữ đường dẫn đến tệp dữ liệu nén (gzip) có tên là kddcup.data\_10\_percent.gz. Dấu r trước chuỗi cho biết rằng đây là chuỗi "thô" (raw string), trong đó các ký tự đặc biệt như \ sẽ không được xử lý.
* Biến cols chứa danh sách các tên cột được phân tách bằng dấu phẩy. Các tên cột này đại diện cho các đặc điểm của dữ liệu trong tệp.

**1.1.3. Ánh xạ nhãn tấn công vào các loại lớn hơn:**

* Các nhãn tấn công cụ thể được ánh xạ thành các loại lớn như: normal, dos, u2r, r2l, và probe.
* **Code:**

****

**-** Từ điển attack\_mapping ánh xạ các tên loại tấn công từ dữ liệu (chẳng hạn như normal., back., buffer\_overflow.) sang một nhãn dễ hiểu hơn (như 'normal', 'dos', 'u2r', 'r2l', và 'probe').

- Các giá trị trong từ điển chỉ ra loại tấn công mà một bản ghi cụ thể thuộc về. Ví dụ, các giá trị 'normal.' sẽ được ánh xạ thành 'normal', và các giá trị 'back.' sẽ được ánh xạ thành 'dos'.

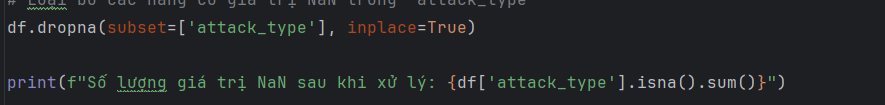
****

**-**  Đoạn mã này sử dụng phương thức map() để ánh xạ giá trị trong cột target của DataFrame df theo từ điển attack\_mapping.

- Mỗi giá trị trong cột target sẽ được tìm kiếm trong từ điển. Nếu tìm thấy, giá trị tương ứng sẽ được gán cho cột mới attack\_type. Nếu không tìm thấy, giá trị sẽ trở thành NaN (Not a Number).

**1.1.4. Xử lý các giá trị NaN và loại bỏ các cột không cần thiết:**

* Kiểm tra và loại bỏ các hàng có giá trị NaN sau khi ánh xạ.
* Loại bỏ các cột không hữu ích như service.
* **Code:**

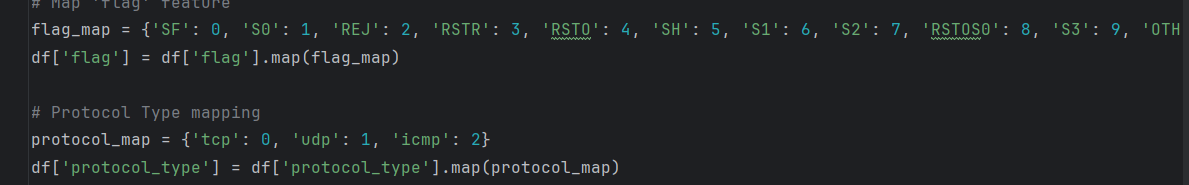
****

**-** Tham số này chỉ định rằng chỉ cần kiểm tra cột attack\_type để tìm các giá trị NaN. Nếu một hàng có giá trị NaN trong cột này, nó sẽ bị xóa khỏi DataFrame.

**-** Đoạn mã này đảm bảo rằng tất cả các hàng trong DataFrame df đều có giá trị hợp lệ cho cột attack\_type. Nếu không có giá trị hợp lệ, các hàng đó sẽ bị loại bỏ, giúp giữ lại chỉ các bản ghi có thông tin đầy đủ cho phân tích tiếp theo.

**1.1.5. Ánh xạ các cột categorical khác:**

* Ánh xạ các giá trị categorical khác như cột flag và protocol\_type
* **Code:**

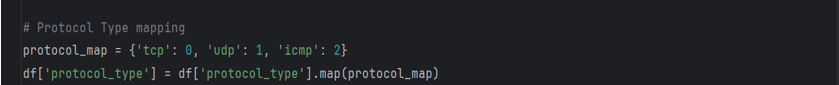
****

* flag\_map là một từ điển ánh xạ các giá trị ký hiệu trạng thái (flag) sang các giá trị số nguyên. Mỗi ký hiệu được ánh xạ tới một số nguyên duy nhất:

Ví dụ: 'SF' được ánh xạ thành 0, 'S0' thành 1, và tiếp tục như vậy cho các ký hiệu khác.

**Ánh xạ giá trị trong DataFrame**:

* df['flag'].map(flag\_map) sẽ áp dụng ánh xạ này lên cột flag của DataFrame df. Mỗi giá trị trong cột flag sẽ được thay thế bằng giá trị số tương ứng từ flag\_map.
* Nếu một giá trị không có trong flag\_map, nó sẽ trở thành NaN.



**Tạo từ điển ánh xạ**:

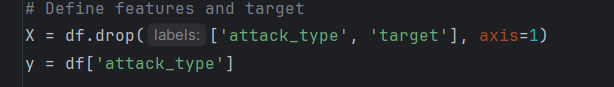
* protocol\_map là một từ điển ánh xạ các loại giao thức (protocol\_type) sang các giá trị số nguyên. Cụ thể:
* 'tcp' được ánh xạ thành 0, 'udp' thành 1, và 'icmp' thành 2.

**Ánh xạ giá trị trong DataFrame**:

* Tương tự như trên, df['protocol\_type'].map(protocol\_map) áp dụng ánh xạ cho cột protocol\_type, thay thế các giá trị phân loại bằng các giá trị số tương ứng.

**1.1.6. Chia tách dữ liệu thành tập đặc trưng và nhãn (features và target):**

* Xác định X là các đặc trưng và y là nhãn (attack\_type).
* **Code:**

****

****

**-**  **df.drop(['attack\_type', 'target'], axis=1)**:

- Phương thức drop() được sử dụng để loại bỏ các cột trong DataFrame df. Ở đây, các cột 'attack\_type' và 'target' được chỉ định để loại bỏ.

**- axis=1**: Tham số này cho biết rằng chúng ta đang xóa các cột (chứ không phải hàng). Nếu axis=0, thì các hàng sẽ bị xóa.

- Biến X sẽ chứa tất cả các cột còn lại trong DataFrame df sau khi đã loại bỏ cột 'attack\_type' và 'target'. Điều này có nghĩa là X sẽ chứa tất cả các biến độc lập (features) mà bạn sẽ sử dụng để xây dựng mô hình.

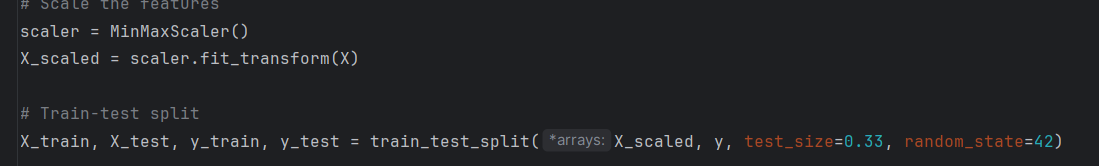
- **df['attack\_type']**:

- Ở đây, biến y sẽ chứa giá trị của cột attack\_type, tức là nhãn mục tiêu mà bạn muốn dự đoán.

- Biến y sẽ được sử dụng trong quá trình huấn luyện mô hình học máy để xác định loại tấn công dựa trên các đặc trưng trong biến X.

**1.1.7. Chuẩn hóa dữ liệu và chia tập huấn luyện - kiểm tra:**

* Dữ liệu được chuẩn hóa bằng MinMaxScaler để đảm bảo các đặc trưng có cùng thang đo.
* **Code:**



****

**MinMaxScaler()**:

* MinMaxScaler là một lớp trong thư viện sklearn.preprocessing được sử dụng để chuẩn hóa dữ liệu.
* Phương pháp chuẩn hóa này sẽ đưa tất cả các giá trị của biến vào khoảng [0, 1]. Cụ thể, mỗi giá trị xxx sẽ được chuyển đổi theo công thức: x′=x−minmax−minx' = \frac{x - \text{min}}{\text{max} - \text{min}}x′=max−minx−min​ trong đó min và max là giá trị nhỏ nhất và lớn nhất trong cột đó.

**scaler.fit\_transform(X)**:

* Phương thức fit\_transform() vừa "học" từ dữ liệu X (tính toán các giá trị tối thiểu và tối đa), vừa áp dụng phép chuẩn hóa cho dữ liệu.
* Kết quả là X\_scaled, một mảng numpy chứa các giá trị đã được chuẩn hóa trong khoảng [0, 1].



**train\_test\_split()**:

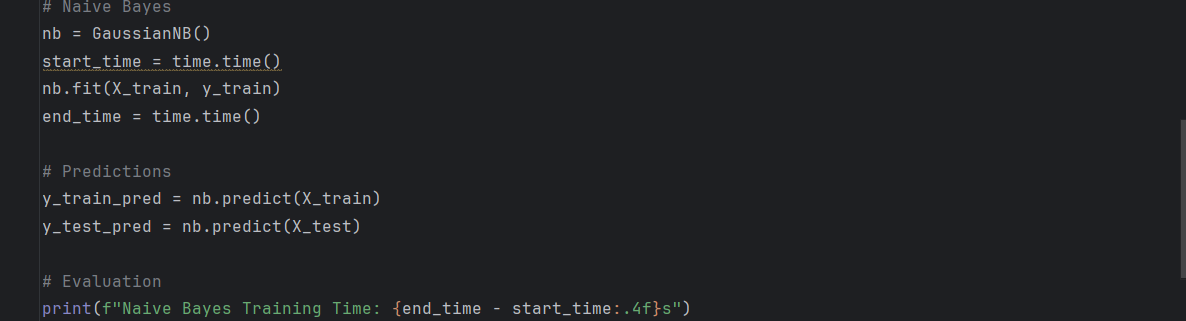
* Hàm này trong thư viện sklearn.model\_selection được sử dụng để chia dữ liệu thành hai tập: tập huấn luyện và tập kiểm tra.

**Các tham số**:

* **X\_scaled**: Đây là mảng chứa các đặc trưng đã được chuẩn hóa mà chúng ta muốn chia.
* **y**: Đây là mảng chứa nhãn mục tiêu tương ứng với các đặc trưng trong X\_scaled.
* **test\_size=0.33**: Tham số này chỉ định rằng 33% dữ liệu sẽ được dành cho tập kiểm tra. Ngược lại, 67% dữ liệu sẽ được sử dụng cho tập huấn luyện.
* **random\_state=42**: Tham số này thiết lập một giá trị ngẫu nhiên cụ thể để đảm bảo rằng việc chia dữ liệu có thể được lặp lại. Điều này có nghĩa là mỗi khi bạn chạy mã này với cùng một random\_state, bạn sẽ nhận được cùng một phân chia dữ liệu.

**1.1.8. Huấn luyện và đánh giá mô hình Naïve Bayes:**

* Huấn luyện mô hình Naive Bayes và đánh giá độ chính xác trên tập huấn luyện và kiểm tra.
* **Code:**

****

**GaussianNB()**:

* Đây là một lớp trong thư viện sklearn.naive\_bayes được sử dụng để xây dựng một mô hình phân loại Naive Bayes với giả định rằng các đặc trưng (features) theo phân phối Gaussian (hay còn gọi là phân phối chuẩn).
* Naive Bayes là một thuật toán phân loại dựa trên định lý Bayes, giả định rằng các đặc trưng là độc lập với nhau.

**fit(X\_train, y\_train)**:

* Phương thức fit() được sử dụng để huấn luyện mô hình Naive Bayes trên tập dữ liệu huấn luyện.
* X\_train là mảng chứa các đặc trưng đã được chuẩn hóa và y\_train là mảng chứa nhãn mục tiêu tương ứng.
* Trong bước này, mô hình sẽ học từ dữ liệu để xác định mối quan hệ giữa các đặc trưng và nhãn mục tiêu.

**predict(X\_train)**:

* Phương thức predict() được sử dụng để thực hiện dự đoán nhãn mục tiêu cho dữ liệu đầu vào.
* Ở đây, mô hình được sử dụng để dự đoán nhãn cho tập huấn luyện (X\_train). Kết quả dự đoán sẽ được lưu vào biến y\_train\_pred.
* Tương tự như trên, nhưng ở đây mô hình được sử dụng để dự đoán nhãn cho tập kiểm tra (X\_test). Kết quả sẽ được lưu vào biến y\_test\_pred.

**accuracy\_score(y\_test, y\_test\_pred)**:

* Hàm accuracy\_score trong thư viện sklearn.metrics được sử dụng để tính toán độ chính xác của mô hình, tức là tỷ lệ phần trăm các dự đoán đúng trong tập kiểm tra.

Tham số y\_test là nhãn mục tiêu thực tế và y\_test\_pred là nhãn mục tiêu mà mô hình dự đoán.

**print(f"Naive Bayes Test Accuracy: {accuracy\_score(y\_test, y\_test\_pred):.4f}")**:

* Dòng lệnh này sẽ in ra độ chính xác của mô hình trên tập kiểm tra với định dạng có 4 chữ số thập phân. Cụ thể, nó sẽ hiển thị một chuỗi cho biết độ chính xác của mô hình Naive Bayes trên tập kiểm tra.

**1.1.9. Huấn luyện và đánh giá mô hình Decision Tree:**

* Huấn luyện mô hình Decision Tree với tiêu chí "entropy" và giới hạn độ sâu là 4.
* **Code:**

**A computer screen shot of a program

Description automatically generated**

**DecisionTreeClassifier:  
-** Sử dụng thuật toán cây quyết định với độ đo entropy cùng với độ sâu tối đa là 4.

**fit(X\_train, y\_train):**

* Đo thời gian huân luyện trên tập dữ liệu X\_train và y\_train.

**Prediction:**

* Dự đoán trên cả tập huấn luyện (y\_train\_pred) và tập kiểm tra (y\_test\_pred).

**Evaluation:**

* Dòng lệnh này sẽ in ra thời gian huấn luyện, độ chính xác trên cả tập huấn luyện và kiểm tra.
* Xuất báo cáo phân loại chi tiết với các chỉ số như precision, recall, và F1-score.

**1.1.10. Huấn luyện và đánh giá mô hình Random Forest**

* Huấn luyện mô hình Random Forest với 100 cây quyết định.
* **Code:**

A computer screen shot of a program code

Description automatically generated

**RandomForestClassifier:**

- Thuật toán Random Forest cho 100 cây quyết định

**fit(X\_train, y\_train)**

**-** Đo thời gian huấn luyện trên tập X\_train và y\_train

**Prediction:**

**-** Dự đoán trên cả tập huấn luyện (y\_train\_pred) và tập kiểm tra (y\_test\_pred).

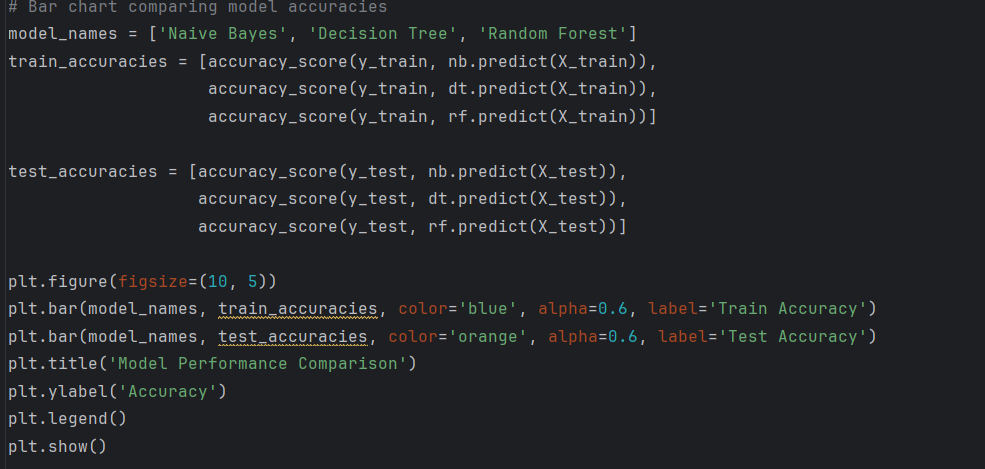
**Evaluation:**

**-** Dòng lệnh này sẽ đánh giá độ chính xác của mô hình (train và test accuracy.

- Xuất báo cáo phân loại (classification\_report) với các chỉ số như precision, recall, F1-score.

**1.1.11. So sánh các mô hình bằng biểu đồ:**

* Biểu đồ so sánh độ chính xác của các mô hình trên tập huấn luyện và kiểm tra.
* **Code:**

****

**model\_names**:

* Đây là một danh sách chứa tên của ba mô hình mà bạn đang so sánh: Naive Bayes, Decision Tree, và Random Forest. Danh sách này sẽ được sử dụng làm nhãn cho các cột trong biểu đồ.

**train\_accuracies**:

* Danh sách này chứa độ chính xác của từng mô hình trên tập huấn luyện.
* accuracy\_score(y\_train, nb.predict(X\_train)): Tính độ chính xác cho mô hình Naive Bayes trên tập huấn luyện.
* accuracy\_score(y\_train, dt.predict(X\_train)): Tính độ chính xác cho mô hình Decision Tree trên tập huấn luyện.
* **accuracy\_score(y\_train, rf.predict(X\_train)):**
* Tính độ chính xác cho mô hình Random Forest trên tập huấn luyện.

**test\_accuracies**:

* Tương tự như trên, nhưng ở đây là độ chính xác của từng mô hình trên tập kiểm tra.
* accuracy\_score(y\_test, nb.predict(X\_test)): Tính độ chính xác cho mô hình Naive Bayes trên tập kiểm tra.
* accuracy\_score(y\_test, dt.predict(X\_test)): Tính độ chính xác cho mô hình Decision Tree trên tập kiểm tra.
* accuracy\_score(y\_test, rf.predict(X\_test)): Tính độ chính xác cho mô hình Random Forest trên tập kiểm tra.

**plt.figure(figsize=(10, 5))**:

* Tạo một figure (hình) mới với kích thước 10x5 inches.

**plt.bar()**:

* Vẽ biểu đồ cột:
  + Cột đầu tiên là độ chính xác của các mô hình trên tập huấn luyện (màu xanh lam).
  + Cột thứ hai là độ chính xác của các mô hình trên tập kiểm tra (màu cam).
* Tham số alpha điều chỉnh độ trong suốt của các cột (0.6 có nghĩa là 60% độ trong suốt).

**plt.title('Model Performance Comparison')**:

* Thiết lập tiêu đề cho biểu đồ.

**plt.ylabel('Accuracy')**:

* Thiết lập nhãn cho trục y (độ chính xác).

**plt.legend()**:

* Hiện thị chú thích cho các cột trong biểu đồ, giúp người xem dễ dàng nhận biết màu sắc tương ứng với tập huấn luyện hay kiểm tra.

**plt.show()**:

* Hiển thị biểu đồ.:

A screenshot of a computer

Description automatically generated

# **CHƯƠNG III. KẾT LUẬN**

Trong đề tài này, chúng tôi đã sử dụng các mô hình học máy bao gồm Naive Bayes, Decision Tree, và Random Forest để giải quyết bài toán phát hiện xâm nhập dựa trên tập dữ liệu KDD Cup 1999. Tập dữ liệu chứa các hành vi mạng khác nhau, được phân thành các loại tấn công và hành vi bình thường. Sau khi tiền xử lý dữ liệu, chúng tôi đã huấn luyện các mô hình học máy để phân loại các mẫu mạng.

Các bước chính trong quá trình thực hiện bao gồm:

* Tiền xử lý dữ liệu: Loại bỏ các cột không cần thiết, ánh xạ các giá trị tấn công thành các nhóm chính (DoS, Probe, U2R, R2L, và Normal), chuyển đổi các thuộc tính dạng chuỗi thành số.
* Chia tách dữ liệu: Sử dụng train\_test\_split để chia dữ liệu thành tập huấn luyện và tập kiểm thử.
* Huấn luyện mô hình: Huấn luyện các mô hình Naive Bayes, Decision Tree, và Random Forest.
* Đánh giá mô hình: Sử dụng các thước đo như độ chính xác (Accuracy), và báo cáo phân loại (Classification Report) để đánh giá hiệu suất của mô hình trên cả tập huấn luyện và tập kiểm thử.

Từ đó thấy dược rằng:

* **Random Forest** có hiệu suất cao nhất với độ chính xác tốt trên cả tập huấn luyện và tập kiểm thử, nhờ khả năng tổng hợp dự đoán từ nhiều cây quyết định.
* **Decision Tree** có độ chính xác thấp hơn một chút so với Random Forest, nhưng vẫn cho kết quả tốt, đặc biệt là với việc sử dụng độ sâu hợp lý.
* **Naive Bayes** có tốc độ huấn luyện nhanh nhưng độ chính xác thấp hơn so với các mô hình khác, do sự giả định đơn giản của nó về tính độc lập của các thuộc tính.

**TÀI LIỆU THAM KHẢO**