

**PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

**----------------**



**BÁO CÁO MÔN KHAI PHÁ DỮ LIỆU**

**Tên đề tài:**

**Phân loại lưu lượng truy cập Internet trên tường lửa của các trường đại học**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | ***Vũ Thị Hạnh*** |
| **Sinh viên thực hiện:** | ***Trần Ngọc Hải - 2051067527*** |
|  | ***Cao Hỷ Khang - 2051067547*** |
|  | ***Lê Trường Duy - 2051067559*** |
| **Lớp:** | ***S23 – 62TH*** |

**TP. Hồ Chí Minh, ngày 22 tháng 10 năm 2023**

Phụ lục

[Lời mở đầu 3](#_Toc149516091)

[DANH MỤC CÁC THUẬT NGỮ 4](#_Toc149516092)

[Chương I: XÁC ĐỊNH VÀ PHÂN TÍCH YÊU CẦU 5](#_Toc149516093)

[1. Giới thiệu chung 5](#_Toc149516094)

[2. Xác định yêu cầu 6](#_Toc149516095)

[3. Dự kiến kết quả đạt được 7](#_Toc149516096)

[4. Công cụ thực hiện đồ án 7](#_Toc149516097)

[Chương II: CƠ SỞ LÝ THUYẾT 8](#_Toc149516098)

[1. Tổng quan môn học 8](#_Toc149516099)

[2. Thuật toán Random Forest 9](#_Toc149516100)

[3. Thuật toán Cây quyết định 9](#_Toc149516101)

[4. Thuật toán Naive Bayes 10](#_Toc149516102)

[Chương III: MÔ HÌNH 11](#_Toc149516103)

[1. Mô tả các chức năng của tập dữ liệu 11](#_Toc149516104)

[2. Xử lý dữ liệu 12](#_Toc149516105)

[2.1 Xem thông tin cơ bản của bộ dữ liệu 14](#_Toc149516106)

[2.2 Tiền Xử Lý Dữ Liệu 15](#_Toc149516107)

[2.3 Chuẩn hóa dữ liệu 22](#_Toc149516108)

[2.4 Huấn luyện mô hình 25](#_Toc149516109)

[3. Thuật toán Decision Tree 26](#_Toc149516110)

[4. Thuật toán Random Forest 30](#_Toc149516111)

[5. Thuật toán Naive Bayes 34](#_Toc149516112)

[6. Nhận xét các thuật toán và test với bộ data mới 36](#_Toc149516113)

[Chương IV: KẾT QUẢ ĐẠT ĐƯỢC 41](#_Toc149516114)

[Chương V: TÀI LIỆU THAM KHẢO 41](#_Toc149516115)

# Lời mở đầu

Với sự phát triển mạnh mẽ của Internet và nhu cầu truy cập thông tin ngày càng cao, các trường đại học hiện nay đều cho phép sinh viên truy cập Internet qua mạng LAN của nhà trường. Tuy nhiên, việc sử dụng Internet cũng tiềm ẩn nhiều nguy cơ về lừa đảo, tin tặc và các nội dung đồi trụy, không lành mạnh. Do đó, công tác giám sát và quản lý lưu lượng truy cập Internet qua tường lửa là vô cùng quan trọng.

Trong đề tài này, chúng em sẽ đi sâu vào việc phân loại lưu lượng Internet đi qua tường lửa của các trường Đại học bằng cách sử dụng các kỹ thuật Data Mining. Chúng em sử dụng các thuộc tính như Cổng nguồn, Cổng đích, Cổng nguồn NAT, Cổng đích NAT, Byte, Gói, Thời gian, pkts\_sent, và pkts\_received,… để khám phá thông tin ẩn bên trong dữ liệu. Qua đó, phân tích thống kê các loại lưu lượng, từ đó đề xuất các giải pháp tăng cường bảo mật và quản lý hiệu quả hơn việc sử dụng Internet lành mạnh trong trường học. Chúng em hy vọng rằng báo cáo này sẽ đánh dấu một bước đi quan trọng trong việc nghiên cứu và quản lý lưu lượng truy cập Internet tại trường đại học của chúng ta.

Đặc biệt, nhóm em gửi lời cảm ơn sâu sắc và chân thành nhất đến Giảng viên Vũ Thị Hạnh. Cô đã trực tiếp hướng dẫn tận tình, sửa chữa và đóng góp ý kiến quý báo giúp nhóm em hoàn thành tốt báo cáo đồ án của mình. Trong thời gian thực hiện đề tài, nhóm em đã vận dụng được những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới.

Xin chân thành cảm ơn các quý Thầy Cô !

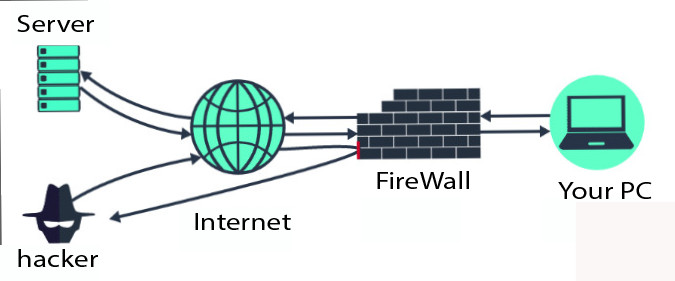
# DANH MỤC CÁC THUẬT NGỮ

|  |  |  |
| --- | --- | --- |
| **Thuật ngữ viết tắt** | **Thuật ngữ đầy đủ** | **Ý nghĩa** |
| Firewall |  | Tường lửa |
| Source Port |  | Cổng nguồn |
| Destination Port |  | Cổng đích |
| NAT | Network Address Translation | Kỹ thuật ánh xạ bộ địa chỉ IP |
| NAT Source Port |  | Cổng nguồn sau khi chuyển đổi NAT |
| NAT Destination Port |  | Cổng đích sau khi chuyển đổi NAT |
| Action |  | Hành động mà tường lửa sau khi kiểm tra sẽ thực hiện |
| Bytes | Kích thước dữ liệu | Kích thước dữ liệu (số byte) |
| Bytes Sent | Kích thước dữ liệu gửi đi | Kích thước dữ liệu gửi đi |
| Bytes Received | Kích thước dữ liệu nhận được | Kích thước dữ liệu nhận được |
| Packets |  | Gói tin |
| Elapsed Time (sec) |  | Thời gian xử lý gói tin(giây) |
| pkts\_sent |  | Số gói tin đã gửi |
| pkts\_received |  | Số gói tin đã nhận |
| CNPM |  | Công nghệ phần mềm |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| NAT | Network Address Translation | Chuyển đổi Địa chỉ Mạng |
| traffic |  | Dữ liệu được truyền tải qua mạng |

# Chương I: XÁC ĐỊNH VÀ PHÂN TÍCH YÊU CẦU

## Giới thiệu chung

Những năm gần đây, trên Internet thường xuyên xảy ra tình trạng có rất nhiều kẻ với mưu đồ xấu, lợi dụng thời cơ để xâm nhập vào những máy tính có kết nối với mạng Internet. Sau hàng loạt vụ tấn công mạng diễn ra liên tiếp với quy mô lớn, vấn đề bảo mật máy tính chính là một trong những chủ đề đang được quan tâm nhiều hơn bao giờ hết. Để đảm bảo an toàn trên internet, bạn cần đặc biệt chú ý tường lửa (Firewall). Nhờ có chiếc rào chắn Firewall này sẽ giúp bạn tạo ra một bức tường thành kiên cố, nâng cao tính bảo mật cho toàn bộ hệ thống máy tính của bạn. Vậy tường lửa là gì?



Tường lửa (Firewall) là một hệ thống an ninh mạng, có thể dựa trên phần cứng hoặc phần mềm, sử dụng các quy tắc để kiểm soát traffic vào, ra khỏi hệ thống. Tường lửa hoạt động như một rào chắn giữa mạng an toàn và mạng không an toàn. Nó kiểm soát các truy cập đến nguồn lực của mạng thông qua một mô hình kiểm soát chủ động. Nghĩa là, chỉ những traffic phù hợp với chính sách được định nghĩa trong tường lửa mới được truy cập vào mạng, mọi traffic khác đều bị từ chối.

Bất kì máy tính nào kết nối tới Internet cũng cần có firewall, giúp quản lý những gì được phép vào mạng và những gì được phép ra khỏi mạng. Việc có một “người gác cổng” như vậy để giám sát mọi việc xảy ra rất quan trọng bởi 2 lý do:

Thứ nhất, bất kì máy tính kết nối mạng nào thường kết nối vĩnh viễn với Internet. Thứ 2, mỗi máy tính trực tuyến lại có một chữ ký điện tử riêng, được gọi là Internet Protocol address (hay còn gọi là địa chỉ IP): Nếu không có firewall hỗ trợ, nó chẳng khác gì chuyện bạn bật tất cả đèn lên và mở rộng cửa để đón trộm vào.

Một firewall được cấu hình chính xác sẽ ngăn chặn điều này xảy ra và giúp máy tính “ẩn” một cách hiệu quả, cho phép người dùng thoải mái thưởng thức những gì thế giới trực tuyến mang lại. Firewall không giống chương trình diệt virus. Thay vào đó, nó làm việc cùng với những công cụ này nhằm đảm bảo rằng máy tính được bảo vệ từ hầu hết các mối tấn công nguy hại phổ biến.

Đề tài phân loại và phân tích lưu lượng truy cập Internet trên tường lửa của trường đại học nhằm mục đích nghiên cứu và phân loại các thông tin lưu lượng đi qua tường lửa của mạng máy tính trong trường đại học.

Với số lượng lớn sinh viên và nhu cầu truy cập Internet ngày càng tăng, việc giám sát và phân tích lưu lượng mạng là hết sức cần thiết để đảm bảo an toàn thông tin cho nhà trường. Do đó, đề tài đề xuất sử dụng các kỹ thuật học máy và phân loại dữ liệu để tự động nhận diện các luồng lưu lượng, phân loại thành lưu lượng bình thường, đáng ngờ hoặc nguy hiểm.

Kết quả của đề tài sẽ giúp nhà trường có cái nhìn sâu sắc về thống kê và xu hướng lưu lượng Internet, từ đó có chiến lược quản lý và đảm bảo an ninh mạng hợp lý, phù hợp.

## Xác định yêu cầu

* Chuẩn bị dữ liệu: tìm data set, tìm hiểu đề tài.
* Tiền xử lý dữ liệu: loại bỏ dữ liệu không cần thiết, xử lý dữ liệu bị thiếu, nhiễu, chuẩn hóa dữ liệu về cùng một domain.
* Chuẩn hóa mô hình: điều chỉnh giá trị phù hợp với mô hình.
* Xây dựng các mô hình phân loại dữ liệu sử dụng các thuật toán để phân loại lưu lượng thành các nhãn: Allow (cho phép), Deny (từ chối), Drop (loại bỏ), Reset-Both (đặt lại kết nối).
* Huấn luyện mô hình để chạy thuật toán
* Đánh giá các mô hình phân loại và chọn ra mô hình phù hợp nhất dựa trên các tiêu chí: độ chính xác, độ nhạy, giá trị trung bình.
* Phân tích các kết quả dự đoán của mô hình, thống kê tỷ lệ các loại lưu lượng.
* Test trên bộ dữ liệu mới.
* Đề xuất các giải pháp nhằm tối ưu hóa việc quản lý và lọc lưu lượng Internet trong trường đại học dựa trên kết quả.

## Dự kiến kết quả đạt được

* Thu thập được dữ liệu.
* Hoàn thành công đoạn tiền xử lý dữ liệu.
* Xây dựng được các mô hình.
* Mô hình đạt khoảng chính xác cao.
* Phân tích và thông kê được tỷ lệ truy cập.
* Test trên bộ data mới có độ chính xác cao
* Đề xuất.

## Công cụ thực hiện đồ án



* Nhóm tụi em dùng **Colaboratory** hay còn gọi là **Google Colab** để thực hiện đồ án.
* Google Colab là một sản phẩm từ Google Research, nó cho phép chạy các dòng code python thông qua trình duyệt, đặc biệt phù hợp với Data analysis, machine learning, data mining và giáo dục.

# Chương II: CƠ SỞ LÝ THUYẾT

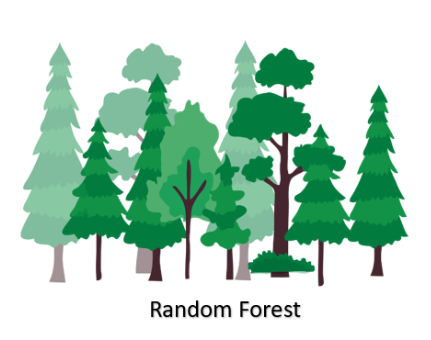
## Tổng quan môn học

Khoa học dữ liệu (Data Science) là một lĩnh vực nghiên cứu và ứng dụng dữ liệu để hiểu, phân tích, dự đoán và tạo ra giá trị từ thông tin chứa trong dữ liệu. Khoa học dữ liệu sử dụng kết hợp các phương pháp từ nhiều lĩnh vực như thống kê, toán học, khoa học máy tính, và nguồn dữ liệu thực tế để trích xuất thông tin hữu ích và kiến thức từ dữ liệu.

Khai phá dữ liệu (Data Mining) là quá trình tự động hoặc bán tự động tìm kiếm thông tin có giá trị từ tập dữ liệu lớn. Mục tiêu chính của khai phá dữ liệu là khám phá các mẫu, quy luật, sự kết nối, và thông tin ẩn trong dữ liệu mà có thể không dễ thấy bằng cách thường xuyên.

Tóm lại, khoa học dữ liệu là một lĩnh vực tổng hợp hơn, bao gồm cả việc khai phá dữ liệu, trong khi khai phá dữ liệu tập trung chủ yếu vào việc khám phá dữ liệu một cách tự động để tìm ra thông tin thú vị và quy luật trong dữ liệu.

## Thuật toán Random Forest



Random Forest là thuật toán xây dựng nhiều cây quyết định phân loại/hồi quy độc lập. Các cây được xây dựng từ tập huấn luyện ngẫu nhiên kéo tầng và các biến đầu vào quan trọng nhất được chọn ngẫu nhiên.

Khi dự đoán, mỗi cây sẽ đưa ra kết quả riêng và kết quả chung sẽ là kết quả phổ biến nhất trong tất cả các cây. Random Forest có khả năng phân loại tốt trên cả dữ liệu cân bằng và dữ liệu không cân bằng. Thuật toán dễ triển khai và có các tham số quan trọng cần điều chỉnh là số cây và số biến được chọn tại mỗi nút cây. Điều này giúp giảm bớt sự phụ thuộc vào các biến đầu vào.

Random Forest có thể xác định được các biến quan trọng nhất thông qua việc đánh giá độ phụ thuộc của các cây vào mỗi biến.

## Thuật toán Cây quyết định



Cây quyết định (Decision Tree) là một mô hình học máy phân loại và dự đoán được sử dụng trong nhiều ứng dụng khác nhau. Nó là một biểu đồ cây có cấu trúc, mô phỏng quá trình ra quyết định hoặc dự đoán bằng cách sử dụng luật điều kiện và giá trị của các thuộc tính.

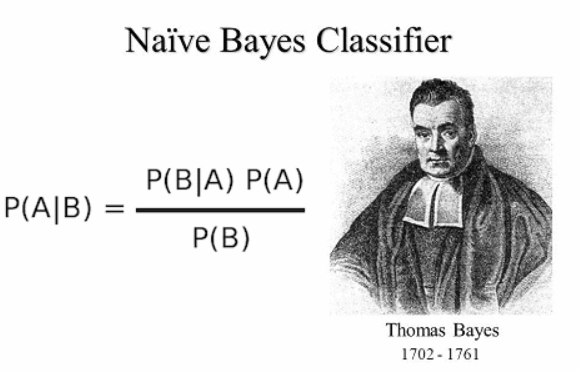
Một cây quyết định được xây dựng từ dữ liệu huấn luyện bằng cách phân chia tập dữ liệu thành các phần con (subsets) dựa trên các thuộc tính. Quá trình này tiếp tục cho đến khi mỗi phần con chỉ chứa các điểm dữ liệu thuộc cùng một lớp hoặc có giá trị dự đoán gần nhau. Các bước chính của thuật toán cây quyết định bao gồm:

- Chọn thuộc tính quan trọng: Chọn thuộc tính tốt nhất để chia dữ liệu tại mỗi nút cây.

- Xây dựng cây: Xây dựng cây quyết định bằng cách chia dữ liệu tại mỗi nút con thành các nút con.

- Dừng khi điều kiện thoả mãn: Cây ngừng phát triển khi một điều kiện dừng được đáp ứng, chẳng hạn như đạt đến độ sâu tối đa hoặc không còn thuộc tính để chia.

## Thuật toán Naive Bayes



Là một thuật toán máy học phân loại dựa trên nguyên tắc xác suất và lý thuyết xác suất Bayes. Ý tưởng cơ bản của thuật toán Naive Bayes là sử dụng các thông tin về xác suất để dự đoán xem một mẫu dữ liệu thuộc vào một lớp (hoặc nhãn) cụ thể nào. Đây là một phần của lý thuyết xác suất Bayes, mà công thức cơ bản được biểu thị như sau:

• P(A | B) = P(B | A) \* P(A) / P(B)

• P(A | B) là xác suất điều kiện của lớp A khi đã biết mẫu dữ liệu B.

• P(B | A) là xác suất của mẫu dữ liệu B khi đã biết lớp A.

• P(A) là xác suất tiên nghiệm của lớp A.

• P(B) là xác suất của mẫu dữ liệu B.

Thuật toán Naive Bayes có một số ưu điểm và nhược điểm đáng xem xét. Ưu điểm của nó bao gồm tính đơn giản và tốc độ huấn luyện nhanh, đặc biệt phù hợp cho dữ liệu nhỏ. Nó cũng hiệu quả trong việc phân loại. Tuy nhiên, Naive Bayes có nhược điểm khi không thể xử lý tốt dữ liệu có đội đối, và giả định về tính độc lập của các đặc trưng có thể không phù hợp với thực tế. Ngoài ra, thuật toán này đòi hỏi ước tính các xác suất từ dữ liệu, và kết quả có thể bị ảnh hưởng khi dữ liệu đào tạo không đủ hoặc có sự thiên vị.

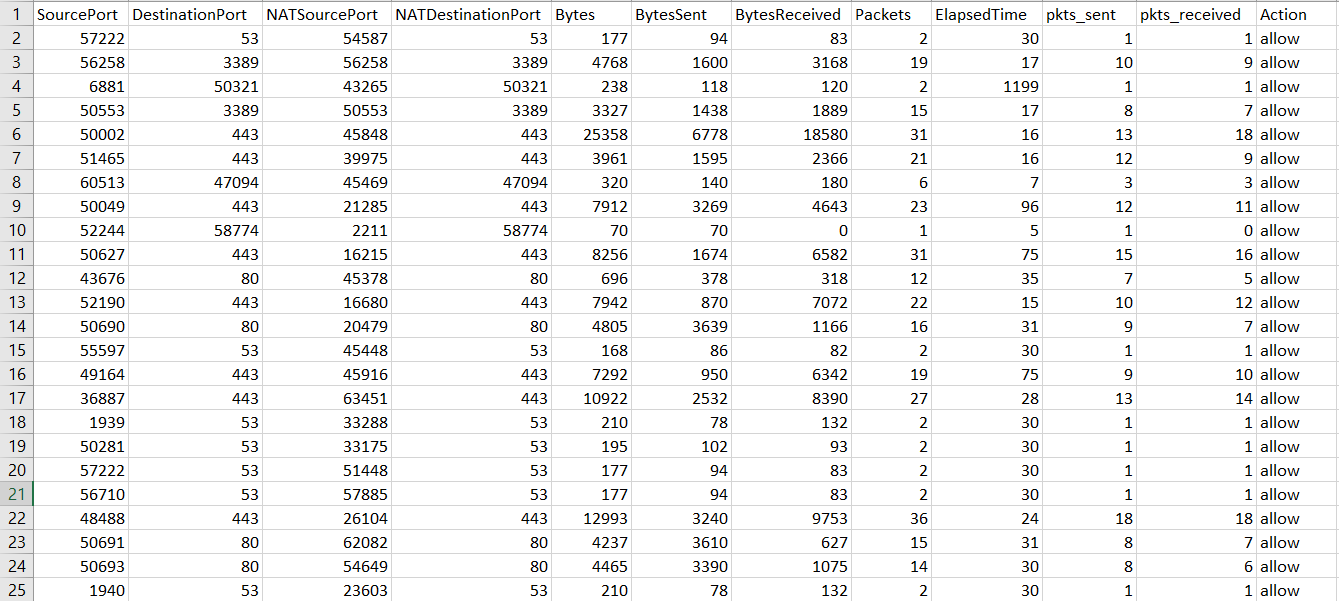
# Chương III: MÔ HÌNH

## Mô tả các chức năng của tập dữ liệu

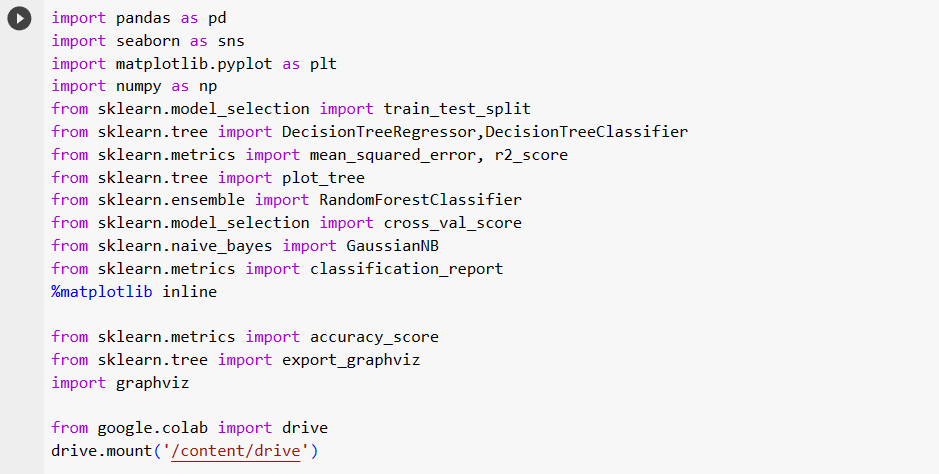
* *Source Port và Destination Port*: Các cổng nguồn và cổng đích thường được sử dụng trong giao thức truyền thông mạng để xác định nguồn và đích của gói tin mạng. Tương quan giữa chúng có thể giúp xác định loại kết nối mạng (ví dụ: dịch vụ, ứng dụng) đang được sử dụng.
* *NAT Source Port và NAT Destination Port*: Các trường này có thể liên quan đến quá trình Network Address Translation (NAT), trong đó địa chỉ IP hoặc cổng của gói tin được thay đổi để đi qua một bộ định tuyến hoặc tường lửa. Tương quan giữa chúng có thể giúp theo dõi các kết nối NAT.
* *Action:* Trường Action thường chỉ hành động được thực hiện trên gói tin mạng, ví dụ: chấp nhận (accept), từ chối (deny), hoặc thả (drop). Tương quan với các trường khác có thể giúp bạn hiểu cách gói tin được xử lý.
* *Bytes, Bytes Sent, và Bytes Received:* Các trường này liên quan đến lưu lượng dữ liệu truyền qua kết nối mạng. Bytes Sent thường chỉ lượng dữ liệu gửi đi, trong khi Bytes Received chỉ lượng dữ liệu nhận được. Tương quan giữa chúng có thể giúp bạn đánh giá tải trọng của kết nối.
* *Packets, pkts\_sent, và pkts\_received:* Các trường này liên quan đến số lượng gói tin truyền qua kết nối mạng. Packets thường chỉ tổng số gói tin, pkts\_sent chỉ số lượng gói tin gửi đi, và pkts\_received chỉ số lượng gói tin nhận được.
* *Elapsed Time (sec):* Trường này thường liên quan đến thời gian mà kết nối mạng hoặc hoạt động đang diễn ra.

## Xử lý dữ liệu

* Đây là bộ dữ liệu thô cần xử lý:



* Khai báo thư viện các thư viện cần thiết và đọc dữ liệu từ google drive



- Pandas (import pandas as pd): Thư viện pandas được sử dụng để làm việc với dữ liệu dưới dạng bảng, thực hiện các phân tích dữ liệu và xử lý dữ liệu dễ dàng hơn.

- Seaborn (import seaborn as sns): Seaborn là một thư viện trực quan hóa dữ liệu. Nó giúp bạn tạo các biểu đồ và đồ thị để hiểu dữ liệu dễ dàng hơn.

- Matplotlib.pyplot (import matplotlib.pyplot as plt): Thư viện matplotlib được sử dụng để tạo biểu đồ và đồ thị tùy chỉnh. Seaborn thường được kết hợp với Matplotlib để tạo biểu đồ đẹp hơn.

- Numpy (import numpy as np): NumPy là một thư viện toán học cơ bản trong Python, được sử dụng để thực hiện các phép toán số học và xử lý dữ liệu nhanh chóng.

-Sklearn.model\_selection(from sklearn.model\_selection import train\_test\_split): Thư viện Scikit-learn (sklearn) cung cấp các công cụ cho việc huấn luyện mô hình học máy. Trong trường hợp này, train\_test\_split được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra.

- sklearn.tree (from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier): Thư viện Scikit-learn cung cấp các thuật toán cây quyết định cho cả việc dự đoán biến số liên tục (DecisionTreeRegressor) và việc phân loại (DecisionTreeClassifier).

- sklearn.metrics (from sklearn.metrics import mean\_squared\_error, r2\_score, accuracy\_score, classification\_report): Scikit-learn cung cấp các hàm để đánh giá hiệu suất của mô hình, chẳng hạn như mean squared error (MSE), r-squared (R^2), accuracy score và classification report.

- sklearn.ensemble (from sklearn.ensemble import RandomForestClassifier): Scikit-learn cung cấp thuật toán RandomForest, một loại ensemble learning, để tạo và đánh giá mô hình phân loại.

- sklearn.naive\_bayes (from sklearn.naive\_bayes import GaussianNB): Thư viện Scikit-learn cung cấp mô hình Naive Bayes, trong trường hợp này là Gaussian Naive Bayes, cho việc phân loại.

- sklearn.tree (from sklearn.tree import plot\_tree, export\_graphviz): Đây là một số công cụ liên quan đến cây quyết định. plot\_tree dùng để vẽ cây quyết định, export\_graphviz để tạo tệp dữ liệu DOT cho việc vẽ cây bằng Graphviz.

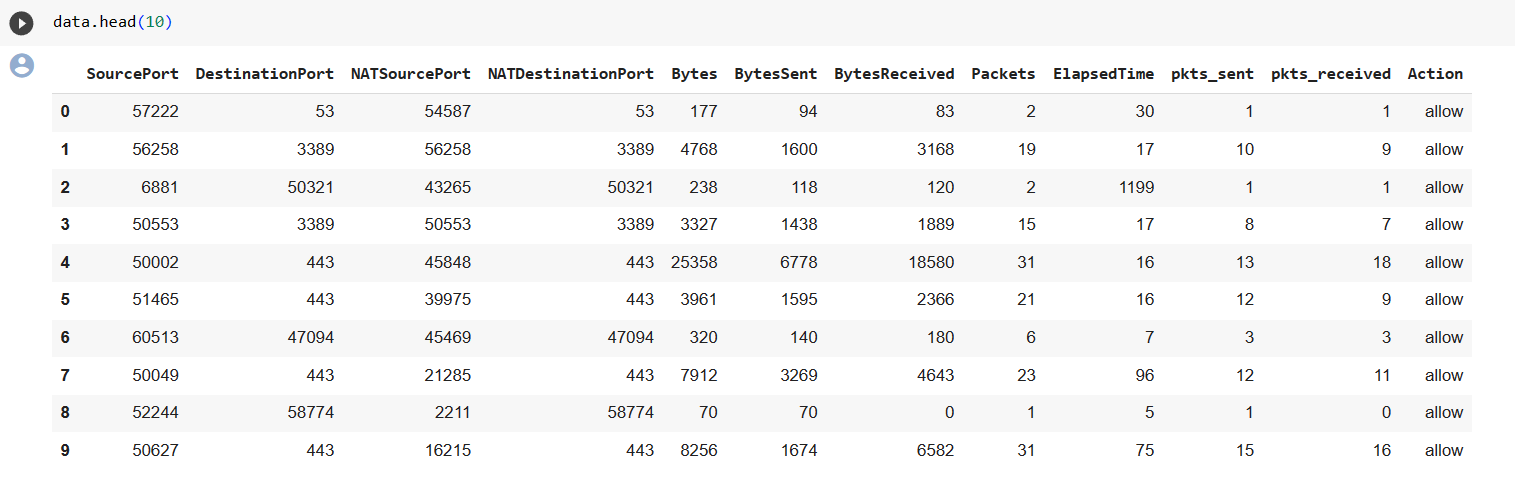
- graphviz: Thư viện này được sử dụng để hiển thị đồ thị từ dữ liệu DOT được tạo bởi export\_graphviz.

- google.colab (from google.colab import drive): Google Colab là một môi trường sử dụng Python trong trình duyệt. Dòng này được sử dụng để kết nối với Google Drive để lưu và tải dữ liệu.

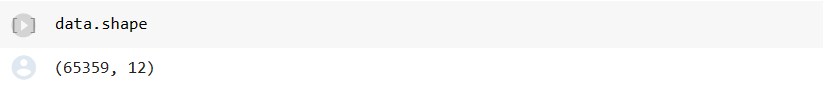
* Tất cả các thư viện và gói này là các công cụ quan trọng để thực hiện các nhiệm vụ liên quan đến xử lý dữ liệu, huấn luyện mô hình học máy và đánh giá mô hình.

### **2.1 Xem thông tin cơ bản của bộ dữ liệu**

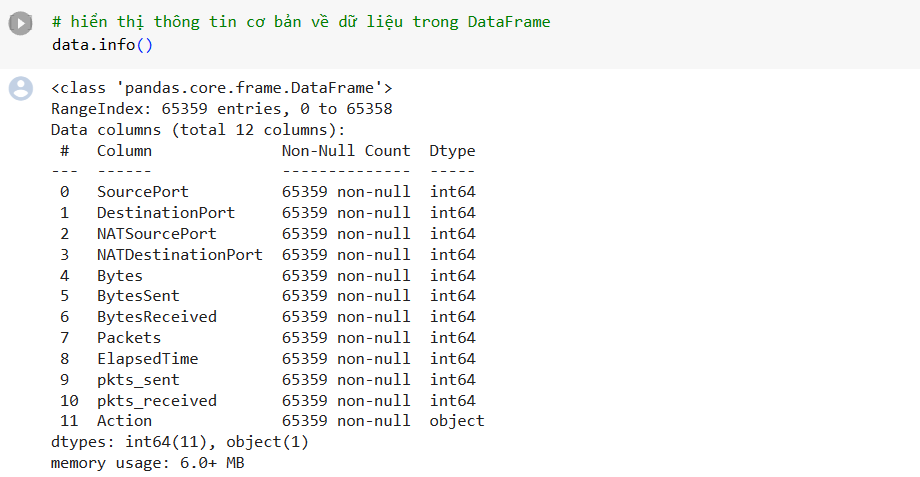
* Bộ dữ liệu thô vừa lấy chưa qua xử lý, xuất 10 dòng dữ liệu đầu tiên



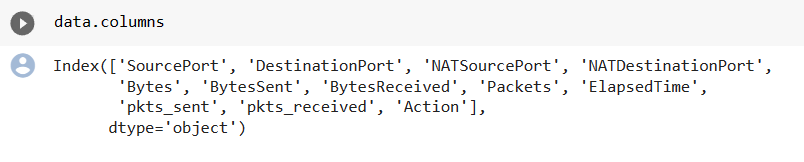
* Tổng có 65359 dòng, 12 cột dữ liệu (thuộc tính)



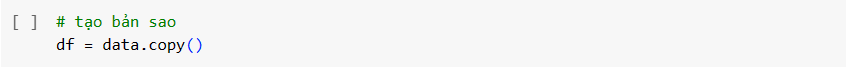
* Tổng quan về dữ liệu thô



* Qua dữ liệu trên ta thấy được kiểu dữ liệu của các thuộc tính đều là kiểu int64, trừ thuộc tính “Action” là kiểu ký tự .
* Xem tên các thuộc tính xuất hiện trong bộ dữ liệu

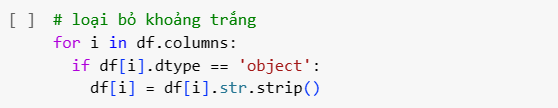


* 1. **Tiền Xử Lý Dữ Liệu**
* Tiến hành tạo bản sao dữ liệu

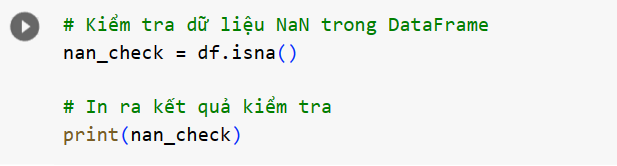


Sau đó:

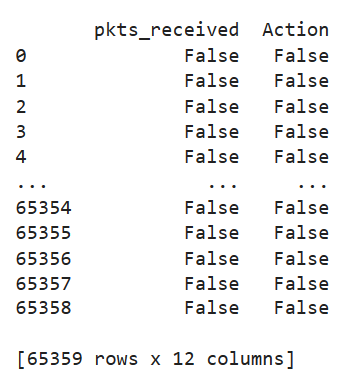
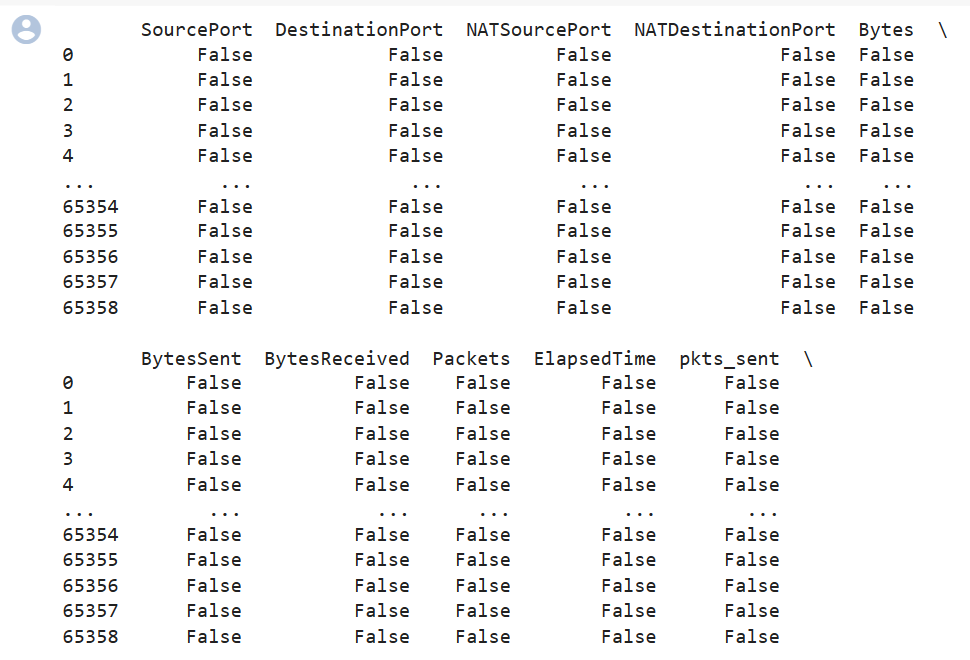
* Loại bỏ khoảng trắng trong dữ liệu



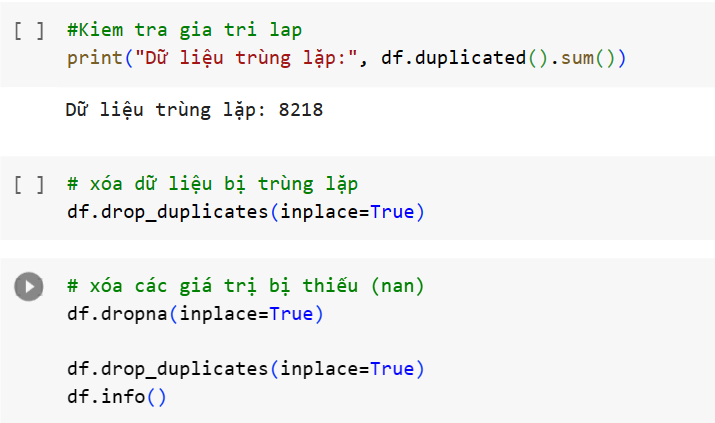
* Kiểm tra các dữ liệu mang giá trị NaN



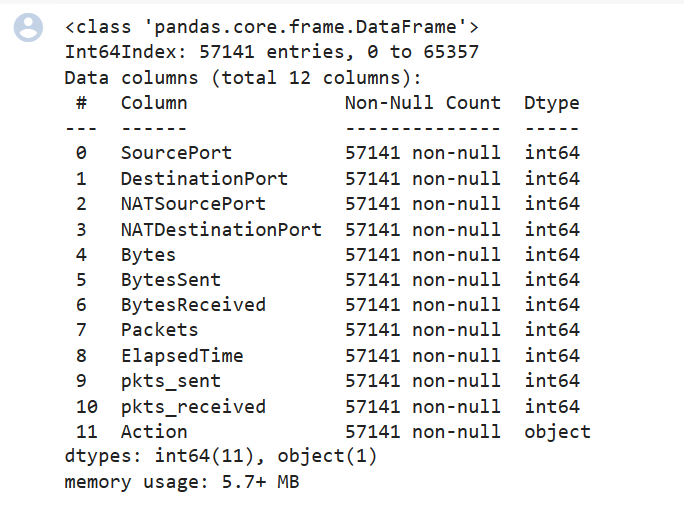
Trong kết quả sau, những dữ liệu False ( không NaN) sẽ được giữ lại, còn dữ liệu True (NaN) ta sẽ tiến hành xóa :



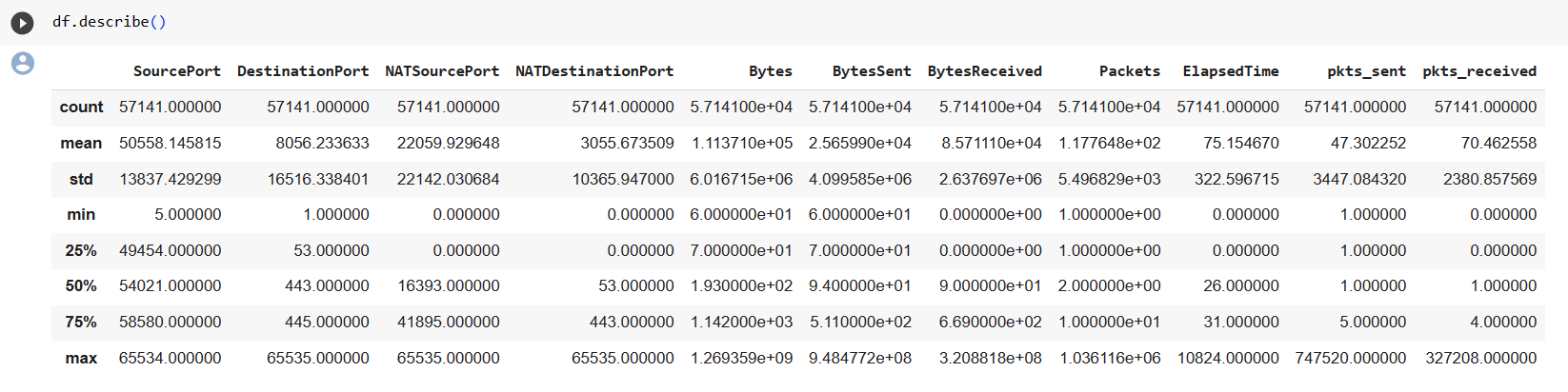
* Kiểm tra và xóa giá trị trùng lặp và giá trị NaN



Ta thấy có tới 8218 dữ liệu bị trùng lặp. Sau khi thực hiện xóa bộ dữ liệu của ta từ 65359 dòng còn 57141 dòng dữ liệu, các thuộc tính trùng lặp và Nan đã được xóa:



* Thống kê dữ liệu



* Nhìn kết quả, ta thấy:

- Số lượng (count): Có 57141 giá trị trong mỗi cột, cho thấy không có giá trị thiếu (null) trong dữ liệu.

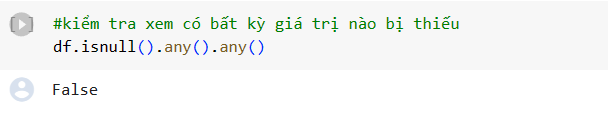
- Trung bình (mean): Giá trị trung bình của các cột có sự biến động khá lớn.

- Độ lệch chuẩn (std): cột Bytes có độ lệch chuẩn là khoảng 6016714, cho thấy sự biến đổi lớn trong dữ liệu.

- Giá trị tối thiểu (min) và giá trị tối đa (max): Cho biết giá trị nhỏ nhất và lớn nhất trong mỗi cột. Ví dụ, giá trị tối thiểu của cột SourcePort là 5 và giá trị tối đa là 65534.

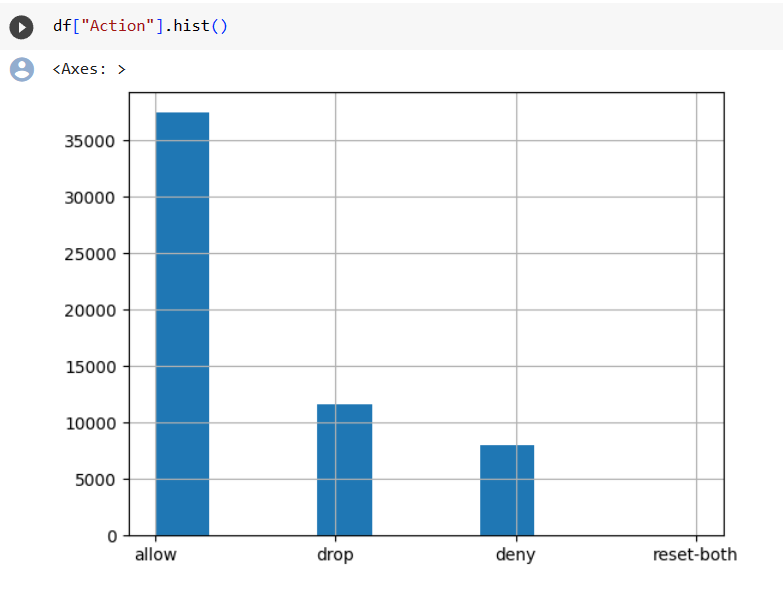
- Percentiles (25%, 50%, 75%): Đây là các phân vị (percentiles) của dữ liệu. Các phân vị này có thể giúp nhận ra đặc điểm phân bố của dữ liệu.

* Kiểm tra giá trị thiếu trong dữ liệu

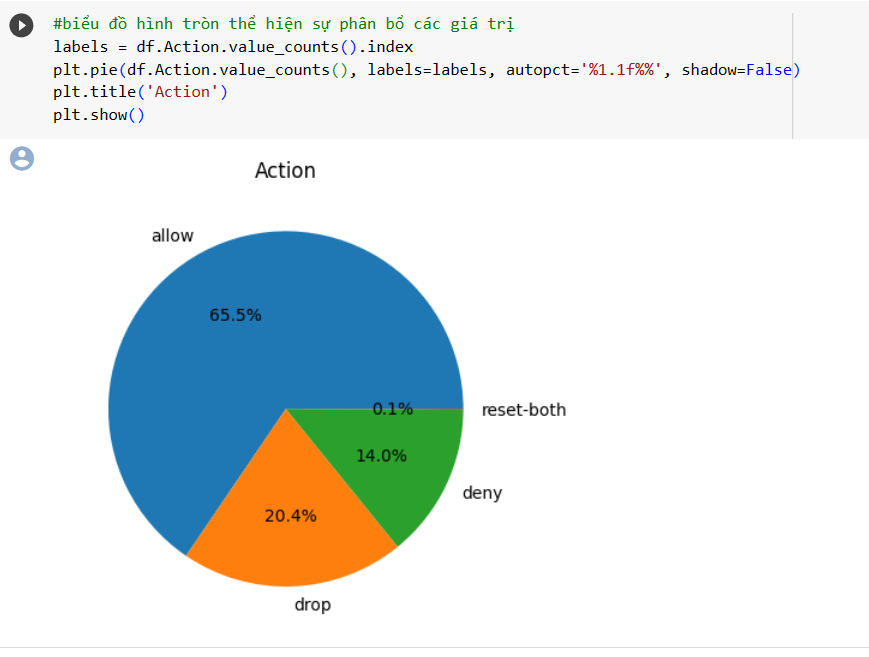


* Không có dữ liệu thiếu nào nên giá trị trả về là False.

* biểu đồ phân phối



* Biểu đồ tròn thống kê sự phân bổ giá trị



* Theo 2 biểu đồ ta thấy sự phân bổ của 4 giá trị “Allow”, “Drop”, “Deny”, “Reset-Both”:

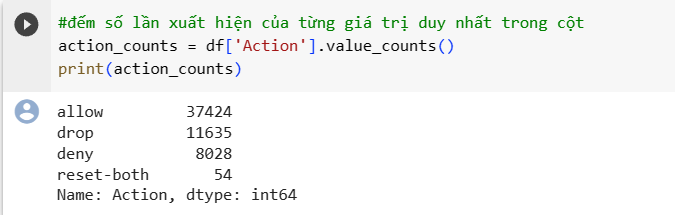
- Allow: chiếm nhiều giá trị nhất, hơn 35 ngàn dòng (65.5%) .

- Drop: chiếm hơn 10 ngàn dòng (20.4%).

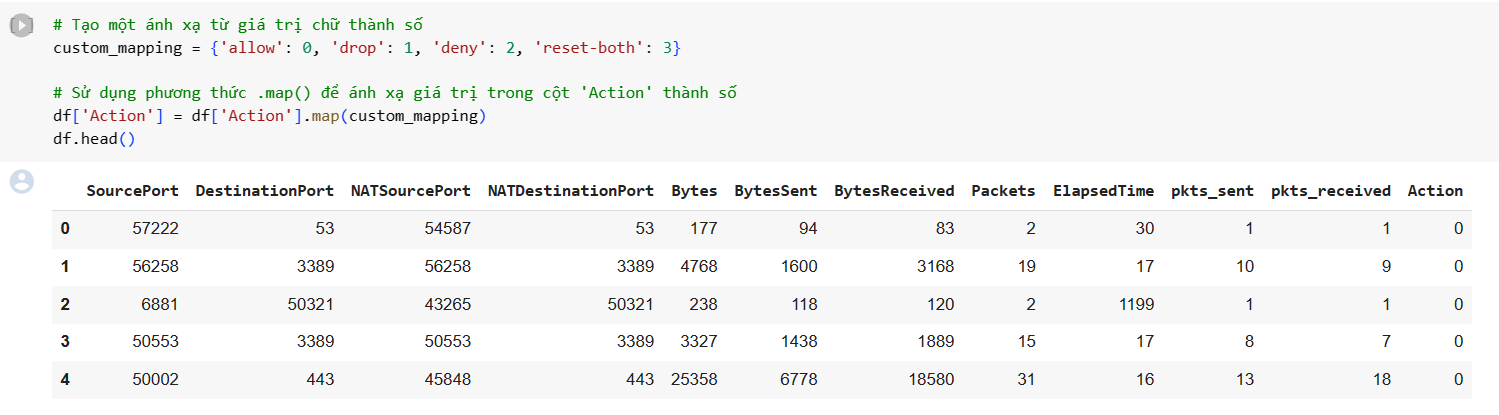
- Deny: chiếm gần 10 ngàn dòng (14.0%).

- Reset-Both: chiếm ít dòng nhất (0.1%).

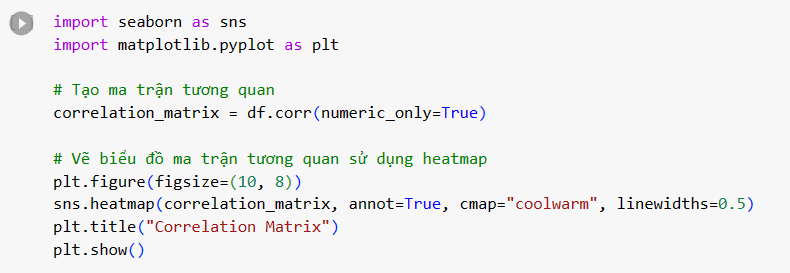
* Giá trị chính xác của từng giá trị

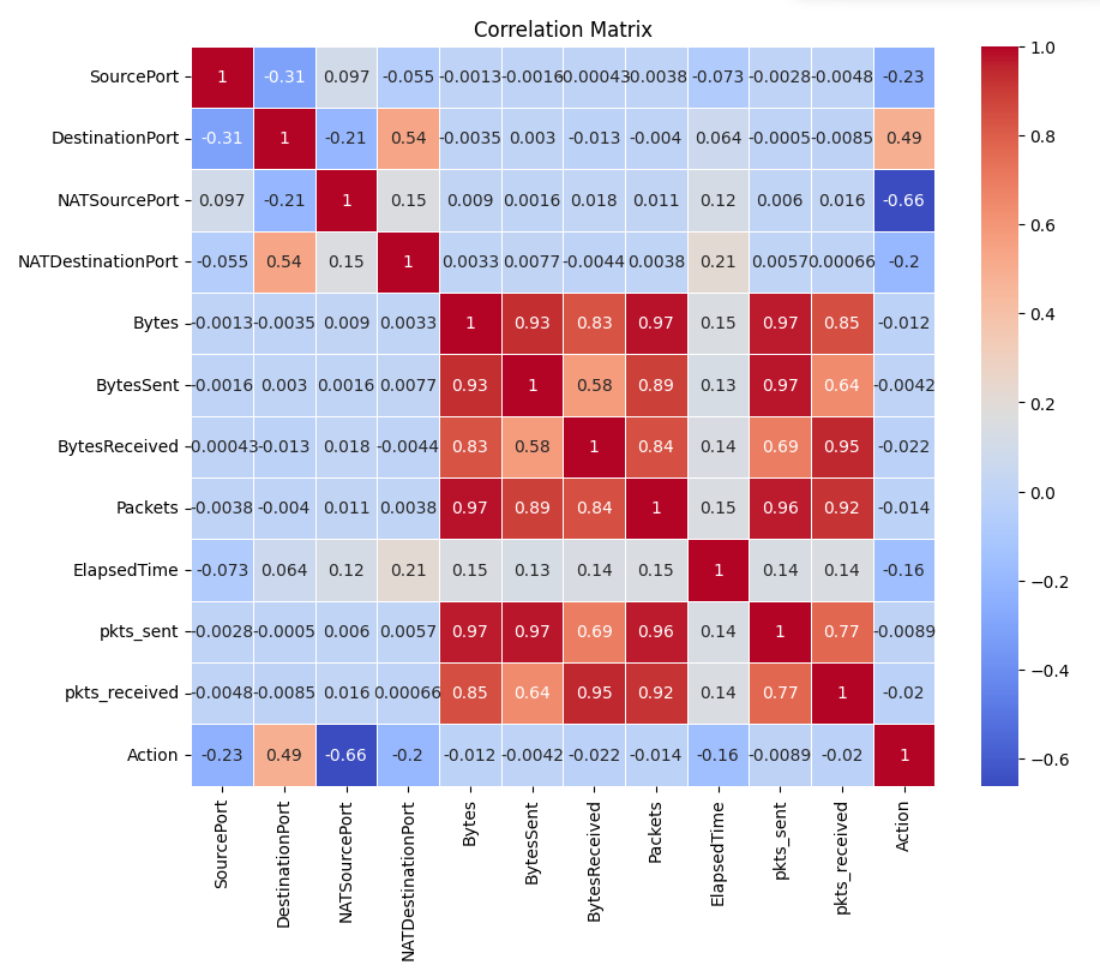


* Ta có thể thấy giá trị Allow chiếm đại đa số trong tổng số giá trị các trạng thái lưu lượng truy cập, kế tiếp sau lần lượt là 2 giá trị “Drop” và “Deny” cũng mang tổng số giá trị khá lớn, cuối cùng là giá trị “Reset-Both” chiếm thiểu số. Qua đó có thể thấy số lần kết nối lại lưu lượng truy cập là không đáng kể.
* Ánh xạ giá trị của cột Action thành số

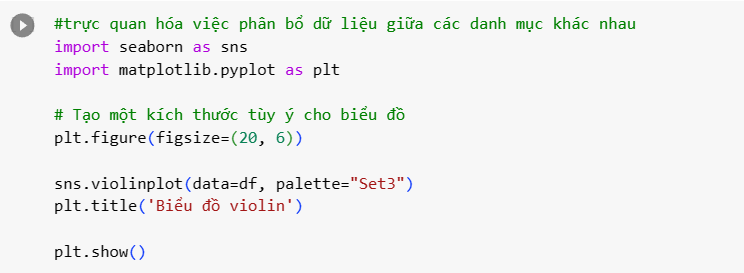


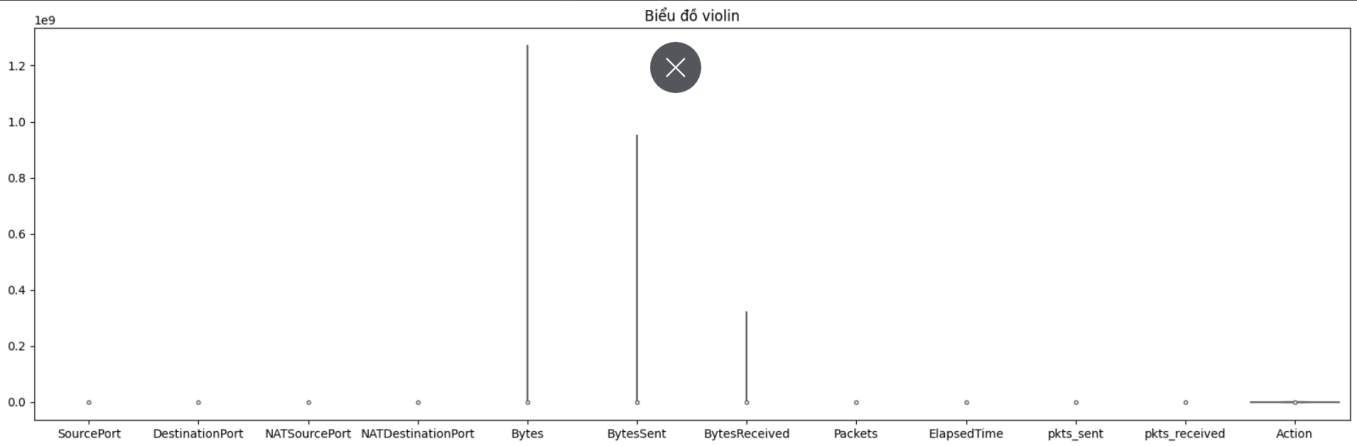
* Các giá trị “Action”, “Drop”, “Deny” và “Reset-Both” lần lượt ánh xạ các giá trị số “0”, “1”, “2”, “3”.
* Ta ánh xạ để tìm độ tương quan cho từng cột so với cột “Action”
* Ma trận độ tương quan của dữ liệu





* Qua ma trận, ta nhận thấy thuộc tính “Source Port” (-0.23), “Destination Port” (0.49) và “Elapsed Time” (-0.16) có độ tương quan cao hơn so với những cái khác.
* Cùng với biểu đồ tương quan trên, ta tiếp tục vẽ biểu đồ tương quan (violin) thể hiện sự phân bố dữ liệu với các thuộc tính khác nhau. Đây là sự kết hợp giữa biểu đồ hộp và biểu đồ mật độ phân bố. Nhầm đưa ra các thuộc tính có mối ràng buộc (tương quan) với thuộc tính target (cụ thể là ‘Action’).
* Biểu đồ Violin kiểm tra, so sánh sự phân bố giữa các thuộc tính

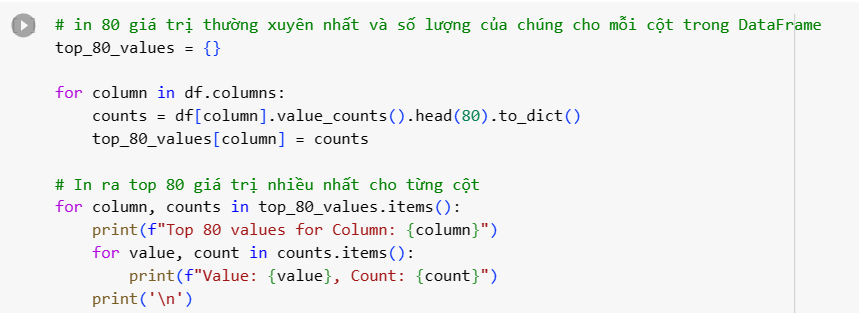




* Qua biểu đồ, ta thấy cột thuộc tính “Bytes”, “Bytes Sent” và “Bytes Received” có sự phân bố lớn, tác động lớn đến đến các giá trị trong thuộc tính “Action”.
* Qua ma trận tương quan và biểu đồ Violin, ta rút ra được các thuộc tính ảnh hưởng đến kết quả của các giá trị trong cột thuộc tính “Action”.
* Xóa đi các thuộc tính có ít độ tương quan và ảnh hưởng là: NATSourcePort, NATDestinationPort, Packets, pkts\_sent, pkts\_recevied.



* 1. **Chuẩn hóa dữ liệu**
* Top 80 các giá trị xuất hiện nhiều nhất trong từng cột thuộc tính



* Từ top 80 giá trị xuất hiện nhiều nhất:

- cột SourcePort ta thấy các giá trị trên 60000 xuất hiện không quá 15 lần nên tiến hành đặt ngưỡng cho cột SourcePort.

- cột DestinationPort ta thấy các giá trị trên 60000 xuất hiện 1079 lần nên tiến hành đặt ngưỡng cho cột DestinationPort.

* Giới hạn ngưỡng dữ liệu



Sau khi xem top 80 giá trị nhiều nhất của từng cột thì ta đưa ra ngưỡng cho từng cột, không để các giá trị trong một cột chênh lệch quá nhiều. Cụ thể như sau:

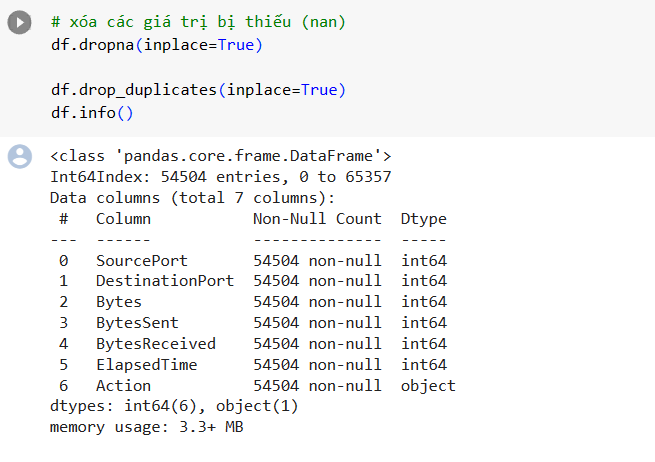
- Cột SourcePort, DestinationPort: Áp dụng ngưỡng 60000, bỏ các giá trị lớn hơn ngưỡng (vượt quá giới hạn chuẩn của cổng).

- Cột Bytes: Áp dụng ngưỡng 1000, bỏ dữ liệu lớn hơn (có thể là dữ liệu bất thường).

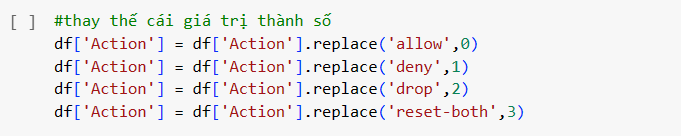
- Cột BytesSent, ElapsedTime: Áp dụng ngưỡng 2000, bỏ dữ liệu lớn hơn.

- Cột BytesReceived: Áp dụng ngưỡng 2000 , bỏ dữ liệu lớn hơn ngưỡng.

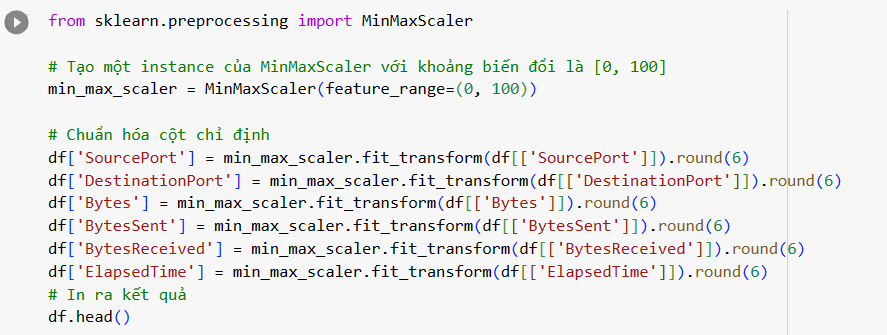
* Chúng ta tiến hành đặt ngưỡng giá trị cho các cột (thuộc tính) trên nhằm mục định loại bỏ bớt giá trị bị nhiễu, nói cách khác là tránh giá trị bị chênh lệch quá nhiều.
* Từ đó ta tiến hành xử lý thuật toán sẽ có hiệu quả tốt hơn.
* Xóa các dữ liệu bị thiếu sau khi thực hiện bước trên.
* Sau khi đặt ngưỡng ta tiến hành xóa các giá trị bị thiếu sau khi chúng ta đặt ngưỡng.



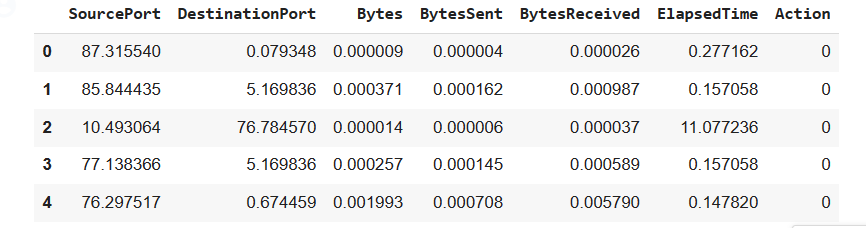
* Định dạng dữ liệu của cột “Action” thành số liệu cụ thể.



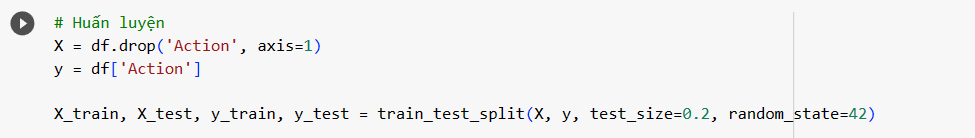
* Các giá trị “Action”, “Drop”, “Deny” và “Reset-Both” lần lượt chuyển đổi sang các giá trị số “0”, “1”, “2”, “3”.
* Mục đích là để đồng bộ 1 kiểu dữ liệu int64 cho việc huấn luyện mô hình.
* Chuẩn hóa bằng phương pháp MinMaxScaler



* Mục đích để ta biến đổi dữ liệu sao cho min, max trong khoản từ 0 đến 100, đồng thời hiển thị kết quả dưới dạng 6 số thập phân thay thế vào các cột dữ liệu cũ.
* Sau khi chuẩn hóa hết các giá trị cần thiết ta có bộ dữ liệu sau.



* 1. **Huấn luyện mô hình**
* Khai báo thư viện và huấn luyện tập dữ liệu



* Ta chia tập dữ liệu thành tập huấn luyện và đánh giá mô hình

Trong đó:

- **X** là các biến độc lập (các cột trừ cột 'Action')

- **y** là biến phụ thuộc cần phân lớp (cột 'Action')

- **train\_test\_split()** tách tập dữ liệu thành 2 tập:

- Tập huấn luyện (**X\_train, y\_train**) để xây dựng mô hình

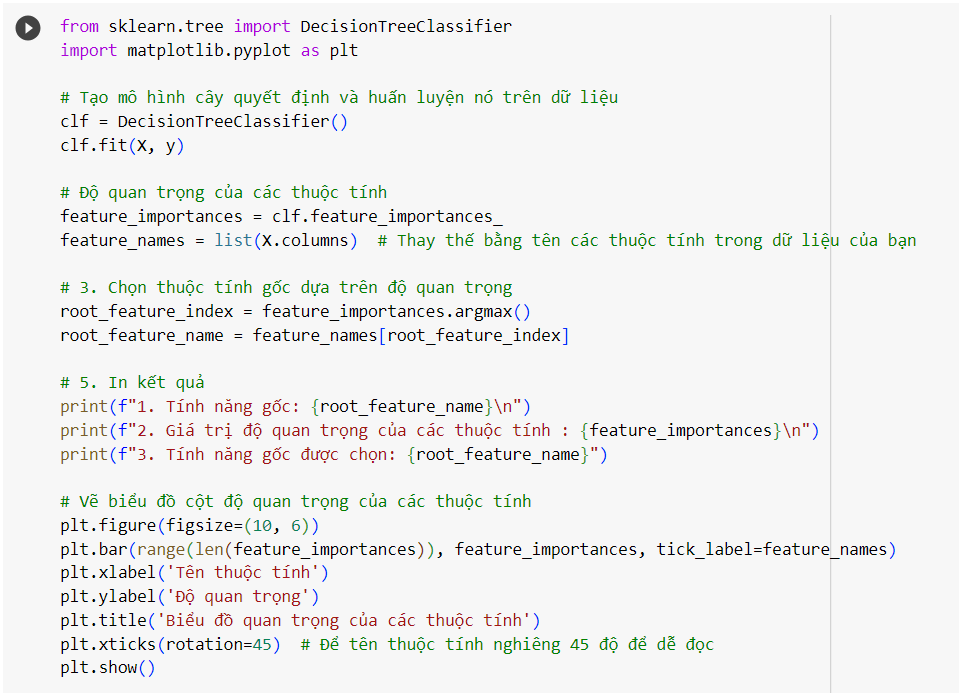
- Tập kiểm tra (**X\_test, y\_test**) để đánh giá mô hình

- **test\_size=0.2** chọn 20% dữ liệu làm tập kiểm tra

- **random\_state=42** cho kết quả lặp lại được

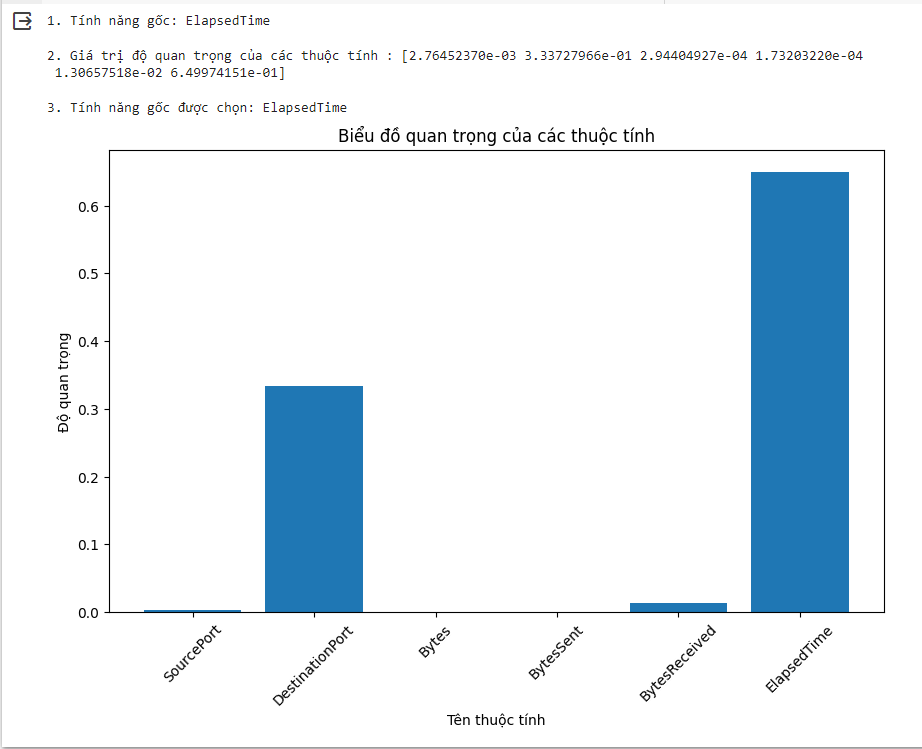
1. **Thuật toán Decision Tree**

Sau đây là code mô hình:

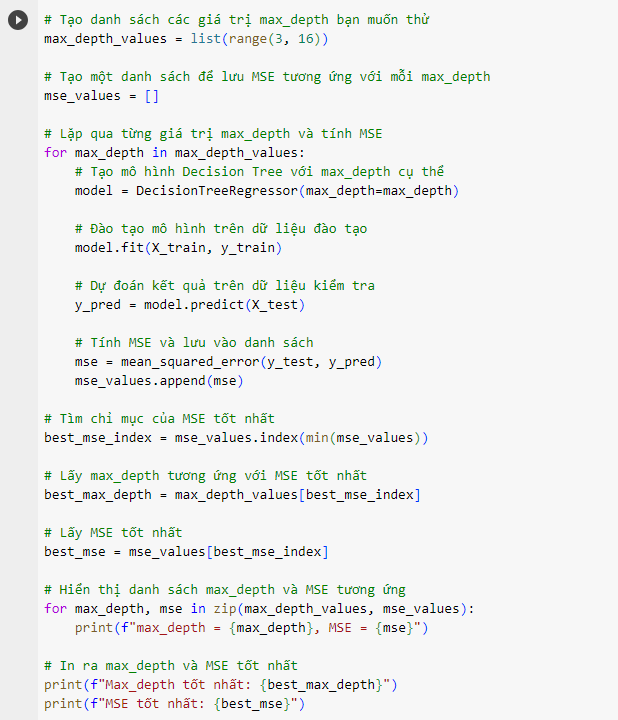


Code trên dựa vào thuộc tính "độ quan trọng" (feature\_importances) được tính toán bởi mô hình cây quyết định để xác định thông số quan trọng của các thuộc tính. Độ quan trọng của một thuộc tính cho biết mức độ mà thuộc tính đó đóng góp vào quá trình ra quyết định của cây quyết định trong việc phân loại hoặc dự đoán dữ liệu. Cách mà code hoạt động:

* Mô hình cây quyết định được huấn luyện trên dữ liệu đào tạo sử dụng **clf.fit(X, y)**. Trong quá trình huấn luyện, cây quyết định tự động xác định cách phân tách dữ liệu dựa trên các thuộc tính.
* Sau khi mô hình được huấn luyện, thuộc tính **feature\_importances** được trích xuất từ cây quyết định. Đây là một mảng chứa giá trị độ quan trọng của tất cả các thuộc tính trong dữ liệu.
* **feature\_importances** để xác định thuộc tính gốc bằng cách tìm chỉ số của thuộc tính có giá trị độ quan trọng cao nhất. Điều này có nghĩa là thuộc tính có đóng góp lớn nhất vào quyết định của cây quyết định sẽ được chọn làm thuộc tính gốc.
* Kết quả cũng hiển thị giá trị độ quan trọng của tất cả các thuộc tính bằng biểu đồ cột để bạn có cái nhìn tổng quan về tầm quan trọng của từng thuộc tính.
* Trong đó: thông số quan trọng của một thuộc tính là một số từ 0 đến 1, 0 có nghĩa là thuộc tính không đóng góp gì vào quyết định của mô hình và 1 có nghĩa là thuộc tính quyết định mọi quyết định.



Từ biểu đồ trên, ta thấy rõ cột thuộc tính”ElapsedTime” có độ quan trọng cao nhất, nối tiếp sau đó là các cột thuộc tính với độ quan trọng giảm dần từ cột “DestinationPort” đến cột thuộc tính “BytesReceived” và “SourcePort” cũng có độ quan trọng nhất định. Cuối cùng là các cột không có độ quan trọng, không gây ảnh hưởng quyết định đến mô hình.

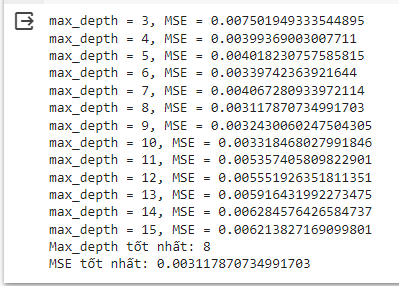


- Đoạn mã này thực hiện việc chọn tham số tối ưu max\_depth cho mô hình DecisionTreeRegressor.

- Ta tạo một danh sách chứa các giá trị max\_depth cần thử nghiệm, từ 3 đến 15. Sau đó tạo một danh sách rỗng để lưu MSE tương ứng với mỗi giá trị ax\_depth.

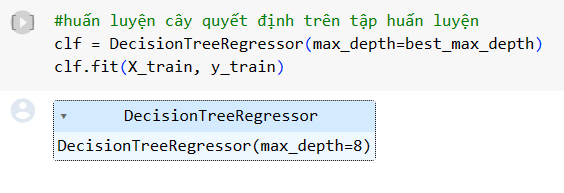
- Tiếp theo ta sử dụng vòng lặp để xây dựng và đào tạo mô hình với từng max\_depth cụ thể. Dự đoán kết quả trên dữ liệu test và tính toán MSE để lưu vào danh sách đã tạo. Sau khi kết thúc vòng lặp, chúng ta tìm vị trí MSE nhỏ nhất trong danh sách, lấy giá trị max\_depth tương ứng.

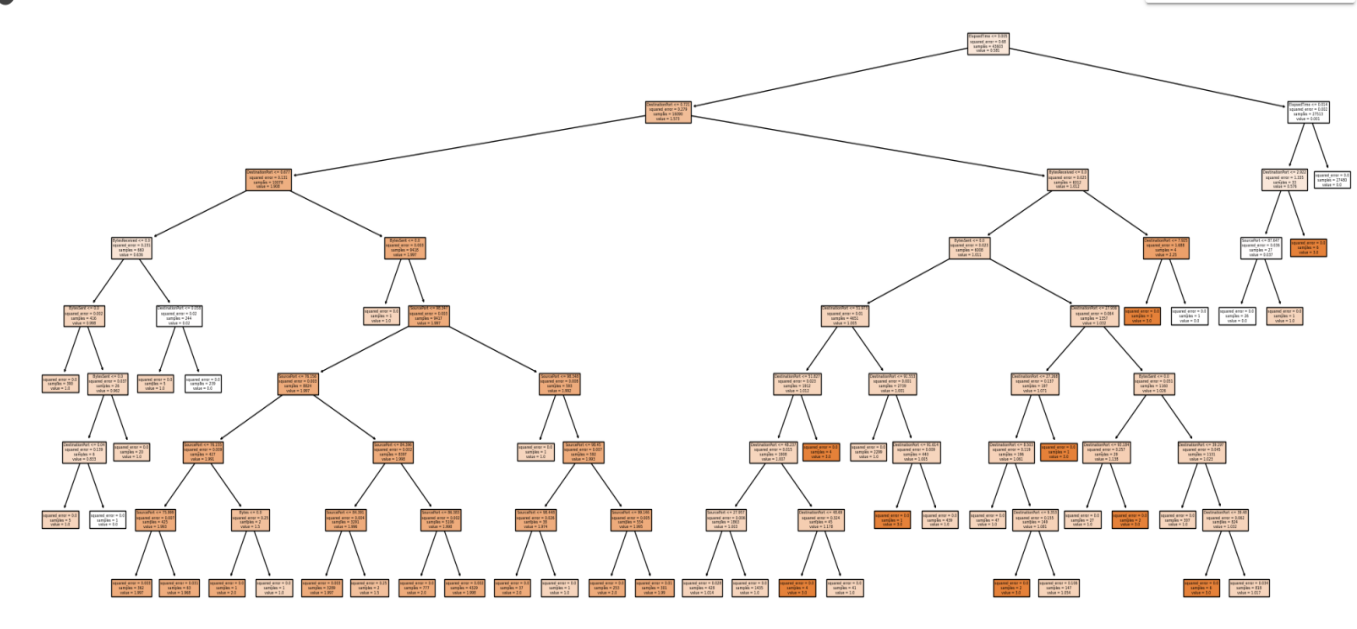
- Cuối cùng hiển thị danh sách max\_depth và MSE, cũng như giá trị max\_depth và MSE tối ưu nhất được chọn.



* Qua đó chúng ta xác định được giá trị max\_depth phù hợp nhất cho mô hình là 8.

Sau khi có max\_depth tốt nhất là 8 ta tiến hành vẽ cây:



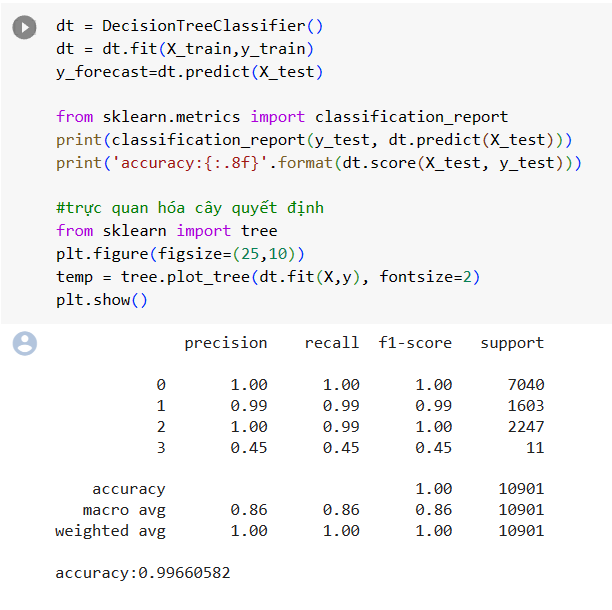


Dựa vào cây quyết định trên ta có thể thấy được sự ảnh hưởng giữa các thuộc tính với nhau như:

-Thuộc tính Elapsed Time được đặt làm nút gốc với 3 khoảng phân chia:

* Elapsed Time <= 0.005: Có tới 16090 giá trị
* 0.095<Elapsed Time <= 0.014: Có 33 giá trị.
* Elapsed Time > 0.014: Có 27480 giá trị.

-Tiếp theo ta đi đánh giá mô hình và trực quan hóa cây quyết định:



**Lớp 0 (precision: 1.00, recall: 1.00, F1-score: 1.00):** Mô hình hoàn toàn chính xác trong việc dự đoán lớp 0. Không có dự đoán sai và không bỏ sót bất kỳ trường hợp nào của lớp này. Điều này là một hiệu suất tốt.

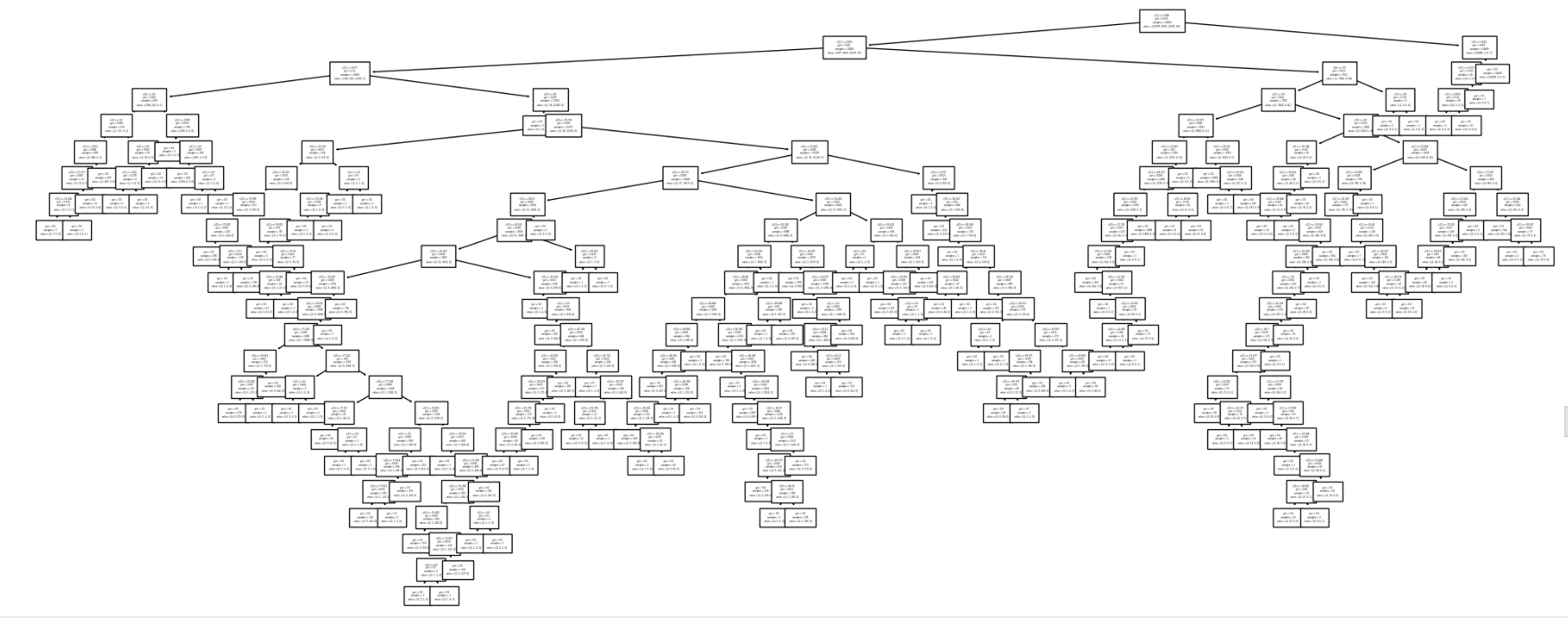
**Lớp 1 (precision: 0.99, recall: 0.99, F1-score: 0.99):** Mô hình có hiệu suất tốt cho lớp 1. Gần như tất cả dự đoán là chính xác và chỉ mất đi một vài trường hợp.

**Lớp 2 (precision: 1.00, recall: 0.99, F1-score: 1.00):** Mô hình hoàn toàn chính xác trong việc dự đoán lớp 0. Không có dự đoán sai và không bỏ sót bất kỳ trường hợp nào của lớp nào và chỉ bỏ sót một số trường hợp nhỏ.

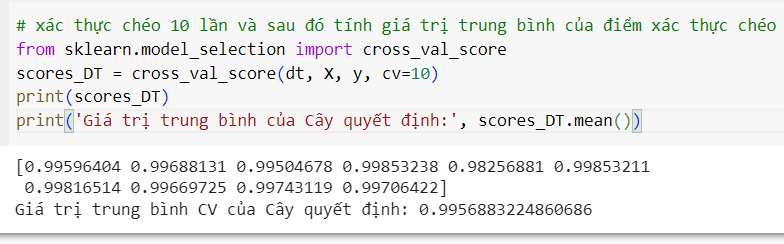
**Lớp 3 (precision: 0.45, recall: 0.45, F1-score: 0.45):** Đối với lớp 3, mô hình có hiệu suất thấp hơn. Mô hình chỉ dự đoán đúng khoảng 45% trường hợp và bỏ sót một số trường hợp quan trọng.

Nhìn chung, ta có thể thấy các thông số đánh giá mô hình có tỉ lệ khá cao, cho thấy bộ dữ liệu tương tác rất tốt với mô hình. Độ chính xác là tỷ lệ dự đoán đúng trên tổng số dự đoán do mô hình đưa ra. Trong trường hợp này, độ chính xác được báo cáo là 0,9967, có nghĩa là độ chính xác tổng thể của mô hình là khoảng 99,67%.

Trực quan hóa cây quyết định:

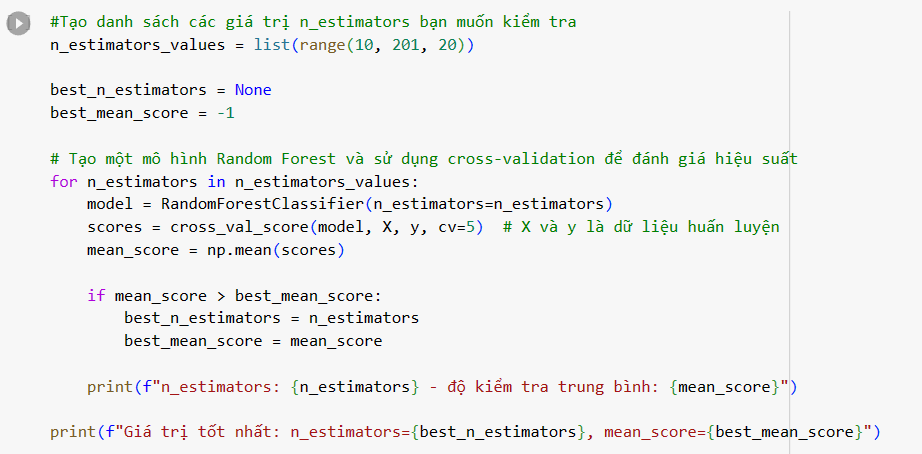


- Cuối cùng đi xác thực chéo sử dụng để tính giá trị trung bình của các điểm số, cho biết mức độ chính xác trung bình của mô hình qua 10 lần xác thực chéo.



mã nguồn trên giúp chúng ta kiểm tra và đánh giá hiệu suất của mô hình Cây quyết định bằng cách sử dụng xác thực chéo và tính giá trị trung bình của các điểm số thu được. Điều này giúp đảm bảo mô hình hoạt động tốt trên nhiều tập dữ liệu khác nhau và giảm nguy cơ overfitting hoặc underfitting và dựa trên kết quả này, mô hình Cây quyết định có vẻ rất tốt và có khả năng dự đoán tốt trên các tập dữ liệu mà nó chưa thấy trước đó.

## Thuật toán Random Forest

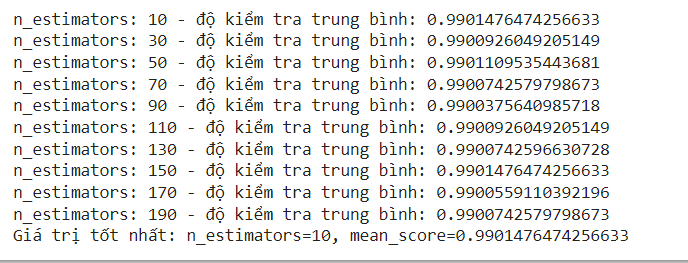


Đầu tiên tạo list “n\_estimators” tìm kiếm giá trị tối ưu cho siêu tham số. Sau đó sử dụng “cross\_val\_score” để đánh giá hiệu suất của mô hình thông qua xác thực chéo.

Sau đó ta tạo 2 biến best\_n\_estimators và best\_mean\_score để theo dõi giá trị tốt nhất của "n\_estimators" và điểm số trung bình tốt nhất mà ta đã tìm thấy cho đến nay. Ban đầu, chúng được đặt thành None và -1 để sử dụng làm điểm tham chiếu để so sánh với các giá trị mới và Sau đó, một vòng lặp được sử dụng để duyệt qua từng giá trị của "n\_estimators" trong danh sách n\_estimators\_values, Trong vòng lặp, một mô hình Random Forest được tạo với "n\_estimators" hiện tại

Sử dụng phương thức cross\_val\_score để chia dữ liệu huấn luyện thành 5 phần và mô hình được đào tạo và kiểm tra trên từng phần, sau đó tính điểm số cho từng lần kiểm tra, Sau đó, điểm số trung bình của mô hình trên 5 fold được tính toán và lưu vào biến mean\_score.  
kiểm tra nếu mean\_score lớn hơn best\_mean\_score hiện tại. Nếu có, thì best\_n\_estimators và best\_mean\_score được cập nhật để lưu giữ giá trị tốt nhất mà ta đã tìm thấy cho đến nay.

Cuối cùng, sau khi hoàn thành vòng lặp, bạn in ra giá trị “n\_estimators” tốt nhất và giá trị “mean\_score” tốt nhất ở bên dưới.



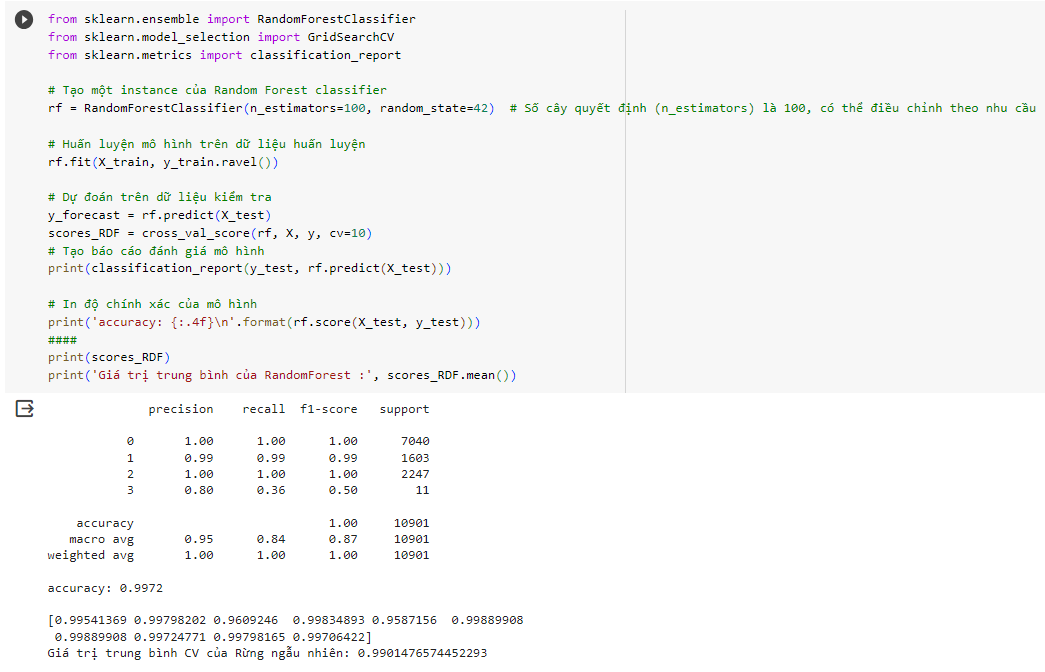
Dựa vào các giá trị độ kiểm tra trung bình (mean score) với các giá trị khác nhau của siêu tham số "n\_estimators" trong mô hình của bạn, ta có thể rút ra một số nhận xét sau:

- Độ kiểm tra trung bình tăng đáng kể khi "n\_estimators" thay đổi từ 10 lên 30, điều này cho thấy việc tăng số cây quyết định (estimators) từ 10 lên 30 đã cải thiện hiệu suất của mô hình, Tuy nhiên, sau khi "n\_estimators" đạt 30, giá trị độ kiểm tra trung bình dường như đã không thay đổi đáng kể (điều này có thể được thấy qua sự giữ nguyên của giá trị độ kiểm tra trung bình từ 30 đến 190).

- Giá trị tốt nhất của "n\_estimators" dựa trên độ kiểm tra trung bình là 10, với một giá trị độ kiểm tra trung bình lên đến khoảng 0.9975. Điều này có nghĩa là sử dụng chỉ 10 cây quyết định trong mô hình đã đủ để có hiệu suất tốt nhất trong trường hợp này.

- Việc sử dụng nhiều cây quyết định hơn không cải thiện đáng kể hiệu suất mô hình, và điều này có thể dẫn đến một mô hình cồng kềnh hơn mà không mang lại sự cải thiện đáng kể.

Sau khi tìm được tham số trên ta tiến hành đi đánh giá mô hình, các chỉ số này được sử dụng để đánh giá hiệu suất của mô hình phân loại và cung cấp thông tin về mức độ chính xác, độ tin cậy và khả năng tìm ra các trường hợp thực tế.



Lớp 0: Mô hình dự đoán chính xác 100% cho lớp 0 với precision, recall và f1-score đều bằng 1. Điều này đề xuất mô hình hoạt động rất tốt đối với lớp này.

Lớp 1: Mô hình có precision, recall và f1-score tương đối cao cho lớp 1, với mức độ chính xác là 99%. Điều này cũng là một hiệu suất tốt.

Lớp 2: Lớp 2 cũng có hiệu suất cao, với precision, recall và f1-score đều bằng 1.

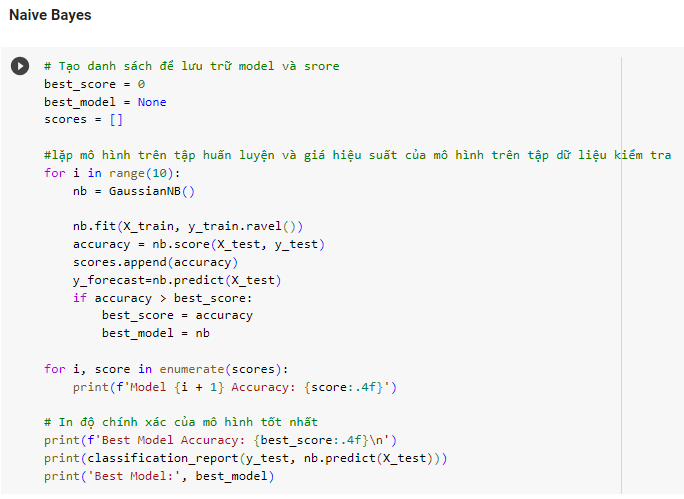
Lớp 3: Lớp 3 có hiệu suất thấp hơn, với recall và f1-score thấp hơn đáng kể. Điều này đề xuất mô hình gặp khó khăn trong việc dự đoán lớp này.

Giá trị trung bình của RandomForest là 0.9901660027029802 là một hiệu suất tốt cho mô hình.

Từ hình trên ta thấy các thông số khá cân bằng với mô hình, “Accuracy” đây là tỷ lệ phần trăm của các dự đoán chính xác trên toàn bộ tập dữ liệu. Trong trường hợp của bạn, accuracy là 0.9972, tức là mô hình của bạn rất chính xác trong việc phân loại.

Cuối cùng, giá trị trung bình của Random Forest là 0.9901476574452293, tức là giá trị trung bình của các điểm xác thực chéo khi mô hình được đánh giá trong 10 lần xác thực chéo. Điều này cho thấy mô hình Random Forest có hiệu suất phân loại tốt trên dữ liệu của bạn.

## Thuật toán Naive Bayes



Sau khi huấn luyện mô hình ở trên ta tiến hành xử lý thuật toán Naïve Bayes:

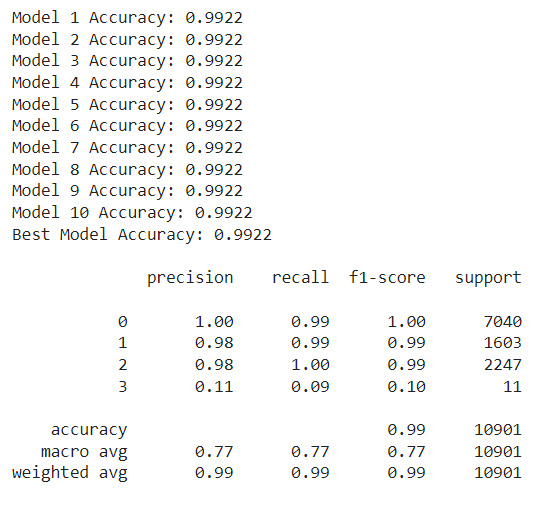
Đầu tiên ta khởi tạo biến “best\_score” và “best\_model” để lưu trữ điểm số tốt nhất và mô hình tốt nhất tìm thấy.

Tiếp theo sử dụng vòng lặp for để tạo và huấn luyện mô hình Gaussian Naive Bayes (NB) 10 lần (với i chạy từ 0 đến 9).

Sử dụng fit để huấn luyện mô hình trên tập huấn luyện X\_train và y\_train, sau đó tính toán độ chính xác của mô hình.

Sau khi vòng lặp hoàn thành, bạn in ra điểm số của tất cả các mô hình được đào tạo.

Cuối cùng tính toán và in báo cáo phân loại cho mô hình tốt nhất, cung cấp số liệu đánh giá chi tiết hơn cho từng lớp.



Độ chính xác của mô hình(accuracy): Tất cả 10 mẫu đều đạt được độ chính xác như nhau khoảng 99,22% trên dữ liệu thử nghiệm.

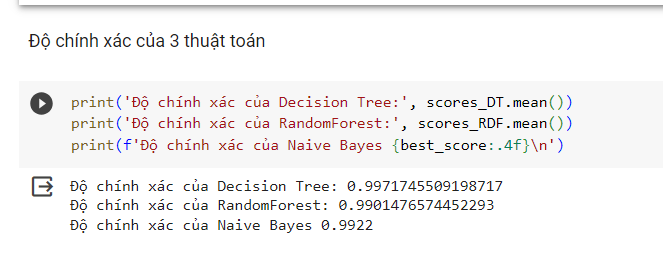
Đối với các lớp 0, 1 và 2, mô hình hoạt động khá tốt với độ chính xác(precision), khả năng thu hồi(recall) và điểm F1 cao, cho thấy hiệu suất phân loại tốt. Các lớp này có số lượng mẫu lớn(support) (ví dụ: 7040, 1603 và 2247), góp phần mang lại khả năng hỗ trợ cao.

Đối với loại 3, hiệu suất của mô hình thấp hơn đáng kể. Độ chính xác, khả năng thu hồi và điểm F1 thấp hơn nhiều và mức hỗ trợ chỉ là 11. Điều này cho thấy rằng lớp 3 đang gặp khó khăn trong việc phân loại và mô hình đang gặp khó khăn với nó.

Cụ thể là do loại 3 chiếm giá trị phân phối khá thấp trong bộ dữ liệu khoảng 0.1%.

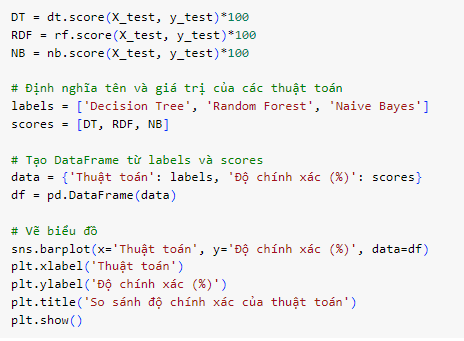
Nhìn chung độ chính xác cao, các thông số cũng khá tốt nhưng vấn đề về sự cân bằng khả năng thu hồi và các số liệu khác còn chút thiên vị.

1. **Nhận xét các thuật toán và test với bộ data mới**

Ta xét độ chính xác của 3 thuật toán

Nhìn chung, ta thấy độ chính xác của 3 thuật toán khá cao và không chênh lệch quá nhiều, nhưng thuật toán Decision Tree có độ chính xác nhỉnh hơn các thuật toán khác, cụ thể là: 0.9956883224860686.

Sau đó ta đi vẽ biểu đồ so sánh để nhìn rõ hơn.



Đầu tiên ta tính độ chính xác của mô hình Decision Tree (dt) trên dữ liệu thử nghiệm (X\_test và y\_test). dt.score là một phương thức trong mô hình Decision Tree được sử dụng để tính độ chính xác. Kết quả được nhân với 100 để biểu thị dưới dạng phần trăm và lưu vào biến DT, sau đó ta tính độ chính xác của mô hình Random Forest (rf) trên dữ liệu thử nghiệm và lưu vào biến RDF, tiếp đến ta tính độ chính xác của mô hình Naive Bayes (nb) trên dữ liệu thử nghiệm và lưu vào biến NB.

Biến labels và scores sẽ lưu tên và giá trị của 3 thuật toán trên với labels sẽ lưu tên, tương ứng scores sẽ lưu giá trị

từ labels và scores ta sẽ tạo DataFrame:

- data = {'Thuật toán': labels, 'Độ chính xác (%)': scores}: Ở đây, một từ điển được tạo với hai khóa. Khóa "Thuật toán" chứa danh sách các tên thuật toán và khóa "Độ chính xác (%)" chứa danh sách các giá trị độ chính xác tương ứng.

- df = pd.DataFrame(data): Dùng pandas để tạo một DataFrame từ dữ liệu trong từ điển data. DataFrame này chứa thông tin về tên thuật toán và độ chính xác của chúng.

Sau cùng ta sẽ vẽ biểu đồ:

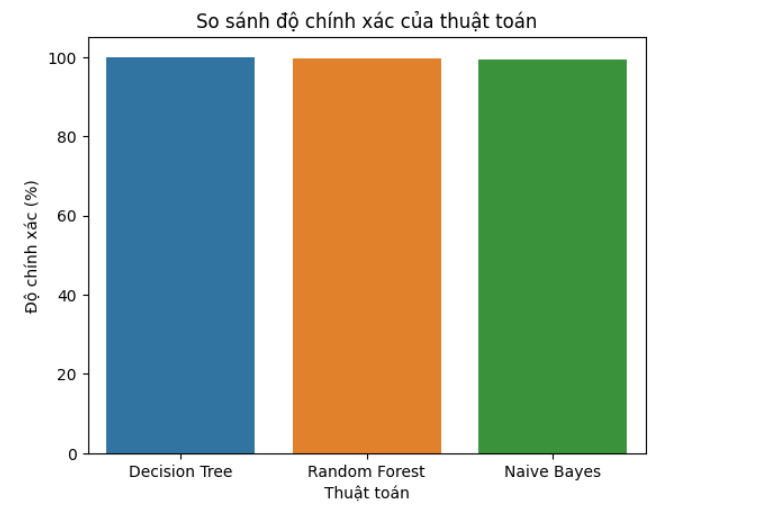
Ta dùng sns.barplot(x='Thuật toán', y='Độ chính xác (%)', data=df): Sử dụng thư viện seaborn để vẽ biểu đồ cột (barplot) với trục x là "Thuật toán" và trục y là "Độ chính xác (%)", sử dụng dữ liệu từ DataFrame df.

Dùng plt.xlabel('Thuật toán'): Đặt nhãn cho trục x trên biểu đồ là "Thuật toán".

Dùng plt.ylabel('Độ chính xác %'): Đặt nhãn cho trục y trên biểu đồ là "Độ chính xác %".

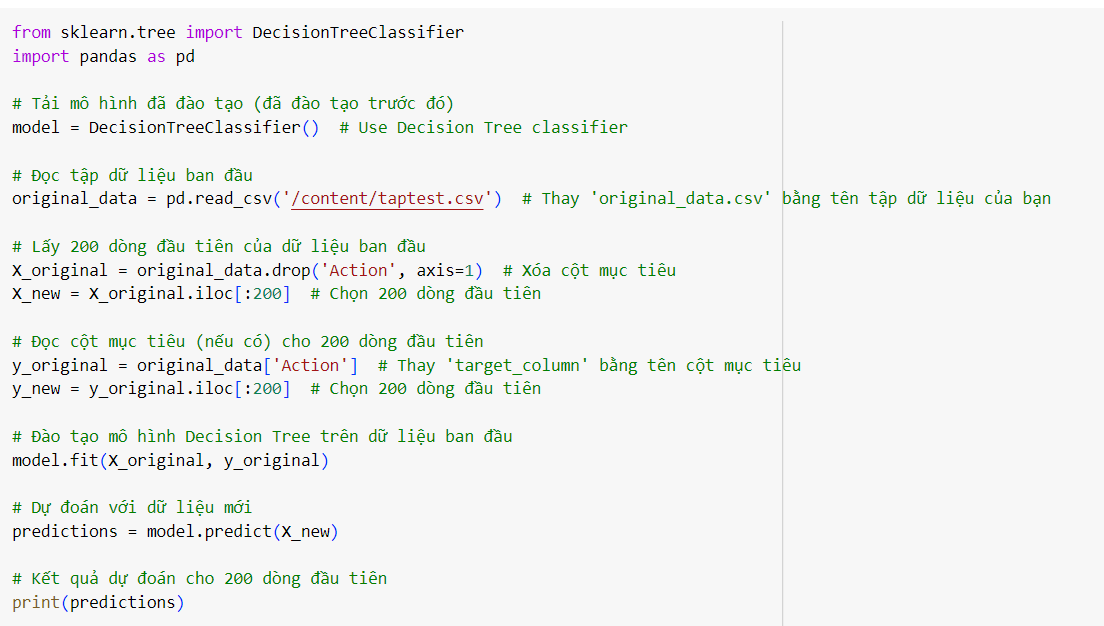
Dùng plt.title('So sánh độ chính xác của thuật toán'): Đặt tiêu đề cho biểu đồ.

Dùng plt.show(): Hiển thị biểu đồ.



Sau khi nhận được độ chính xác của Decision Tree là cao nhất ta đi test thuật toán đó với bộ data mới.

Chúng ta tiến hành tải mô hình trước đó, đọc dữ liệu mới (cụ thể là tệp lấy từ 700 dòng dữ liệu từ tệp góc), sau đó cho test với 200 dòng.



Đầu tiên ta khai báo 2 thư viện DecisionTreeClassifier từ scikit-learn và thư viện pandas với alias là pd. Pandas được sử dụng để làm việc với dữ liệu dạng bảng (DataFrame).

Dùng thuật toán model = DecisionTreeClassifier(): Tạo một đối tượng Decision Tree Classifier và lưu nó trong biến model. Đây sẽ là mô hình cây quyết định mà chúng ta sẽ đào tạo.

Dùng original\_data = pd.read\_csv('/content/drive/MyDrive/Colab DataMining/taptest.csv'): Sử dụng pandas để đọc dữ liệu từ tập tin CSV có đường dẫn là '/content/drive/MyDrive/Colab DataMining/taptest.csv' và lưu nó vào biến original\_data. Đây là tập dữ liệu gốc mà chúng ta sẽ sử dụng để đào tạo và dự đoán.

Dùng X\_original = original\_data.drop('Action', axis=1): Lấy toàn bộ dữ liệu trừ cột có tên là 'Action' (cột mục tiêu) và lưu kết quả vào biến X\_original. Điều này tạo ra một DataFrame chứa các đặc trưng (features) mà chúng ta sẽ sử dụng để đào tạo mô hình.

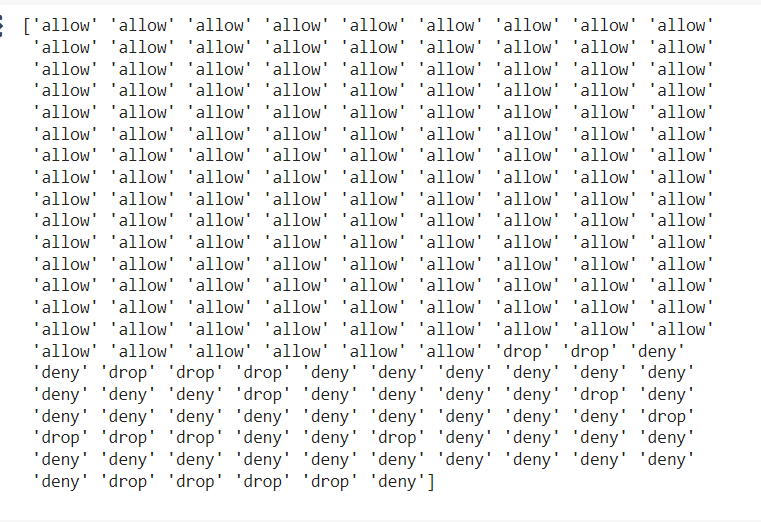
Dùng X\_new = X\_original.iloc[:200]: Chọn 200 dòng đầu tiên từ X\_original và lưu chúng trong biến X\_new. Điều này tạo ra một tập dữ liệu mới gồm 200 dòng đầu tiên.

Dùng y\_original = original\_data['Action']: Lấy cột có tên 'Action' từ original\_data và lưu nó trong biến y\_original. Đây là cột mục tiêu mà chúng ta sẽ cố gắng dự đoán.

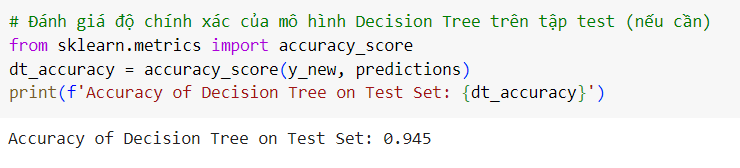
Dùng y\_new = y\_original.iloc[:200]: Tương tự như X\_new, chọn 200 giá trị đầu tiên từ y\_original và lưu vào biến y\_new. Điều này tạo ra một tập dữ liệu mục tiêu mới gồm 200 dòng đầu tiên.

Từ những khai báo trên ta dùng mô hình model để đào tạo mô hình Decision Tree trên dữ liệu X\_original và y\_original. Mô hình sẽ học từ dữ liệu này để có khả năng dự đoán cột mục tiêu 'Action'.( model.fit(X\_original, y\_original)) từ đó dùng dự đoán cột mục tiêu cho tập dữ liệu mới X\_new và lưu kết quả vào biến predictions.( predictions = model.predict(X\_new))  
cuối cùng dùng print(predictions): In kết quả dự đoán (giá trị của cột 'Action') cho 200 dòng đầu tiên của tập dữ liệu mới.

Kết quả nhận được là:



Cuối cùng ta tính toán và in độ chính xác của mô hình Cây quyết định trên tập kiểm tra và báo cáo rằng độ chính xác là 0,945. Điều này có nghĩa là mô hình Cây quyết định dự đoán chính xác 94,5% số mẫu trong bộ kiểm tra.



# Chương IV: KẾT QUẢ ĐẠT ĐƯỢC

* Kết quả đạt được:
* Tìm hiểu được môn học và tầm quan trọng của môn học.
* Tìm hiểu được đề tài và sơ bộ kiến thức liên quan.
* Hoàn thành công đoạn tiền xử lý dữ liệu và chuẩn hóa dữ liệu.
* Xây dựng được các mô hình.
* Mô hình đạt khoảng chính xác cao.
* Phân tích và đánh giá các thông số của mô hình.
* Test trên bộ data mới có độ chính xác cao.
* Khả năng nhận định lỗi và fix lại chương trình.
* Hiểu khái quát việc phân tích dữ liệu và đề xuất hướng đi.
* Khả năng làm việc nhóm, tương tác với nhóm.
* Hướng phát triển
* Tối ưu hóa hiệu suất mô hình.
* Liên tục cập nhật và theo dõi.
* Có thể vận dụng các mô hình học máy khác để phát triển thêm.
* Học máy tăng cường (Reinforcement Learning).
* Kiểm tra và đảm bảo tính an toàn của mô hình.

# Chương V: TÀI LIỆU THAM KHẢO

* Bài giảng, bài tập trên lớp
* Video trên You Tube
* Link\_1:<https://www.kaggle.com/code/adepvenugopal/firewall-log-analysis-eda>
* Link\_2:<https://github.com/morai8/Internet-Firewall-Data/blob/main/Tugas_Siskamancer_1.ipynb>
* Link\_3: <https://github.com/Shoaibdar/Internet-Firewall-Data>
* Link\_4: <https://www.kaggle.com/code/docxian/internet-firewall-analysis>
* Ngoài ra còn có các link Kaggle, Githup, Google khác.
* Link\_source\_code:<https://colab.research.google.com/drive/1sAYRDm_EhI5y2MM2OkJ-TZU-Pn9PcE0p#scrollTo=8MhAkDyYnXVQ>