**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN II**

-----

|  |
| --- |
|  |
|  |  |

**IOT VÀ ỨNG DỤNG**

**BÁO CÁO ĐỀ TÀI**

**TÌM HIỂU VỀ HỆ THỐNG DEEP LEARNING ĐƯỢC TỐI ƯU HÓA DỰA TRÊN PHÂN TÍCH THỐNG KÊ PHÁT HIỆN XÂM NHẬP**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GVHD: Đàm Minh Lịnh**   |  |  | | --- | --- | | Nguyễn Minh Thuận | N19DCCN202 | | Nguyễn Xuân Thường | N19DCCN206 | | Lớp: D19CQCNPM02 | Tên nhóm: Nhóm 38 | |  | | |

**TPHCM, tháng 12 năm 2022**

Mục Lục

**[1.](#_Toc123086472)****[SƠ LƯỢC ĐỀ TÀI:](#_Toc123086472)** [4](#_Toc123086472)

[1.1 Tóm tắt: 4](#_Toc123086473)

[1.2 Bảng viết tắt: 5](#_Toc123086474)

[1.3 Giới thiệu sơ lược: 5](#_Toc123086475)

[1.4 Cách tiếp cận: 5](#_Toc123086476)

**[2.](#_Toc123086477)****[PHƯƠNG PHÁP THỰC HIỆN:](#_Toc123086477)** [6](#_Toc123086477)

[2.1 Khái quát một số khái niệm: 6](#_Toc123086478)

[2.2 Tìm hiểu và phân tích dataset NSL – KDD 7](#_Toc123086479)

[2.3 Tiền xử lý dữ liệu: 7](#_Toc123086480)

[2.4 Trích chọn thuộc tính: 7](#_Toc123086481)

[2.5 Phân lớp theo mô hình: 7](#_Toc123086482)

**[3.](#_Toc123086483)****[DATSET NSL - KDD:](#_Toc123086483)** [7](#_Toc123086483)

[3.1 Giới thiệu về dataset NSL - KDD: 7](#_Toc123086484)

[3.2 Data file: 7](#_Toc123086485)

[3.3 Số lượng đặc trưng: 7](#_Toc123086486)

[3.4 So sánh với KDD’99: 8](#_Toc123086487)

[3.5 Nhìn nhận thống kê: 8](#_Toc123086488)

[3.6 Thống kê bản ghi trong 2 tập train và test: 8](#_Toc123086489)

**[4.](#_Toc123086490)****[PHÁT HIỆN BẤT THƯỜNG SỬ DỤNG MÔ HÌNH DEEP AUTOENCODER:](#_Toc123086490)** [14](#_Toc123086490)

[4.1 Tiền xử lý dữ liệu: 14](#_Toc123086491)

[i. Outlier Analysis (phân tích giá trị ngoại lai): 14](#_Toc123086492)

[ii. One hot encoding: 14](#_Toc123086493)

[iii. Quy trình tiền xử lý dữ liệu trong bài báo cáo: 15](#_Toc123086494)

[4.2 Trích chọn thuộc tính: 17](#_Toc123086495)

[4.3 Khái quát mô hình Deep AutoEncoder: 18](#_Toc123086496)

[4.4 Phân lớp theo mô hình Deep AE: 21](#_Toc123086497)

**[5.](#_Toc123086498)****[ĐÁNH GIÁ MÔ HÌNH VÀ KẾT QUẢ THỰC NGHIỆM:](#_Toc123086498)** [22](#_Toc123086498)

[5.1 Các hệ số đánh giá: 22](#_Toc123086499)

[5.2 Đánh giá model qua các hệ số: 23](#_Toc123086500)

[Với tấn công dạng DoS: 23](#_Toc123086501)

[Với tấn công dạng Probe: 24](#_Toc123086502)

[Với tấn công dạng U2R: 24](#_Toc123086503)

[Với tấn công dạng R2L: 24](#_Toc123086504)

[5.3 Nhận xét kết quả: 26](#_Toc123086505)

**[6.](#_Toc123086506)****[KẾT LUẬN:](#_Toc123086506)** [26](#_Toc123086506)

**[7.](#_Toc123086507)****[TÀI LIỆU THAM KHẢO:](#_Toc123086507)** [26](#_Toc123086507)

1. **SƠ LƯỢC ĐỀ TÀI:**
   1. Tóm tắt:

Ngày nay, Attackers thì ngày càng thông minh và tinh vi. nếu không có cách khác phục thì mức độ thiệt hại cao. Khi một quốc gia chặn dữ liệu tài chính được mã hóa bị tấn công thì hậu quả rất đáng sợ. Do đó, các hệ thống an ninh mạng thông minh đã trở nên quan trọng để cải thiện khả năng bảo vệ chống lại các mối đe dọa độc hại. Tuy nhiên, khi các cuộc tấn công bằng phần mềm độc hại tiếp tục gia tăng đáng kể về số lượng và độ phức tạp, việc phát hiện và giảm thiểu mối đe dọa trở nên khó khăn hơn bao giờ hết đối với các công cụ phân tích truyền thống. Vì vậy, chúng ta cần đề xuất một hệ thống học sâu được tối ưu dựa trên phân tích thống kê sáng tạo để phát hiện xâm nhập. Hệ thống phát hiện xâm nhập **([IDS](https://bizflycloud.vn/tin-tuc/chi-can-5-phut-de-hieu-ids-la-gi-ids-so-voi-ips-va-tuong-lua-20200725112126311.htm" \l ":~:text=IDS%20l%C3%A0%20vi%E1%BA%BFt%20t%E1%BA%AFt%20c%E1%BB%A7a,v%E1%BB%87%20h%E1%BB%87%20th%E1%BB%91ng%20th%C3%B4ng%20tin." \t "_blank)) - Intrusion Detection System** được đề xuất trích xuất các tính năng được tối ưu hóa và tương quan hơn bằng cách sử dụng các phương pháp phân tích thống kê và trực quan hóa dữ liệu lớn, tiếp theo là bộ mã hóa tự động sâu **(AE) - Auto Encoder** để phát hiện mối đe dọa tiềm ẩn. Cụ thể, một mô-đun tiền xử lý sẽ loại bỏ các giá trị không hợp lệ và chuyển đổi các biến phân loại thành **one-hot-encoder vectors.** *“The feature extraction module (trích xuất đặc trưng) discards (loại bỏ) features with null values (những giá trị null)”* **lớn hơn 80%** và chọn *“the most significant features”* làm “*input”* cho *“the deep autoencoder model trained”* theo cách “*a greedy-wise manner*”. **Dataset NSL - KDD** (là phiên bản được cải thiện từ KDD) từ Viện An ninh mạng Canada được sử dụng làm chuẩn để đánh giá tính khả thi và hiệu quả của kiến trúc được đề xuất. Kết quả mô phỏng chứng minh tiềm năng của hệ thống IDS được đề xuất của chúng tôi để cải thiện khả năng phát hiện xâm nhập so với các phương pháp tiên tiến nhất hiện có và **đạt độ chính xác tỷ lệ lên đến 87%.** Công việc đang thực hiện bao gồm tối ưu hóa hơn nữa và đánh giá theo thời gian thực (real time) về IDS được đề xuất của chúng tôi.

Bài báo cáo này được dựa trên tài liệu nguyên cứu có tên “Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection” [1].

Key words: Cybersercurity, Deep Learning, Auto Encoder, NSL – KDD

* 1. Bảng viết tắt:

|  |  |
| --- | --- |
| DL | Deep Learning |
| IDS | Intrusion Detection System |
| AE | AutoEncoder |
| MLP | Multi-layer Perceptron |
| DNN | Deep neural Network |
| RNN | Recurrent Neural Network |

* 1. Giới thiệu sơ lược:

Tính không đồng nhất (heterogeneity) của dữ liệu trong các mạng hiện đại và nhiều giao thức mới đã làm cho việc phát hiện xâm nhập trở nên phức tạp và thách thức hơn. Deep Learning khắc phục đc những hạn chế. DL đã cho thấy đạt được hiệu suất ở cấp độ con người trong một số ứng dụng trong thế giới thực (nhận dạng hình ảnh, chăm sóc sức khỏe, phân tích segment), nên gần đây, các nhà nghiên cứu đã đề xuất một số thuật toán an ninh mạng mới dựa trên Deep Learning. Các giải pháp dựa trên Deep Learning có khả năng phân tích “Big Data” một cách hiệu quả và xác định các cấu trúc thời gian trong các chuỗi dài phức tạp trong thời gian thực.

Trong bài báo này [1], một hệ thống Deep Learning được tối ưu hóa dựa trên phân tích thống kê sáng tạo để phát hiện xâm nhập được đề xuất. Cụ thể, một bộ mã hóa (AE - AutoEncoder) dựa trên thống kê được phát triển để phát hiện các mẫu lưu lượng truy cập bình thường (normal) và bất thường (abnormal). Khung đề xuất đã được đánh giá bằng cách sử dụng bộ dữ liệu NSL-KDD với (phiên bản cập nhật của bộ dữ liệu KDD Cup 99 - KDD99 trước đó) và bao gồm ba module chính: **tiền xử lý dữ liệu** - *“nó loại bỏ các giá trị không hợp lệ và chuyển đổi các biến phân loại thành các vector thành dạng One-Hot-Encoding”*; **trích xuất đặc trưng** – *“ nó chọn các tính năng tương quan nhất và loại bỏ các tính năng có giá trị null lớn hơn 80%”*; **phân loại** – *“ AE và Mạng MLP (Multi-layer Perceptron) được phát triển để phân loại các danh mục khác nhau của bộ dataset NSL-KDD (Normal, DoS, R2L, Probe)”*.

Autoencoder và mạng MLP được đề xuất cũng được so sánh với bốn mô hình gần đây dựa trên bộ dữ liệu NSL-KDD. Kết quả thử nghiệm cho thấy mạng AE vượt trội so với tất cả các phương pháp khác, đạt độ chính xác 87%.

* 1. Cách tiếp cận:

Alrawashdeh và cộng sự [5] đã phát triển một mạng niềm tin sâu sắc (DBN) dựa trên các module Restricted Boltzmann Machine (RBM), theo sau là multi-class softmax layer. Mô hình đã được thử nghiệm trên 10% bộ dữ liệu thử nghiệm KDD99 và đạt được độ chính xác phát hiện lên tới 97,9% với tỷ lệ cảnh báo sai là 2,47%.

Tang và cộng sự [6] đã đề xuất một phương pháp học sâu để phát hiện sự bất thường dựa trên luồng trong môi trường Software Defined Networking (SDN). Các tác giả đã phát triển Deep Neural Network (DNN) với 3 hid, được đào tạo trên dataset NSL-KDD để chỉ thực hiện phân lớp nhị phân “binary classification” (normal, anomaly) bằng sáu đặc trưng cơ bản. Tuy nhiên, độ chính xác được báo cáo là 75,75%.

Kim và một nhóm người khác [7] đã phát triển một kiến ​​trúc DNN khác (với 4 hidden layer và 100 hidden units) và được đào tạo tối ưu hóa bằng *thuật toán tối ưu hóa adam*. Tuy nhiên, hiệu suất được đo bằng bộ dữ liệu KDD99.

Javaid và một nhóm người khác [8] đã đề xuất phương pháp **self-taught learning** (STL) dựa trên các bộ mã hóa tự động thưa thớt để phát hiện (abnormal) bất thường. Bộ dữ liệu NSLKDD được sử dụng làm điểm chuẩn để định lượng hiệu suất. đã đề xuất Mạng Recurrent Neural Network (RNN) để phát hiện bất thường bằng cách sử dụng cùng một điểm chuẩn (bechmark), khẳng định độ chính xác lần lượt là 83,28% và 81,29% trong phân loại binary và multiclass.

Shone và cộng sự [9] đã đề xuất mô hình **non-symmetric deep auto-encoder** (NDAE) để phát hiện xâm nhập được thử nghiệm trên cả bộ dữ liệu KDD99 và NSL - KDD, đạt tỷ lệ chính xác 5 lớp tương ứng lên tới 97,85% và 85,42%.

Mới đây, Diro và nhóm người khác [10] đã đề xuất một kiến ​​trúc DL mới dựa trên **autoencoders** để phát hiện tấn công trong *fog-to-things computing*, sử dụng NSL - KDD. Tuy nhiên, việc đánh giá chỉ giới hạn ở phát hiện nhị phân (normal, anomaly).

Trong bài báo này, chúng tôi đề xuất một phương pháp học sâu dựa trên thống kê đổi mới để phát hiện xâm nhập mạng. Bộ dữ liệu NSL-KDD được sử dụng để ước tính độ tin cậy của mô hình đối với phân loại nhị phân và đa lớp và những hạn chế nói trên đã được giải quyết.

1. **PHƯƠNG PHÁP THỰC HIỆN:**
   1. Khái quát một số khái niệm:
   2. Tìm hiểu và phân tích dataset NSL – KDD
   3. Tiền xử lý dữ liệu:
      1. Xử lý giá trị ngoại lại:

Outlier analysis (phân tích giá trị ngoại lai): Việc loại bỏ các giá trị ngoại lai khỏi tập dữ liệu trước khi thực hiện chuẩn hóa dữ liệu là một nhiệm vụ thiết yếu. Trong nghiên cứu này, công cụ ước tính Độ lệch tuyệt đối trung bình - Median Absolute Deviation (MAD) được sử dụng để phát hiện các giá trị ngoại lai.

Công thức: MAD = C ∗ median (xij − |median (xij)|)

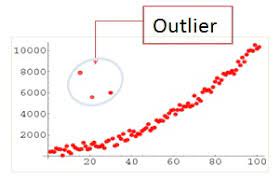
* + - * + Trong đó: C=1,4826 là hằng số
        + xij được coi là ngoại lệ khi xij > k ∗ MAD (với k=10).
        + Kích thước ban đầu của tập train và test đã giảm từ 125973 xuống 85421 và từ 22544 đến 11925.

Figure 12: ảnh minh họa cho outlier analysis

* + 1. One hot encoding

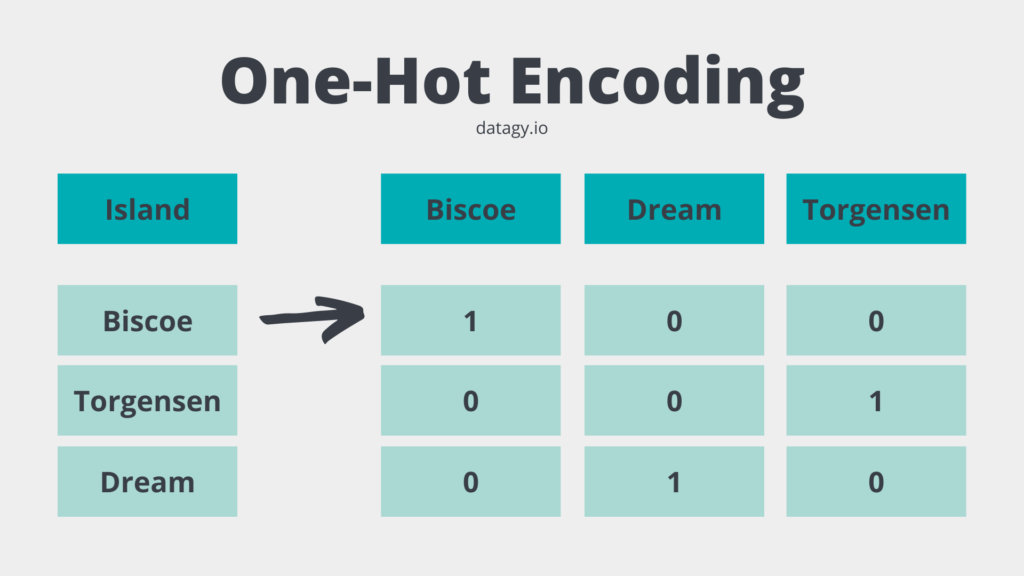
One hot encoding: chuyển đổi dữ liệu dạng categorical sang dạng numberic. Vì các đặc trưng x2, x3, x4 (protocol type, service and flag) bao gồm các giá trị phân loại, các đặc trưng này đã được chuyển đổi thành one hot encoded vectơ. Ví dụ, đặc tính loại giao thức bao gồm 3 thuộc tính: tcp, udp và icmp, và được biểu diễn lần lượt là (1,0,0), (0,1,0), (0,0,1). Tương tự, các service and flag features được biểu thị bằng các giá trị nhị phân. Procedure maps này các 41-dimensional features thành 122-dimensional features: 38 continuous và 84 Statistical Analysis Driven Optimized DL System cho ID (phát hiện xâm nhập - Intrusion Detection).

Figure 13: minh họa one hot encoding

* 1. Trích chọn thuộc tính
  2. Phân lớp theo mô hình:

1. **DATSET NSL - KDD:**
   1. Giới thiệu về dataset NSL - KDD:

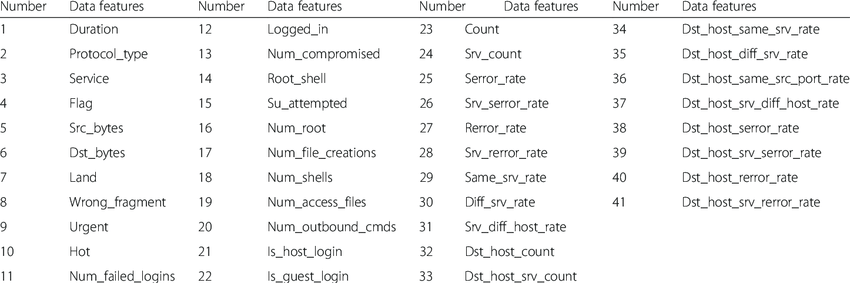
Bộ dataset NSL – KDD được ra đời nhằm giải quyết những vấn đề của KDD’99. Số lượng record trong tập huấn luyện và kiểm tra của bộ NSL-KDD là hợp lý. Nó phù hợp với các phương pháp phát hiện xâm nhập khác nhau.

* 1. Data file:

|  |  |
| --- | --- |
| **Data file** | **Ý nghĩa** |
| KDDTrain+.ARFF | Tập huấn luyện NSL-KDD đầy đủ với các nhãn nhị phân ở định dạng ARFF |
| KDDTrain+.TXT | Bộ đào tạo NSL-KDD đầy đủ bao gồm nhãn loại tấn công và mức độ khó ở định dạng CSV |
| KDDTrain+\_20Percent.ARFF | Tập con 20% của tệp KDDTrain+.arff |
| KDDTrain+\_20Percent.TXT | Tập con 20% của tệp KDDTrain + .txt |
| KDDTest+.ARFF | Bộ kiểm tra NSL-KDD đầy đủ với các nhãn nhị phân ở định dạng ARFF |
| KDDTest + .TXT | Bộ kiểm tra NSL-KDD đầy đủ bao gồm nhãn kiểu tấn công và mức độ khó ở định dạng CSV. |
| KDDTest–21.ARFF | Tập hợp con của tệp KDDTest+.arff không bao gồm các bản ghi có độ khó 21/21 |

* 1. Số lượng đặc trưng:

Có 38 feature (numeric) và 3 feature (categorical) là protocol\_type, service, flag.



* 1. So sánh với KDD’99:

Đã loại bỏ các bản ghi dư thừa, trùng lặp trong huấn luyện được đề xuất => do đó, hiệu suất của người học không bị sai lệch bởi các phương pháp có tỷ lệ phát hiện tốt hơn trên các bản ghi thường xuyên.

Số lượng bản ghi trong tập huấn luyện và tập kiểm tra là hợp lý, giúp cho việc chạy thử nghiệm trên tập hoàn chỉnh có thể thực hiện được mà không cần phải

* 1. Nhìn nhận thống kê:

Số lượng lớn các bản ghi dư thừa, khiến thuật toán học bị thiên về các bản ghi thường xuyên và do đó ngăn chúng học các bản ghi không thường xuyên thường gây hại hơn cho mạng như U2R và tấn công R2L.

* 1. Thống kê bản ghi trong 2 tập train và test:

Thống kê các bản ghi dư thừa trong **tập train**

|  |  |  |  |
| --- | --- | --- | --- |
| Thống kê các bản ghi dư thừa trong **tập train** | | | |
|  | **Original records** | **Distinct records** | **Reduction rate** |
| Attacks | 3,925,650 | 262,178 | 93.32% |
| Normal | 972,781 | 812,814 | 16.44% |
| Total | 4,898,431 | 1,074,992 | 78.05% |

|  |  |  |  |
| --- | --- | --- | --- |
| Thống kê các bản ghi dư thừa trong **tập test** | | | |
|  | **Original records** | **Distinct records** | **Reduction rate** |
| Attacks | 250,436 | 29,378 | 88.26% |
| Normal | 60,591 | 47,911 | 20.92% |
| Total | 311,027 | 77,289 | 75.15% |

Thống kê các bản ghi dư thừa trong **tập test**

Cấu trúc của dataset NSL-KDD: được sắp xếp thành tập train gồm 125973 mẫu (KDDTrain+) và tập test gồm 22544 mẫu (KDDTest+). Bộ dữ liệu này có xi (i = 1, 2, ...41) đặc trưng với 38 numberic (dạng số) và 3 categorical (dạng danh mục). Đặc biệt, *protocol type, service, flag* (x2, x3, x4) đại diện cho 3 giá trị categorical.

|  |  |
| --- | --- |
| Các nhãn theo từng loại tấn công DoS, R2L, U2R, Probe trên tập train | |
| **Attack class** | **Attack type** |
| Dos | back, land, neptune, pod, smurf, teardrop |
| R2L | ftp write, guess passwd, imap, multihop, phf, spy, warezclient, warezmaster |
| U2R | buffer overflow, loadmodule, perl, rootki |
| Probe | ipsweep, nmap, portsweep, satan |

Các nhãn theo từng loại tấn công DoS, R2L, U2R, Probe trên tập train.

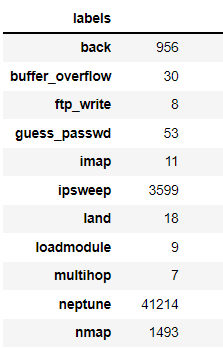
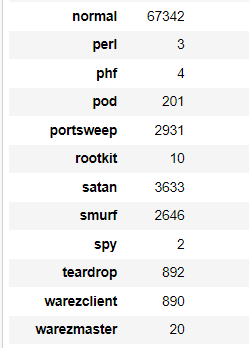
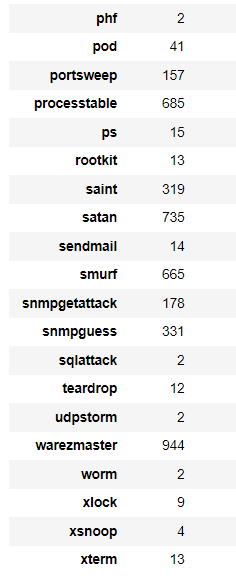
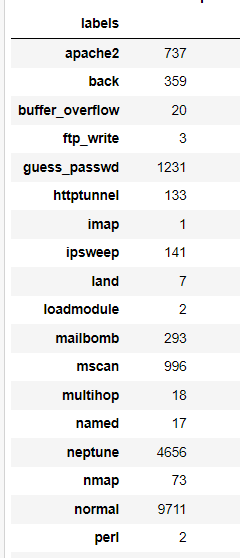


Figure 3: Số lượng mẫu theo từng loại của tập KDDTest+

Figure 4: Số lượng mẫu theo từng loại của tập KDDTest+

Figure 1: số lượng mẫu theo từng loại của tập KDDTrain+

Figure 2: Số lượng mẫu theo từng loại của tập KDDTrain+

Thống kế số lượng mẫu theo từng loại trên 2 tập Train và Test của NSL – KDD

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Thống kế số lượng mẫu theo từng loại trên 2 tập Train và Test của NSL - KDD | | | | | | |
| **NSL - KDD** | **Total** | **Normal** | **Dos** | **Probe** | **R2L** | **U2R** |
| KDDTrain+ | 125972 | 67342 | 45927 | 11656 | 995 | 52 |
| KDDTest+ | 22543 | 9711 | 5741 | 2199 | 1106 | 37 |

 Kích thướt file train:

Figure 5: Kích thướt file train

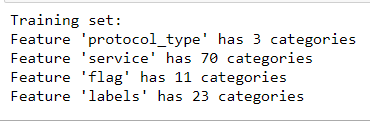
 Số lượng từng đặc trưng danh mục của tập train:

Figure 6: Số lượng từng đặc trưng danh mục của tập train

Đọc file dataset với pandas:

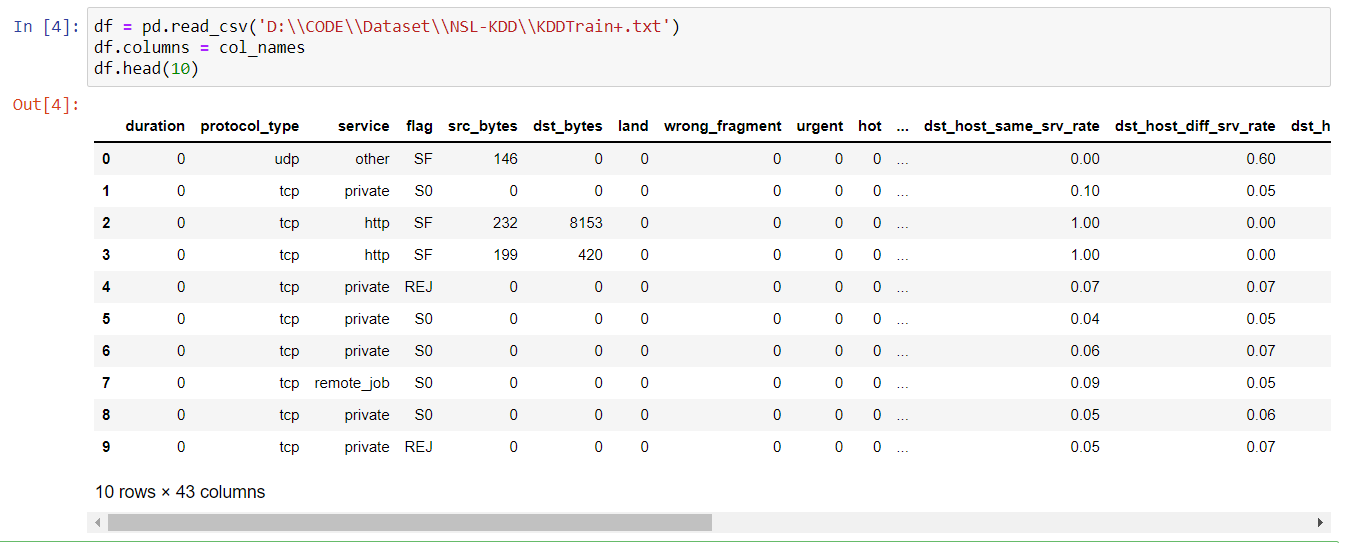
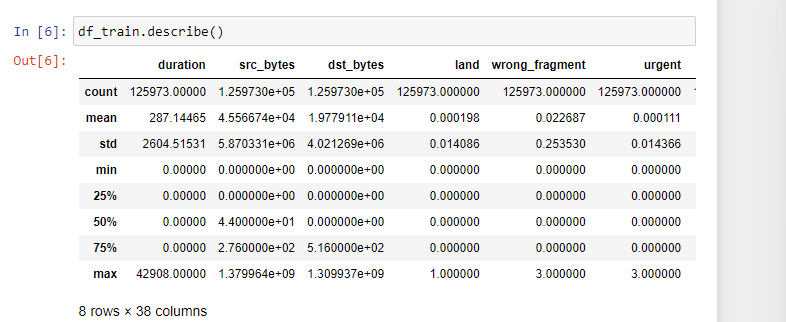


Figure 7: Đọc file dataset với pandas

Mô tả dataset với pandas: 

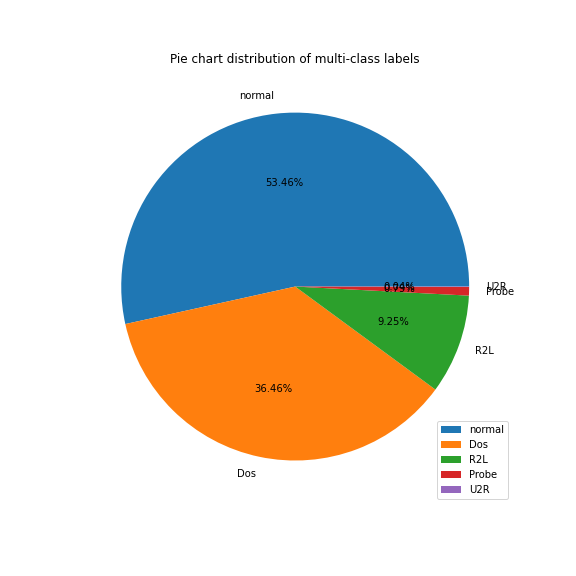
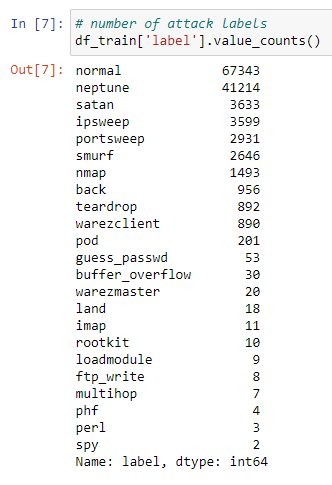
Figure 9: Mô tả dataset với pandas

Figure 8: Phân phối của các nhãn DoS, Probe, R2L, U2R

Số lượng nhãn tấn công của dataset:

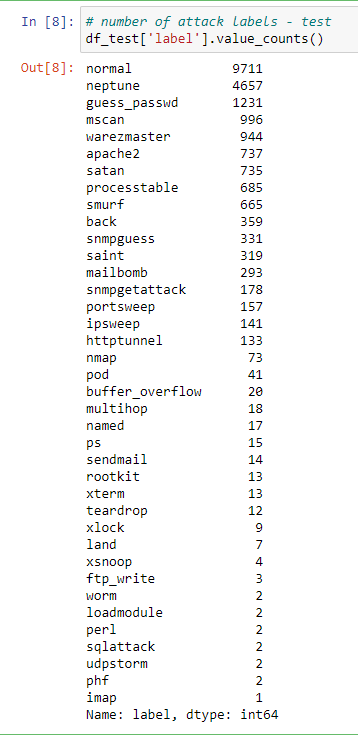
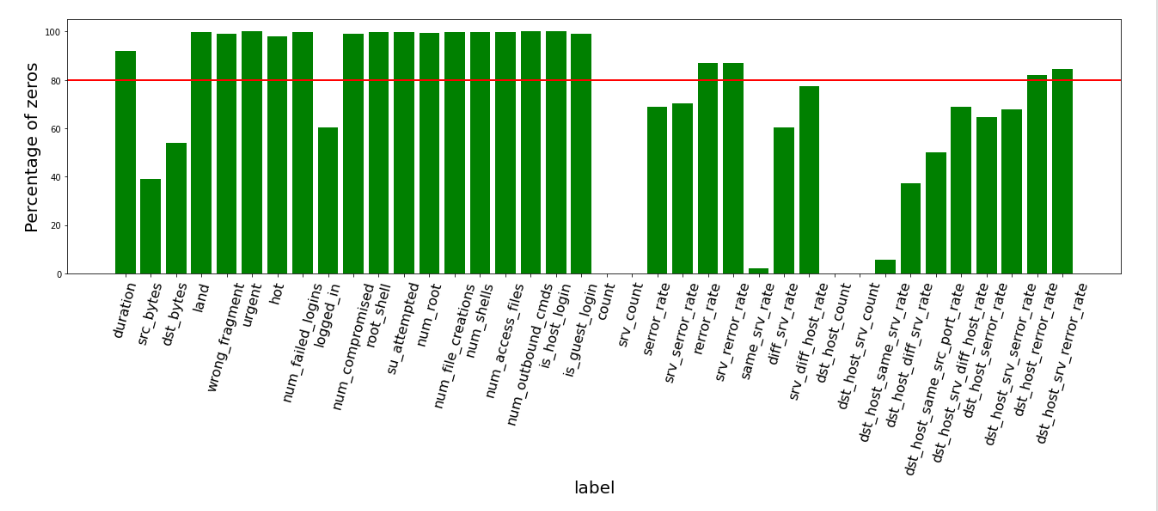


Figure 10: Số lượng nhãn tấn công của file train

Figure 11: Số lượng nhãn tấn công của tập test

1. **PHÁT HIỆN BẤT THƯỜNG SỬ DỤNG MÔ HÌNH DEEP AUTOENCODER:**
   1. Tiền xử lý dữ liệu:

Quy trình tiền xử lý dữ liệu trong bài báo cáo: Phân phối số lượng số 0 trong mỗi tính năng số của tập huấn luyện. Các tính năng có giá trị null lớn hơn 80% được mô tả bằng màu đỏ và bị loại bỏ khỏi phân tích.

Đọc dữ liệu từ file KDDTrain+.txt.

Figure 14: percentage Of Zeros

Sau khi gán tên cho từng cột, chúng ta nhận thấy cột với tên ‘level’ là dư thừa không cần thiết cho quá trình train ***vì vậy tiến thành bỏ toàn bộ giá trị cột này.*** Tiếp tục thay đổi tên các nhãn tấn công ứng với từng loại (DoS, R2L, Probe, U2R).

Data Scaling …

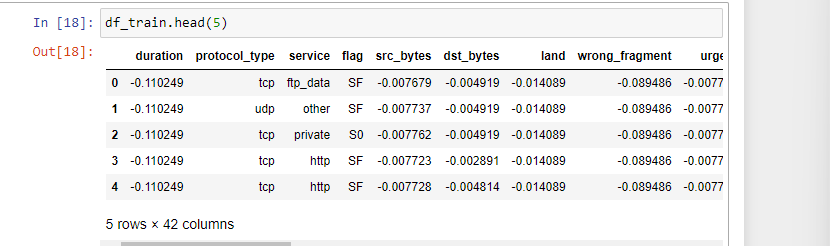
Sau khi hiệu chỉnh nhãn, loại bỏ cột không cần thiết cho quá trình train data, chúng ta sẽ thực hiện chuẩn tất cả dữ liệu dạng số (numeric) với cách thức “StandardScaler” được sự hỗ trợ từ thư viện Scikit learn.

Figure 15: minh họa quá trình Data Scaling

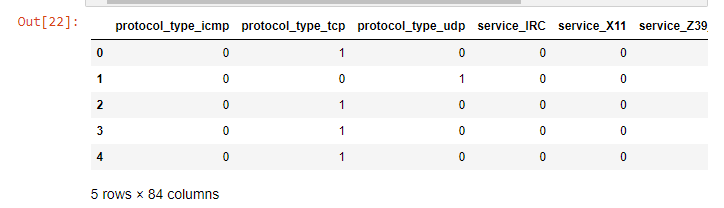
Với các đặc trưng ở dạng danh mục (categorical feature) thì xử lý chúng với **“one hot encoding”.**

Figure 16: minh họa quá trinh one hot encoding trong báo cáo này.

Đặt tên giả cho các đặc trưng danh mục:

Xử lý nhãn với LabelEncoder trong thư viện Scikit Learn, nhớ fit và transform quá trình này lại. Bước tiếp theo chúng ta sẽ trích chọn đặc trưng cho dataset.

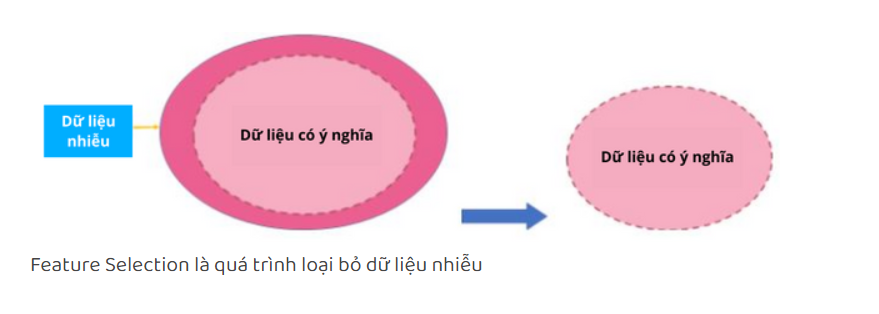
* 1. Trích chọn thuộc tính:

\* Tại sao cần lựa chọn các đặc trưng?

Các mô hình Machine Learning luôn hoạt động theo một nguyên tắc đơn giản: học từ tất cả các dữ liệu được đưa vào. Điều này cũng có nghĩa là nếu chúng ta đưa các dữ liệu rác vào để hệ thống học tập, kết quả đầu ra cũng sẽ là rác, không chính xác như chúng ta mong muốn.

Do đó, để đào tạo một mô hình, chúng ta cần thu thập một lượng lớn dữ liệu có chất lượng để máy tính có thể học hỏi tốt hơn. Thông thường, có 1 phần trong dữ liệu sẽ không có ý nghĩa, không đóng góp quá nhiều vào hiệu suất hoạt động của mô hình Machine Learning. Hơn nữa, việc có quá nhiều dữ liệu có thể làm chậm quá trình đào tạo và khiến mô hình huấn luyện chậm hơn, mô hình cũng có thể học từ các dữ liệu rác này dẫn đến kết quả trả về sau đó không chính xác.

Figure 17: minh họa quá trình trích chọn thuộc tính.

Trích chọn đặc trưng với chọn theo hệ số Độ tương quan (Correlation). Sử dụng độ tương quan giữa 2 hay nhiều biến cũng là một cách hay để loại bỏ những feature có độ tương quan thấp. Việc loại bỏ các feature có độ tương quan cao với nhau giúp mô hình linear hoạt động tốt hơn, tránh bias giữa các features. *Tìm các thuộc tính có tương quan hơn 0,5 với thuộc tính nhãn tấn công được mã hóa.* Nhớ sau khi loại bỏ những giá trị dưới ngưỡng 0.5, thì phải trả lại nhãn phân lớp theo đúng thứ tự.

Trong quá trình này có sinh ra giá trị NAN, tiếp tục xử lý giá trị bị thiếu này bằng SimpleImputer với thư viện Scikit Learn.

Biểu đồ giá trị giao động sau khi trích chọn thuộc tính:

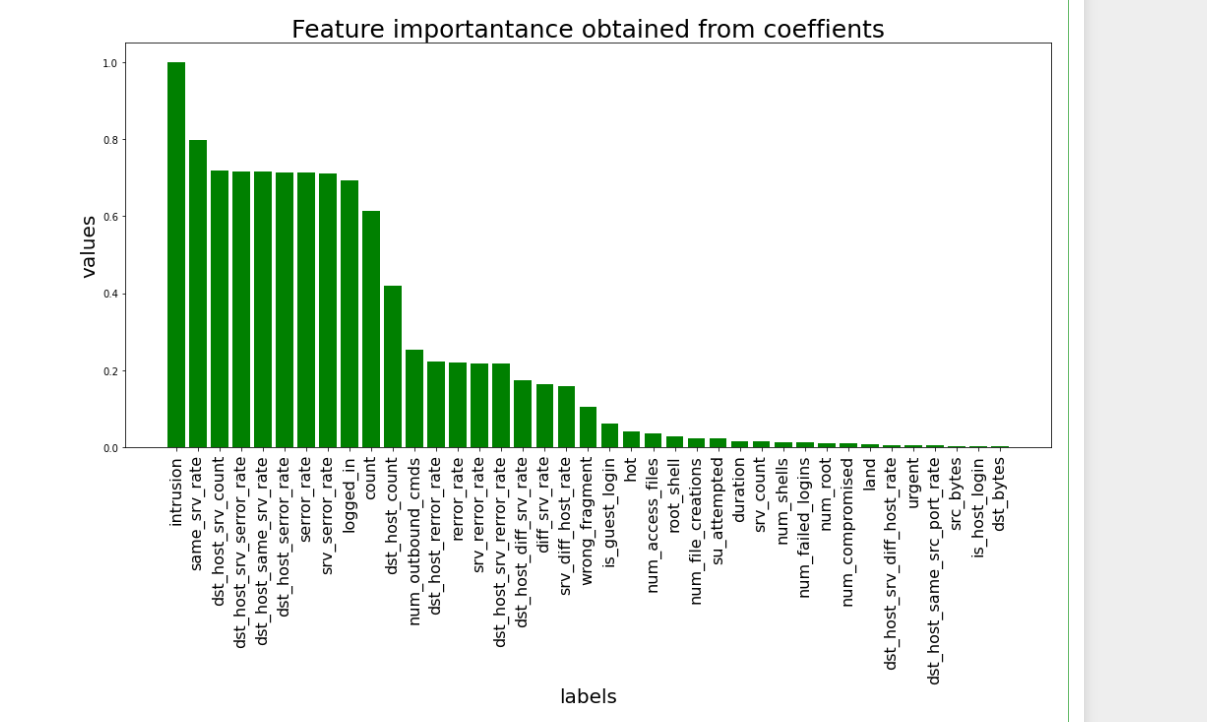
Đến gây là gần hoàn thiện quá trình tiền xử lý dữ liệu và trích chọn đặc trưng. Tiếp theo “Join” nối đặc trưng đã chọn với những đặc trưng tdanh mục được one-hot-encoded thành một dataframe duy nhất.

Figure 18: Feature importance obtained from coeffients

Lưu lại toàn bộ quá trình này ra file csv để phục vụ cho quá trình tiếp theo là phân lớp với mô hình Deep AutoEncoder.

* 1. Khái quát mô hình Deep AutoEncoder:

Mô hình Deep AE: Deep AE Classifier: Deep AE Bộ mã hóa tự động là một loại thuật toán học tập không giám sát, thường được sử dụng cho mục đích giảm kích thước. Cấu hình tiêu chuẩn của AE bao gồm một lớp đầu vào, một lớp đầu ra và một lớp ẩn, như thể hiện trong. Nó nén dữ liệu đầu vào x thành kích thước h thấp hơn thông qua quá trình mã hóa:

**(Encoding) *h* = *g* (*xw* + *b*)**

Trong đó x, w, b lần lượt là vectơ đầu vào, ma trận trọng số, vectơ độ lệch và g là hàm kích hoạt. Sau đó, nó cố gắng tái tạo lại cùng một bộ đầu vào (x) từ biểu diễn nén (h) thông qua quá trình giải mã:

**(Decoding) *x* = *g* (*hwT* + *b*)**

Kiến trúc của bộ phân loại sâu AE được hiển thị trong hình dưới đây, đặc trưng đã được trích xuất là đầu vào của AE lớp ẩn duy nhất đã nén không gian đầu vào từ 100 thành 50 tính năng tiềm ẩn. Ở giai đoạn này, AE được đào tạo bằng cách học không giám sát thông qua thuật toán gradient liên hợp được chia tỷ lệ, cho 100 lần lặp lại. Hàm truyền tuyến tính bão hòa (The *saturating linear transfer function*) (*g*(*z*) = 0 if *z ≤* 0, *g*(*z*) = *z* if 0 *< z <* 1, *g*(*z*) = 0 if *z ≥* 1) và hàm truyền tuyến tính (the *linear transfer function*) (*g*(*z*) = *z*) được sử dụng cho các hoạt động mã hóa và giải mã (encoding => decoding).

Việc xây dựng lại các tính năng đầu vào (x) được đo lường thông qua hệ số lỗi bình phương trung bình (MSE). AE đề xuất đã đạt được lỗi tái tạo là 0,0083. Sau đó, 50 tính năng đã nén được đưa vào lớp đầu ra softmax được đào tạo bằng phương pháp học có giám sát để thực hiện tác vụ phát hiện nhiều lớp.

Cuối cùng, toàn bộ mạng (AE+softmax) được đào tạo bằng phương pháp học có giám sát (thuật toán lan truyền ngược) để cải thiện hiệu suất phân loại (phương pháp tinh chỉnh).

Quá trình đào tạo bị dừng khi hàm mất cross entropy bão hòa. Trong nghiên cứu này, sự hội tụ được quan sát sau 300 lần lặp.

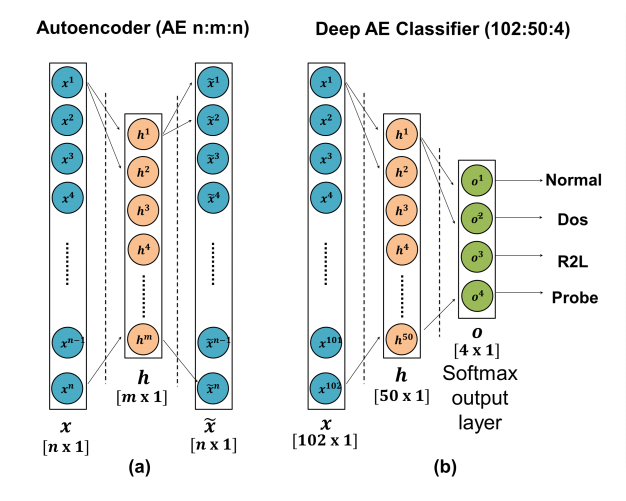
Khái quát mô hình AutoEncoder:

Figure 19: Khái quát mô hình AutoEncoder

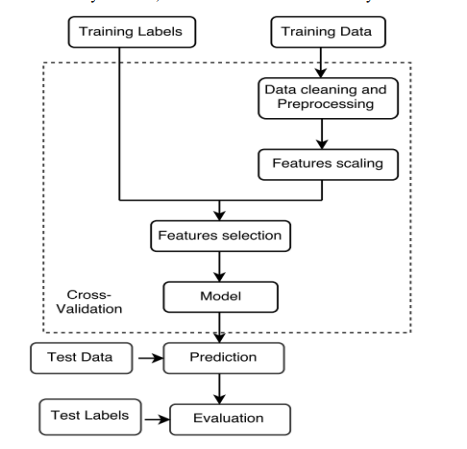
* 1.  Phân lớp theo mô hình Deep AE:

Figure 20: minh họa toàn bộ quá trình thực hiện.

Sau khi thực hiện toàn bộ các bước:

Step 1: Data preprocessing => Step 2: Feature Scaling => Step 3: Feature Selection => Step 4: Build the model

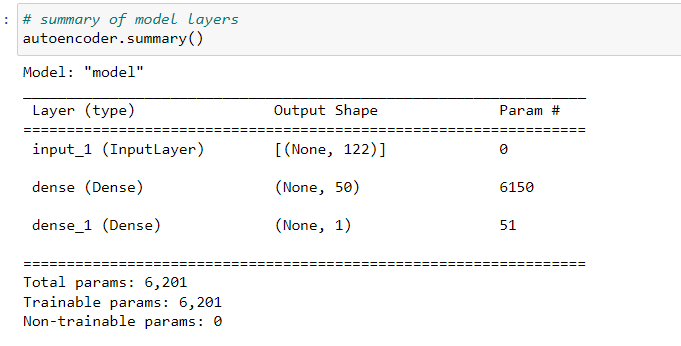
Tại đây, chúng ta sẽ thực hiện Build model Deep AutoEncoder Classifier.

Figure 21: model Summary

Nhận thấy model được xây dựng chỉ dừng lại mức độ cơ bản chưa có sự cải thiện nhiều các hệ số. Vì bài báo cáo này kết quả mà tác giả muốn hướng đến là hiểu quá trình tiền xử lý, cách xây dựng model Deep AE Classifier, một vài hệ số sẽ cải thiện được kết quả như: “Drop out”, đi qua nhiều hơn một lớp Dense, … và những hệ số được coi là có khả năng cải thiện model khi thực hiện quá trình training.

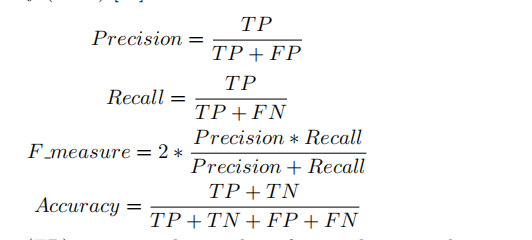
1. **ĐÁNH GIÁ MÔ HÌNH VÀ KẾT QUẢ THỰC NGHIỆM:**
   1.  Các hệ số đánh giá:

Figure 23: Công thức tính các hệ số đánh giá

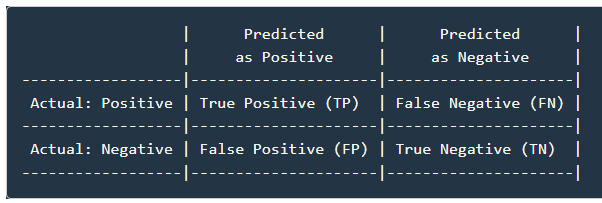
Trong những bài toán này, người ta thường định nghĩa lớp dữ liệu quan trọng hơn cần được xác định đúng là lớp Positive (P-dương tính), lớp còn lại được gọi là Negative (N-âm tính). Ta định nghĩa True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) dựa trên confusion matrix chưa chuẩn hoá như sau:

Figure 24: Ma trận nhầm lẫn với nhãn nhị phân (Confution matrix)

**True positives:**Các điểm Positive thực được nhận Đúng là Positive

**False positives:**Các điểm Negative thực được nhận Sai là Positive

**True negatives:**Các điểm Negative thực được nhận Đúng là Negative

**False negatives:**Các điểm Positive thực được nhận Sai là Negative

**Recall:**  Thể hiện khả năng phát hiện tất cả các postivie, tỷ lệ này càng cao thì cho thấy khả năng bỏ sót các điểm Positive là thấp.

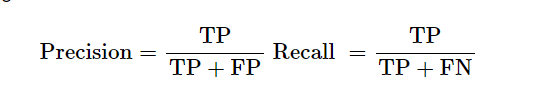
**Precision:**Thể hiện sự chuẩn xác của việc phát hiện các điểm Positive. Số này càng cao thì model nhận các điểm Positive càng chuẩn.

Figure 25: Công thức tính Precision và recall

**F1 score:** Là số dung hòa Recall và Precision giúp ta có căn cứ để lựa chọn model. F1 càng cao càng tốt ;).

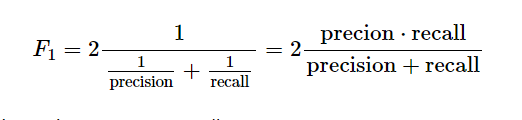
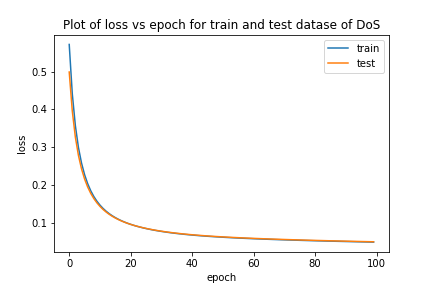
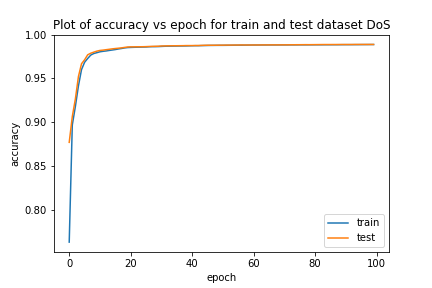
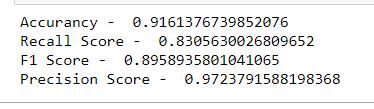
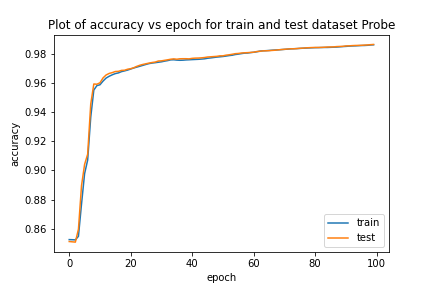


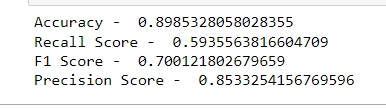
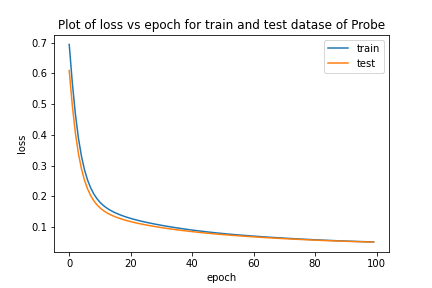
Figure 26: Công thức tính F1

* 1. Đánh giá model qua các hệ số:

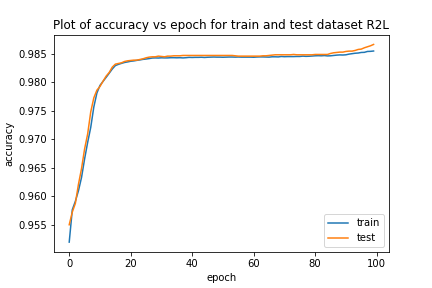
Với tấn công dạng DoS:

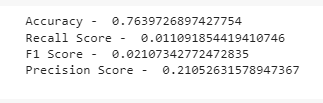
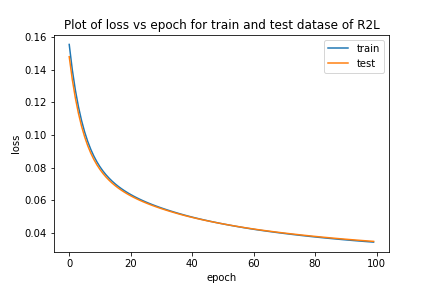
Với tấn Với tấn công dạng Probe:

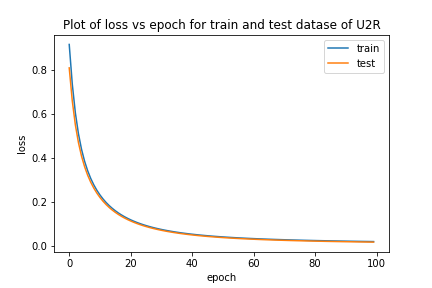
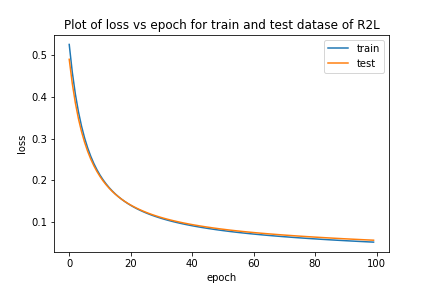
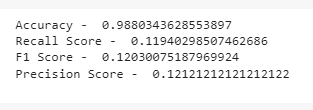




### Với tấn công dạng R2L:



Với tấn công dạng U2R:



* 1. Nhận xét kết quả:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atack Class** | **Deep AutoEncoder Classifier** | | | |
| **Accuracy** | **Pression** | **Recall** | **F\_measure** |
| DoS | 91% | 97% | 89% | 83% |
| Probe | 91% | 80% | 73% | 76% |
| R2L | 77% | 26% | 10% | 20% |
| U2R | 98% | 12% | 12% | 12% |

Theo bảng thông kê phía trên, ta thấy với mô hình Deep AE classifier cho ra kết quả Pression % từng loại mã tấn công khác nhau theo thứ tự giảm dần như sau: DoS, Probe, R2L, U2R. Với mã thuộc loại U2R có mức lệch khá cao do nó có số lượng mẫu trong dataset khá ít và ngược lại với mã DoS cho ra kết quả ổn định hơn vì nó có số lượng data nhiều hơn hẳn các mã còn lại.

1. **KẾT LUẬN:**

Trong bài báo này, chúng tôi đã giới thiệu một hệ thống DL được tối ưu hóa dựa trên thống kê để phát hiện xâm nhập. Bộ dữ liệu NSL-KDD được sử dụng làm điểm chuẩn để xác định các mẫu lưu lượng mạng bình thường và bất thường. Các tính năng tương quan nhất được trích xuất bằng các phương pháp thống kê và là đầu vào của trình phân loại AE sâu. Tính khả thi và hiệu quả của mô hình đề xuất được đánh giá bằng cách sử dụng các phép đo chính xác, thu hồi, đo lường F và độ chính xác. Đánh giá so sánh giữa bộ mã hóa tự động sâu được đề xuất với bộ phân loại MLP nông và các mô hình hiện đại cho thấy bộ phân loại AE sâu vượt trội hơn tất cả các phương pháp khác và đạt độ chính xác 87%. Các tính năng hoạt động bao gồm một hệ thống mạnh mẽ hơn có khả năng xử lý các hạn chế do bộ dữ liệu NSL-KDD thể hiện (các giá trị tính năng không nhất quán, các lớp không cân bằng) và khả năng mở rộng của nó trong các ứng dụng thời gian thực phát hiện xâm nhập theo ngữ cảnh.

1. **TÀI LIỆU THAM KHẢO:**

[1] Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection, Cosimo Ieracitano, Ahsan Adeel, Mandar Gogate, Kia Dashtipour, Francesco Carlo Morabito, Hadi Larijani, Ali Raza, and Amir Hussain, United Kingdonm.

[2] [Link download NSL - KDD Dataset](https://www.unb.ca/cic/datasets/nsl.html" \t "_blank).

[3] [Autoencoder là gì? Kiến trúc và cách tạo Autoencoder.](https://bizflycloud.vn/tin-tuc/autoencoder-la-gi-20220526165157229.htm" \l ":~:text=Autoencoder%20l%C3%A0%20m%E1%BB%99t%20lo%E1%BA%A1i%20m%E1%BA%A1ng,%C4%91%C3%B3%20m%E1%BB%99t%20c%C3%A1ch%20ch%E1%BA%B7t%20ch%E1%BA%BD.)

[4] [Mạng MLP (Multi-layer Perceptron) là gì? Nền tảng của Deep Learning.](https://gocnhinso.com/mang-mlp-multi-layer-perceptron-la-gi-nen-tang-cua-deep-learning/)

[5] Alrawashdeh, K., Purdy, C.: Toward an online anomaly intrusion detection system based on deep learning. In: Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on. pp. 195–200. IEEE (2016).

[6] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for network intrusion detection in software defined networking. In: Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on. pp. 258–263. IEEE (2016).

[7] Kim, J., Shin, N., Jo, S.Y., Kim, S.H.: Method of intrusion detection using deep neural network. In: Big Data and Smart Computing (BigComp), 2017 IEEE International Conference on. pp. 313–316. IEEE (2017).

[8] Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). pp. 21–26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2016).

[9] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence 2(1), 41–50 (2018).

[10] Abeshu, A., Chilamkurti, N.: Deep learning: The frontier for distributed attack detection in fog-to-things computing. IEEE Communications Magazine 56(2), 169– 175 (2018)

[11] Improving Performance of Autoencoder-based Network Anomaly Detection on NSL-KDD dataset, WEN XU, JULIAN JANG-JACCARD, AMARDEEP SINGH, YUANYUAN WEI and FARIZA SABRINA202.

[12] [Network-Intrusion-Detection-Using-Deep-Learning, Abhinav Bhardwaj, 2020 [12]](https://github.com/abhinav-bhardwaj/Network-Intrusion-Detection-Using-Machine-Learning).

[13] [Phương pháp lựa chọn feature trong Machine Learning](https://web888.vn/phuong-phap-lua-chon-feature-trong-machine-learning/).

[[14] Stop Using 0.5 as the Threshold for Your Binary Classifier, Eduardo Blancas, 2022.](https://towardsdatascience.com/stop-using-0-5-as-the-threshold-for-your-binary-classifier-8d7168290d44)

[[15] A Subset Feature Elimination Mechanism for Intrusion Detection System, (Herve Nkiama, Syed Zainudeen Mohd Said, Muhammad Saidu), Malaysia, 2016.](https://thesai.org/Downloads/Volume7No4/Paper_19-A_Subset_Feature_Elimination_Mechanism_for_Intrusion_Detection_System.pdf)

CẢM ƠN THẦY ĐÃ ĐỌC BÀI BÁO CÁO CỦA NHÓM CHÚNG EM!