Project Report: Team 3

Đỗ Nguyễn Anh Khoa[1, 3], Nguyễn Phúc Khang[1, 3], Nguyễn Minh Tú[2, 3], Bùi Ngọc Tuyết Nhi[2, 3]
[1]Khoa Khoa học và Kỹ thuật thông tin, Trường Đại học Công nghệ Thông tin
[2]Đại học Quốc gia Thành phố Hồ Chí Minh
{21522219, 21522194, 21522742, 20521712}@gm.uit.edu.vn

Abstract

0.1 Bối cảnh

- Theo Check Point Rearch (CPR), số lượng các cuộc tấn công mạng toàn cầu trong 2022 đã tăng 38% so với năm 2021.

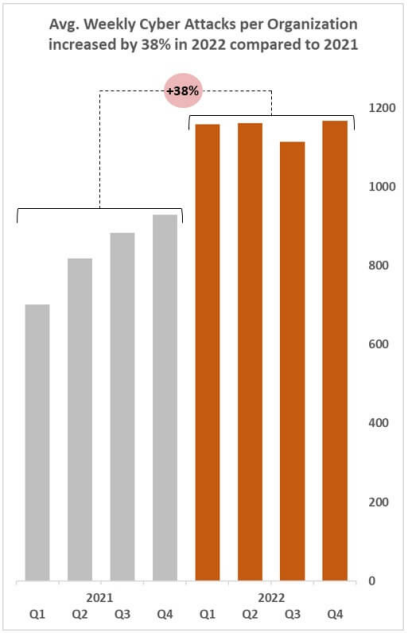

Figure 1: Số lượng các vụ tấn công theo quý của năm 2021 so với 2022

- Lĩnh vực giáo dục và nghiên cứu là lĩnh vực ghi nhận số lượng tấn công nhiều nhất, số lượng các cuộc tấn công tăng 43% so với năm 2021. Trung bình ghi nhận 2.314 cuộc tấn công mỗi tuần. Vì sự chuyển dịch của mô hình học tập trực tuyến ngày càng phát triển, nhiều cơ sở giáo dục chưa chú trọng nhiều vào vấn đề bảo mật dẫn đến tạo nên mãnh đất màu mỡ cho các hacker xâm nhập vào mạng thông qua nhiều phương tiện khác nhau. Việc mọi người học tập từ khắp nơi, thông qua các thiết bị riêng và Wifi công cộng đã tạo cơ hội cho các hacker thực hiện các mưa đồ xấu.(Team, 2023)

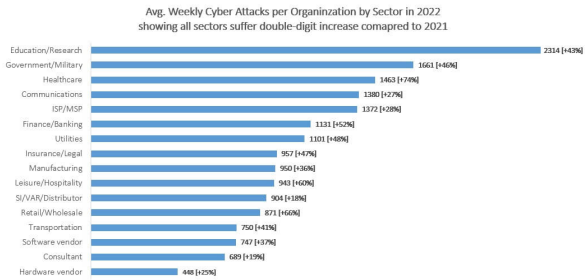

Figure 2: Số các vụ tấn công ghi nhận trong các lĩnh vực trong năm 2022

- Cùng với sự bùng nổ của trào lưu AI trong năm 2023, đặt biệt là ChatGPT, các thông tin cá nhân hay doanh nghiệp nếu không được che dấu và bảo mật tốt sẽ dễ dàng bị đánh cắp và khai thác phục vụ cho các mục đích xấu.
- Đáng nguy ngại hơn, Việt Nam cũng là nước nằm trong 10 nước có số lượng tấn công mạng nhiều nhất thế giới theo CyberProof.(DavidPur, 2022)

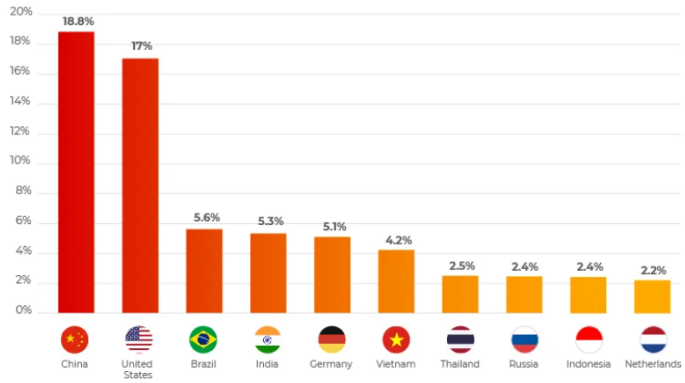

Figure 3: Top 10 quốc gia có số lượng tấn công mạng lớn nhất

0.2 Phương hướng giải quyết

- Tuy nhiên với những công nghệ như Machine Learning (Máy học) hoặc Deep Learning (Học sâu) cũng có thể được dùng để tổng hợp và phân tích những dữ liệu để có thể dự đoán được các mối đe dọa. Đó cũng là phương hướng mà nhóm chúng em đã và đang thực hiện, nhằm mang lại sự an toàn và tránh các cuộc tấn công mạng không mong muốn.

- Trong bài báo cáo này, chúng em đã sử dụng ngôn ngữ lập trình python và các thư viện hỗ trợ như Numpy, Pandas, Matplotlib... để làm sạch dữ liệu, trực quan hóa dư liệu, rút trích các đặc trưng cần thiết, chúng em đã thử nghiệm huấn luyện qua nhiều thuật toán Machine Learning (Logistic Regression, Decision Tree, XgBoost, RandomForestClassifier) và Deep Learning (Neural Network) nhằm tìm ra thuật toán tối ưu cho vấn đề. Chúng em đã huấn luyện qua nhiều bộ dữ liệu khác như NSL-KDD, KDD99... nhằm có được hiểu quả mô hình cao nhất.

1 Phân tích và khám phá dữ liệu

- Bộ dữ liệu NSL-KDD là một bộ dữ liệu được Tavallaee công bố vào năm 2009, là một phiên bản được rút gọn từ bộ dữ liệu KDD Cup năm 1999 với việc loại bỏ đi một số bản ghi bị thừa, các cột không cần thiết và một số thông tin bị trùng lắp(unb). Bộ dữ liệu được sử dụng rất nhiều trong các nghiên cứu khoa học và huấn luyện mô hình.

- Bộ dữ liệu gồm 3 tập CSV: Train, Dev và Test
  - Tập Train: 125973 Samples và 41 Features
  - Tập Dev: 4509 Samples và 41 Features
  - Tập Test: 18035 Samples và 41 Features

- Dưới đây là tên các thuộc tính có trong dataset.

duration	num_failed_logins
is_host_login	srv_diff_host_rate
protocol_type	logged_in
is_guest_login	dst_host_count
service	num_compromised
count	dst_host_srv_count
flag	root$_shell$
srv_count	dst_host_same_srv_rate
src_bytes	su_attempted
serror_rate	dst_host_diff_srv_rate
dst_bytes	num_root
srv_serror_rate	dst_host_same_src_port_rate
land	num_file_creations
rerror_rate	dst_host_srv_diff_host_rate
wrong_fragment	num_shells
srv_rerror_rate	dst_host_serror_rate
urgent	num_access_files
same_srv_rate	dst_host_srv_serror_rate
hot	num_outbound_cmds
diff_srv_rate	dst_host_rerror_rate
dst_host_srv_rerror_rate	

- Phân loại các thuộc tính

Kiểu	Thuộc tính
Catagory	Protocol_type(2), service(3), flag(4)
Numeric	Duration(1), src_bytes(5),dst_bytes(6),Land(7), logged_in(12),root_shell(14), su_attempted(15),is_host_login(21), is_guest_login(22), wrong_fragment(8),urgent(9), hot(10),num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18),num_access_files(19), num_outbound_cmds(20), count(23),srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29), diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

- Tập dữ liệu gồm 23 nhãn kiểu tấn công và có thêm 37 nhãn trong tập test, được rút gọn lại thành 2 nhãn:

  1. **Normal**: là những tập dữ liệu bình thường, không chứa mã độc
  2. **Attack**: là những mà độc mang mục đích tấn công mạng

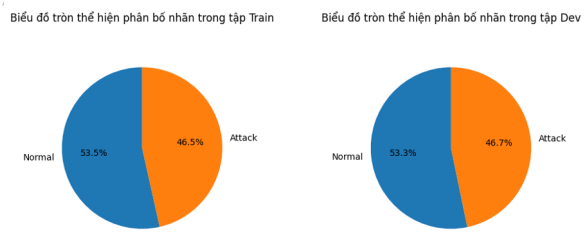

Figure 4: Biểu đồ trong thể hiện các nhãn trong tập train và dev

2 Data Preprocessing

- Dữ liệu NSL-KDD là dữ liệu đã được qua sử lý và đã rút gọn từ bộ dữ liệu KDD99(Tavallaee et al., 2009), nên bộ dữ liệu khá sạch và không có giá trị null. Những bên cạnh đó dữ liệu cũng còn những thuộc tính không có tác dụng và đóng góp trong việc học máy.

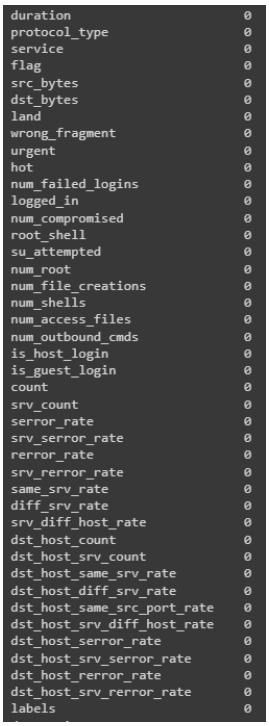

Figure 5: Thống kê các giá trị null trên các thuộc tính

- Biến đổi các kiểu dữ liệu chữ thành số để dễ dàng trong việc huấn mô hình. Sử dụng kĩ thuật Standard Scaler để cân bằng những thuộc tính số.

- Trong bộ data có những thuộc tính có giá trị là chữ, nhóm chúng em đã sử dụng kĩ thuật OneHotEncoder để sử lý các giá trị. Đối với những kiểu dữ liệu số, chúng em sử dụng Standard Scaler để cân bằng các giá trị số trong thuộc tính.

3 Feature & Model Selection

3.1 Đánh giá độ quan trọng của thuộc tính

- Thông qua hàm feature_importances có trong thư viện máy học và correlation giữa feature với labels chúng em đã rút ra được 10 thuộc tính quan trọng như sau :

  1. **Src_bytes**: Số bytes được gửi từ máy chủ nguồn sang máy chủ đích
  2. **Dst_bytes**: Ổ byte được gửi từ máy chủ đích sang máy chủ nguồn
  3. **Protocol_type**: Loại giao thức được sử dụng trong một kết nối. VD: TCP, UDP...
  4. **Flag**: Các cờ được sử dụng trong một gói tin VD: SYN,ACK,FIN...
  5. **Dst_host_srv_rerror_rate**: Tỷ lệ các kết nối đến cùng một dịch vụ trên cùng một máy chủ đích bị từ chối.
  6. **Hot**: Chỉ ra liệu kết nối có được sử dụng nhiều hay không.
  7. **Logged_in**: Chỉ ra liệu người dùng đã đăng nhập thành công hay không.
  8. **Dst_host_same_rate**: Tỷ lệ các kết nối đến cùng một dịch vụ trên cùng một máy chủ đích.
  9. **Dst_host_serror_rate**: Tỷ lệ các kết nối đến cùng một máy chủ đích bị lỗi
  10. **Same_srv_rate**: Tỷ lệ các kết nối đến cùng một dịch vụ

- Dưới đây là các bảng Feature Importances của từng thuật toán

	Thuộc tính	Random Forest
1	Src_bytes	15.68%
2	Dst_bytes	15.67%
3	Flat	8.59%
4	Dst_host_same_srv_rate	6.34%
5	Diff_srv_rate	6.23%
6	Logged_in	4.67%
7	Same_srv_rate	4.39%
8	Protocol_type	3.82%
9	Dst_host_diff_srv_rate	3.43%
10	Dst_host_srv_count	3.42%

	Thuộc tính	XGBoost
1	Src_bytes	36.08%
2	Protocol_type	14.88%
3	Dst_host_srv_rerror_rate	11.10%
4	Hot	7.54%
5	Count	5.07%
6	Dst_host_srv_count	4.41%
7	Dst_host_same_src_port_rate	3.80%
8	Logged_in	3.0%
9	Dst_bytes	2.18%
10	Dst_host_same_srv_rate	1.35%

- Sự tương quan giữa các feature so với labels

```
dst_host_srv_serror_rate       0.655609
dst_host_serror_rate           0.652828
serror_rate                    0.651523
srv_serror_rate                0.648992
count                          0.577001
flag                           0.498350
dst_host_count                 0.377096
service                        0.335215
srv_rerror_rate                0.253502
dst_host_srv_rerror_rate       0.253452
rerror_rate                    0.253082
dst_host_rerror_rate           0.252104
dst_host_diff_srv_rate         0.244205
diff_srv_rate                  0.204454
wrong_fragment                 0.095435
dst_host_same_src_port_rate    0.092089
dst_host_srv_diff_host_rate    0.062911
duration                       0.049528
land                           0.008052
src_bytes                      0.006064
dst_bytes                      0.004644
srv_count                      0.000088
urgent                        -0.003108
num_failed_logins             -0.006323
num_shells                    -0.008351
num_compromised               -0.010189
hot                           -0.013193
root_shell                    -0.019990
num_file_creations            -0.021369
su_attempted                  -0.022839
num_access_files              -0.036074
is_guest_login                -0.039524
srv_diff_host_rate            -0.118613
protocol_type                 -0.281233
logged_in                     -0.689973
dst_host_same_srv_rate        -0.695119
dst_host_srv_count            -0.722660
same_srv_rate                 -0.752783
dtype: float64
```

Figure 6: Sự tương quan giữa các feature

3.2 Phân tích các thuộc tính

- Để hiểu rõ hơn về 10 thuộc tính quan trọng này và tại sao chúng lại ảnh hưởng lớn tới việc ra quyết định của model. Nhóm chúng em đã tiến hành trực quan các thuộc tính này
- Ở thuộc tính flag ta thấy rằng những sample có label là normal tập trung chủ yếu ở ở cờ SF, và ngược lại labels attack tập trung chủ yếu ở cờ SO,REJ,RSTR.

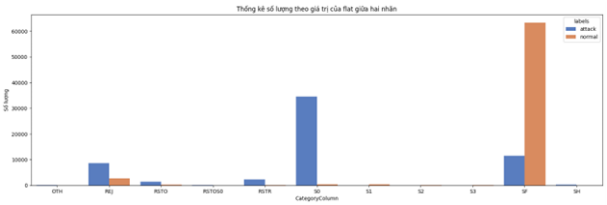

Figure 7: Phân bố của nhãn trên từng giá trị của thuộc tính flag

- Đối với thuộc tính Logged_in attack chủ yếu vào giá trị 0, và ngược normal sẽ tập trung vào giá trị 1.

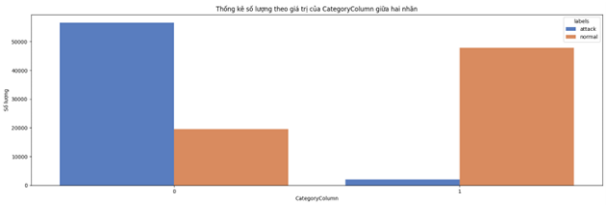

Figure 8: Phân bố của nhãn trên từng giá trị của thuộc tính Logged_in

- Protocol type: Giá trị của thuộc tính Protocol type tập trung chủ yếu vào nhãn 1 và hơn hẳn so với 2 nhãn còn lại (0 và 2). Và nhóm em nhận ra rằng số lượng từng nhãn trong từng giá trị của Protocol type phân bố đều và không có sự chênh lệch quá lớn.

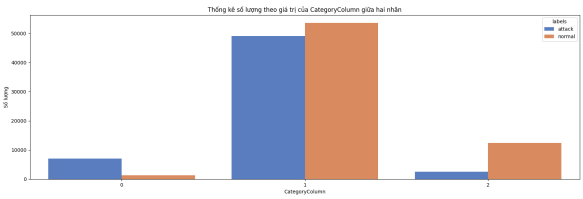

Figure 9: Phân bố của nhãn trên từng giá trị của thuộc tính protocol type

- Thông qua trực quan của thuộc tính dst_host_srv_rerror_rate, ta thấy rằng những chấm màu đỏ tượng trương cho sample có nhãn là attack chủ yếu tập trung ở 3 cụm chính. Chấm xanh tượng trưng cho normal thì rãi rác ở xung quanh khác.

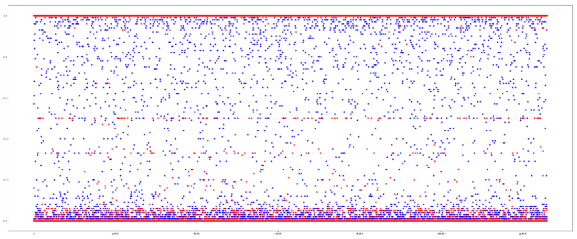

Figure 10: Phân bố của nhãn trên từng giá trị của thuộc tính dst_host_srv_reror_rate

- Tương tự với thuộc tính dst_host_same_rate, ta thấy rằng nhãn normal tập trung chính từ 0.0 tới 0.1 và 1.0

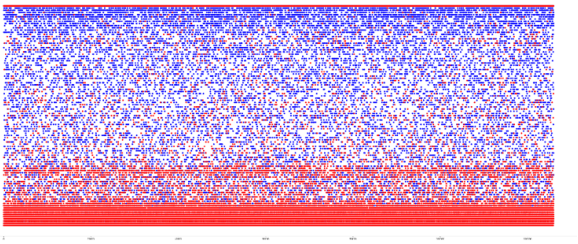

Figure 11: Phân bố của nhãn trên từng giá trị của thuộc tính Dst_host_same_rate

- Đối với host, chúng ta cũng có thể dễ dàng nhập ra những nhãn attack tập trung chính ở khoảng từ 0.0 tới 0.3

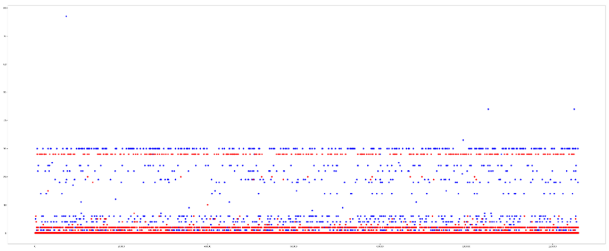

Figure 12: Phân bố của nhãn trên từng giá trị của thuộc tính Host

- Src byte là số byte được gửi từ máy chủ nguồn sang máy chủ đích, những sample có giá trị attack có giá trị src byte trung bình cao hơn sample normal rõ rệt

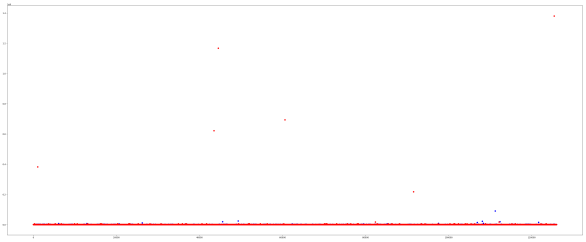

Figure 13: Phân bố của nhãn trên từng giá trị của thuộc tính Src bytes

- Tương tự src byte, các giá trị của dst byte cũng tương tự.

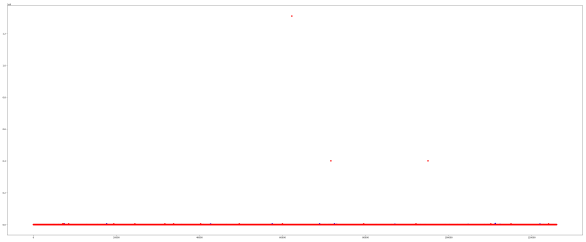

Figure 14: Phân bố của nhãn trên từng giá trị của thuộc tính Dst bytes

- Dst_host_serror_rate: Các giá trị trong thuộc tính Dst_host_serror_rate có sự phân chia rõ ràng, trong đó các nhãn Normal tập trung nhiều từ trong khoảng [0.2, 0.8], còn Attack tập trung nhiều vào các giá trị < 0.2 và > 0.8.

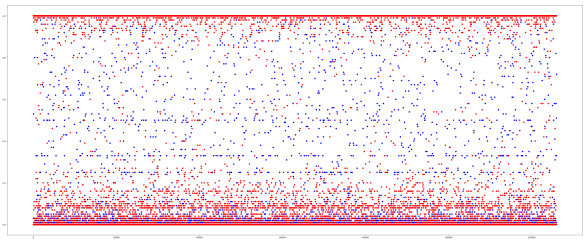

Figure 15: Phân bố của nhãn trên từng giá trị của thuộc tính Dst host serror bytes

- Same srv rate : Các giá trị trong thuộc tính Same srv rate tập trung chủ yếu vào khoảng

<= 0.2 . Do đó nếu 1 giá trị nếu <= 0.2 khả năng cao sẽ mang nhãn Attack.

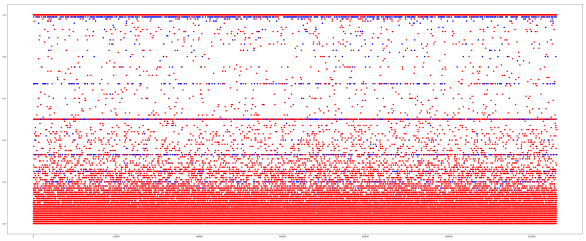

Figure 16: Phân bố của nhãn trên từng giá trị của thuộc tính Same srv rate

3.3 Các thuật toán

- Bài toán chúng ta cần giải quyết là phân lớp, dựa vào những kiến thức đã học trên lớp chúng em đã quyết định sử dụng 5 thuật toán sau:
  1. Logistic Regression
  2. Decision Tree
  3. Random Forest
  4. XGBoost
  5. Neuron Network
- Nhóm chúng em chọn 5 mô hình này vì đây là những mô hình có kết quả tốt khi được huấn luyện trên các bài toán phân lớp và dạng bảng. Nhóm chúng em cũng đã chạy mô hình này lúc chưa chọn lọc feature và tinh chỉnh tham số và đã đạt được kết quả khả quan

3.3.1 Logistic Regression

- Mô hình sau khi được chỉnh tham số với GridSearchCV

```
model_lr = LogisticRegression()

param_grid = {
    'penalty': ['l1', 'l2'],
    'C': np.logspace(-4, 4, 10),
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 200, 300],
    'fit_intercept': [True, False]
}

model_lr = GridSearchCV(model_lr, param_grid, cv=5, scoring='accuracy')
model_lr.fit(X_train, y_train)
print("Best parameters found: ", model_lr.best_params_)
```

Figure 17: Tham số đầu vào của GridSearchCV của Logistic Regression

- Tham số tối ưu đối với mô hình Logistic Regression

Best parameters found:{'C': 0.046415888336127774, 'fit_intercept': True, 'max_iter': 300, 'penalty': 'l1', 'solver': 'liblinear'}

Figure 18: Tham số tối ưu của mô hình Logistic Regression

3.3.2 Decision Tree

- Mô hình Decision Tree sau khi được tinh chỉnh tham số với GridSearchCV

```
model = DecisionTreeClassifier()

param_grid = {
    'max_depth': [None, 10, 20, 100, 1000],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_features': ['int', 'float', 'auto', 'sqrt', 'log2'],
}

model = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
model.fit(X_train, y_train)
print("Best parameters found: ", model.best_params_)
```

Figure 19: Tham số đầu vào của GridSearchCV của Decision Tree

- Tham số tối ưu đối với mô hình Decision Tree

Figure 20: Tham số tối ưu của mô hình Decision Tree

Best parameters found: {'criterion': 'entropy', 'max_depth': 1000, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2}

3.3.3 Random Forest

- Mô hình Random Forest sau khi được tinh chỉnh tham số với GridSeidSearchCV

```
model = RandomForestClassifier()

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
model = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
model.fit(X_train, y_train)
print("Best parameters found: ", model.best_params_)
```

Figure 21: Tham số đầu vào của GridSearchCV của Random Forest

- Tham số tối ưu đối với mô hình Random Forest

Figure 22: Tham số tối ưu của mô hình Random Forest

Best parameters found: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}

3.3.4 XGBoost

- Mô hình XGBoost sau khi được tinh chỉnh tham số với GridSearchCV

```
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
    'gamma': [0, 1, 5]
}

model_xgb = XGBClassifier()

model_xgb = GridSearchCV(model_xgb, param_grid, cv=5, scoring='accuracy')
model_xgb.fit(X_train, y_train)
```

Figure 23: Tham số đầu vào của GridSearchCV của XGBoost

- Tham số tối ưu đối với mô hình XGBoost

```
Best parameters found:  {'colsample_bytree': 1.0, 'gamma': 0, 'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 300, 'subsample': 0.8}
```

Figure 24: Tham số tối ưu đối với mô hình XGBoost

Best parameters found: {'colsample_bytree': 0.8, 'gamma': 0, 'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 300, 'subsample': 1.0}

3.3.5 Neuron Network

- Mô hình thuật toán Neuron Network

```
# Tạo một mô hình Sequential
model = Sequential()

# Thêm các layer vào mô hình
model.add(Dense(units=123, activation='relu', input_shape=(10, )))  # Layer input
model.add(Dense(units=128, activation='relu'))  # Hidden layer
model.add(Dropout(0.01))
model.add(Dense(units=64, activation='relu'))  # Hidden layer
model.add(Dropout(0.01))
model.add(Dense(units=32, activation='relu'))  # Hidden layer
model.add(Dropout(0.01))
model.add(Dense(units=16, activation='relu'))  # Hidden layer
model.add(Dropout(0.01))
model.add(Dense(units=8, activation='relu'))  # Hidden layer
model.add(Dropout(0.01))
model.add(Dense(units=4, activation='relu'))  # Hidden layer
model.add(Dropout(0.01))
model.add(Dense(units=2, activation='softmax'))
```

Figure 25: Mô hình thêm các layer và mô hình của neuron network

4 Result & Discussion

- Để đánh giá được sự hiểu quả của các mô hình, nhóm chúng em đã chọn 2 độ đo chính là Accuracy và F1 score để đánh giá. Để có thể thấy được sự hiệu quả của xử lý dữ liệu và chọn feature, nhóm chúng em so sánh kết quả trước khi tinh chỉnh và sau khi tinh chỉnh so sánh với nhau và đã đạt được kết quả khá tốt

Đánh giá / Mô hình	Trước khi tinh chỉnh		Sau khi tinh chỉnh	
	Accuracy	F1 Score	Accuracy	F1 Score
Logistic Regression	82.26%	82.17%	90.08%	90.04%
Decision Tree	92.89%	92.87%	94.02%	94.01%
Random Forest	92.65%	92.61%	93.22%	93.2%
XGBoost	93.47%	93.45%	93.34%	93.32%
Neuron Network	92.01%	91.97%	93.57%	93.56%

5 Conclusion

Trong báo cáo này, chúng em đã trình bày và thực hiện huyến luyện mô hình phát hiện xâm nhập bằng máy học và học xâu. Để triển khai và đo lường hiệu suất hệ thống, chúng em đã sử dụng bộ dữ liệu NLS-KDD và đã đạt được tỷ lệ phát hiện khá cao trong các mô hình đã huấn luyện. Trong 5 thuật toán chúng em triển khai, neuron network là mô hình phù hợp nhất và cho ra kết quả tốt nhất so với 4 thuật toán còn lại, tuy nhiên neuron network cũng có tương đối nhiều hạn chế trong việc dự đoán và triển khai. Chúng em hi vọng rằng bài báo cáo này có thể phần nào đó góp phần bảo đảm hệ thống an ninh mạng ở Việt Nam và thế giới. Trong tương lai, chúng em sẽ tiếp tục cải tiến và phát triển mô hình ngày càng tối ưu hóa hơn, bên cạnh đó chúng em sẽ cải thiện mô hình làm sao hệ thống có thể tự lấy dữ liệu trực tiếp về và tự học nhằm giúp hệ thống có thể nhanh chóng cập nhập và cải tiến để có khả năng phát hiện nhiều cuộc tấn công mạng đang ngày càng nhiều và khó để phát hiện hơn.

References

Canadian institute for cybersecurity datasets. https://www.unb.ca/cic/datasets/index.html. Accessed: January 4, 2024.

Niv DavidPur. 2022. Which countries are most dangerous? cyber attack origin – by country.

Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6.

Check Point Research Team. 2023.