Isolation Forestを用いた IoT向け異常検知手法に関する考察

菅田 大輔[1,a)]　石井 将大[2,b)]　松浦 知史[2,c)]

概要：本論文では, Isolation Forest を用いた IoT 向け異常検知手法の改善を行った. この研究の背景には, IoT（Internet of Things）の普及がある. 2030 年には約 300 億台ものデバイスが利用されると予測されており, すべての IoT 機器のセキュリティ確保が重要な課題となっている. 特に, 2016 年の Mirai 型マルウェアによる DDoS 攻撃は, その必要性を浮き彫りにした. 本研究では, 専門的なセキュリティ対策が難しい家庭内ネットワークなどの小規模環境に着目し, 軽量で高速な異常検知システムの提案を目指した. AbuAlghanam らが提案した Isolation Forest を用いた IoT 向け IDS は, その軽量性と高速性から今回の想定に適していると考えた. しかし, 彼らの手法には二つの問題が存在している. 第一に, 閾値を手動で設定する必要がある点, 第二に, 閾値による異常判定アルゴリズムの精度に限界がある点である. これらの課題を解決するため, 本研究では異常判定にロジスティック回帰を応用した判定アルゴリズムを提案する. 提案手法を二つのデータセットで実験した結果, 精度がそれぞれ 82.1%から 90.7%, 85.7%から 94.9%に改善したことを確認した.

キーワード：Isolation Forest, IoT, IDS, 異常検知

A Study on Anomaly Detection Method for IoT using Isolation Forest

Daisuke Sugata[1,a)]　Masahiro Ishii[2,b)]　Satoshi Matsuura[2,c)]

Abstract: This paper presents an enhancement of the anomaly detection method for IoT using Isolation Forest. The background of this research is the widespread adoption of the Internet of Things (IoT). By 2030, it is predicted that approximately 30 billion devices will be in use, making the security of all IoT devices a critical issue. The necessity of such security measures was particularly highlighted by the DDoS attacks carried out by the Mirai malware in 2016. This study aims to propose a lightweight and fast anomaly detection system, particularly suited for small-scale environments such as home networks, where implementing specialized security measures is challenging.The IoT-oriented IDS using Isolation Forest proposed by AbuAlghanam et al. was considered suitable for our scenario due to its lightweight and fast nature. However, their method has two major issues: the need to manually set thresholds and the limited accuracy of the anomaly detection algorithm based on these thresholds. To address these challenges, this study proposes a detection algorithm that applies logistic regression to anomaly detection. Experimental results on two datasets showed that the accuracy improved from 82.1% to 90.7% and from 85.7% to 94.9%, respectively.

Keywords: Isolation Forest, IoT, IDS, Anomaly Detection

[1] 東京工業大学 情報理工学院 数理・計算科学系
Department of Mathematical and Computing Sciences, School of Computing, Tokyo Institute of Technology
[2] 東京工業大学 学術国際情報センター
Global Scientific Information and Computing Center
[a)] sugata.d.aa@m.titech.ac.jp
[b)] mishii@gsic.titech.ac.jp
[c)] matsuura@gsic.titech.ac.jp

1. はじめに

Isolation Forest（以下, IF）は, その計算効率の良さから異常検知に広く用いられている. 例えば, AbuAlghanam らは IoT 向けの IDS（Intrusion Detection System）として IF を用いた手法を提案している [1]. しかし, AbuAlghanam らの手法には, 手動で閾値を設定する必要がある点や, 判定の組み合わせ方法における精度の限界といった課題が存在していた. 本研究では, これらの問題に対処するために IF の異常判定にロジスティック回帰を応用し, AbuAlghanam らの研究手法の改善を行なった.

この研究の背景として, IoT（Internet of Things）の急速な普及が挙げられる. 実際に 2030 年には約 300 億台ものデバイスが利用されると予測されており [8], すべての IoT 機器のセキュリティ確保が重要な課題となっている. 特に 2016 年の Mirai 型マルウェアによる DDoS 攻撃はその必要性を浮き彫りにした [5]. 家庭用のルーターやデジタルビデオレコーダーなどの管理不十分なデバイスが Mirai 型マルウェアに感染し, 一斉に特定のサーバーにアクセスを行うよう強制され, GitHub, Twitter, Reddit, Netflix など, いくつかの有名ウェブサイトがアクセス不能になった.

公的機関や企業組織といった大規模な環境では, セキュリティの専門家が構築し運用するフレームワークによって IoT 機器の安全性が確保されている. しかし, 先述した通り IoT 機器は多くの場所で使用されており, 家庭内のネットワークなどの小さな組織の場合となると, これらを管理・運用するための人員リソースを確保することや, 高価なセキュリティシステムを導入することは困難である. そのため, 安価な計算機環境でも高速に動作するようなセキュリティシステムが求められている.

こうした背景から, 小規模な環境に適した IDS (Intrusion Detection System) の設計について考察する. 2 章で IF のアルゴリズムの解説と IDS の概要を説明する. 3 章で従来の IF を用いた異常検知手法の問題点を整理し, これらの問題を解決する提案手法について述べる. 4 章では, UNSW-NB15 と NSL-KDD の 2 つのデータセットを用いて, 提案手法の有効性を評価した.

2. 研究方法

2.1 Isolation Forest の説明

Isolation Forest（以下, IF）は, Liu らが提案した [2] 外れ値を効率的に検出するためのアルゴリズムである. この手法は異常データの数が少なく, 他の正常データから離れて存在するという特性に基づいて設計されている. データをランダムに分割していく過程で異常データは正常データに比べて相対的に早く孤立するため, データが孤立するまでに必要な分割回数に着目することで, 異常データを検出できる. IF の実行は以下のステップに分かれる.

2.1.1 データの分割

IF は, ランダムに選んだ特徴量を基にデータを分割する. 具体的には, 選んだ特徴量のランダムな値を閾値として使用し, その閾値より大きいグループと小さいグループにデータを二分割する. この分割を全てのデータが孤立するまで繰り返し行い, 一つの決定木 ($iTree$) を作成する. このプロセスが終了したら, 複数の決定木を構築する.

2.1.2 異常スコアの算出

異常スコアは, データ点が生成された $iTree$ の枝をたどって到達するまでのパス長に基づいて計算される. パス長が短いほど, そのデータ点は異常である可能性が高い. 異常スコア $V(x,n)$ の計算式は式 1 の通りである.

$$V(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{1}$$

ここで $E(h(x))$ はデータ点 x の平均パス長 $c(n)$ はデータセットのサイズ n に依存する調整用の定数である.

2.1.3 異常判定

計算された異常スコアを使用して, 異常検知を行う. scikit-learn のライブラリを用いた実装では, パラメータ contamination で閾値を設定する. contamination は float 型で $(0, 0.5]$ の間の値をとり, データセット内で異常と見なすデータの割合を指定する. このパラメータに基づいて, トレーニングデータ内で指定された割合に相当するスコアの値が閾値として設定される. この閾値を超えるスコアを持つデータは異常と判定される.

2.2 AbuAlghanam らの提案手法

本研究では, 小規模環境でも軽量に動作する,AbAlghanam らが提案した IF を用いた IDS の手法 [1] に注目し, 彼らの研究を大いに参考にし IDS の設計を行った.

2.3 全体の設計

AbuAlghanam らが提案した IDS の設計 [1] は, 図 1 に示すように, データの前処理, 特徴量選択, 異常判定の 3 つのステップで構成される.

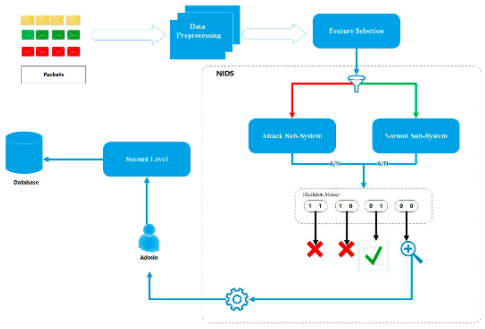

図 1 AbuAlghanam らの IDS の全体の設計 (出典:[1])

2.3.1　データの前処理

はじめにタイムスタンプや IP アドレスといった, IF で使用しない特徴量を削除する. 次に, IF が数値データのみを入力として受け付けるため, カテゴリカルデータを数値データに変換する. この変換には one-hot エンコーディングを利用し, 異なるカテゴリの特徴を個別の数値特徴として表現する. さらに, データセットは攻撃データと正常データに事前に分割され, それぞれ別のデータセットとして扱われる.

2.3.2　特徴量選択

AbuAlghanam らの研究では, 特徴量選択の手法として主成分分析（PCA）および特徴重要度（FI）法を使用した [1] との記述があったが, 具体的な手法については記載されていなかった. 本研究では特徴量の Random Forest アルゴリズムを用いて重要度を計算し, 設定された閾値を超える特徴量のみを選択した.

2.3.3　異常判定

AbuAlghanam らの手法では, データを攻撃通信と正常通信の 2 つに分割し, それぞれに対して IF のトレーニングを行う. この 2 つのトレーニングされた IF をサブシステムと呼ぶ. 2 つのサブシステムは各データに対して異常スコアを算出する. 異常スコアはどの程度データが異常であるかを定量的に表す指標である. その後, 設定した閾値を超える異常スコアを持つデータを Anomaly と判定する. AbuAlghanam らの手法では, 最終的な判定は 2 つのサブシステムの判定を表 1 のように組み合わせることで行われる. 本研究の IDS では, この判定の組み合わせ手法の代わりにロジスティック回帰を用いた判定アルゴリズムを採用した.

表 1　AbuAlghanam らの判定組み合わせ手法

正常サブシステム	攻撃サブシステム	判定結果
Normal	Anomaly	Normal
Anomaly	Normal	Anomaly
Normal	Normal	Anomaly
Anomaly	Anomaly	Unknown

3.　問題点の整理

まず, IF を異常検知手法として適用する際にどのような問題があるのかを明らかにするため, 事前実験を行った. はじめにデモデータを使用して IF の挙動を確認し, その考察をもとに特徴量選択手法の提案を行なった. また, AbuAlghanam らの提案した IF を用いた異常検知手法 [1] の問題点を整理した.

3.1　ノイズ特徴量が混入すると精度が悪化する問題

使用するデモデータは, (2.5, 2.5) と (-2.5, -2.5) を中心とした正常データ群と, 10 から-10 の範囲に一様に分布した異常データ群からなる. 二次元の場合のデモデータを図 2 に示す.

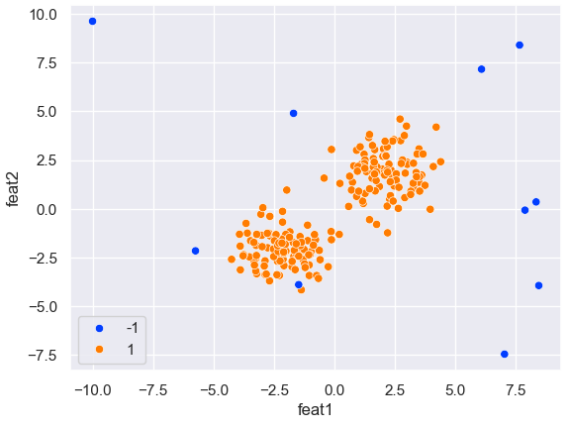

図 2　デモデータの 2 次元グラフ

3.1.1　特徴量数と精度の関係

はじめに, デモデータと特徴量数の関係を調査した. 図 3 に示すように, 特徴量数が増えるにつれて, 異常検知の精度が単調に向上することがわかった. また, 精度ののびは増加に反比例して緩やかになっていることもわかる. ゆえに, IF は目標とする精度に対して十分な特徴量数が存在すると言える.

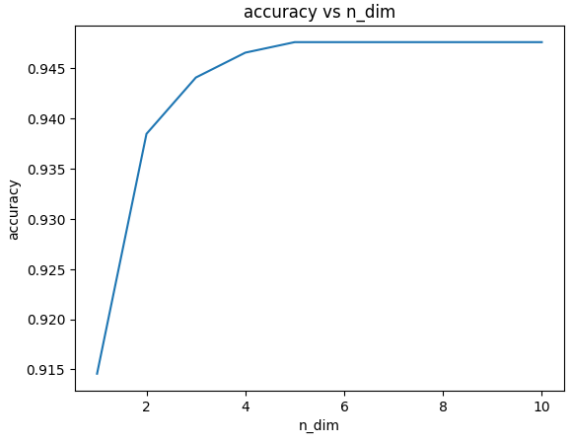

図 3　判定に有効な特徴量数と IF の精度の関係

3.1.2　ノイズ特徴量の影響

続いて, デモデータにノイズ特徴量を含めたときの精度の悪化について調査した. パケット通信を監視して得られたデータセットの全ての特徴が, 異常検知に有効なわけではない. IF は特徴量同士の重みづけを行わないため, これらの判定に有効でない特徴量が混ざると精度が低下すると考えられる. そこで, デモデータにノイズ特徴量を含めていき, 精度の変化を調査した. 結果は図 4 に示すように, ノイズとなる特徴量が混ざると精度が低下することがわかった. また, 今回の実験の場合だと, ノイズ特徴量が判定に有

効な特徴量数の 2 倍以上になると, 精度が急激に低下することがわかった.

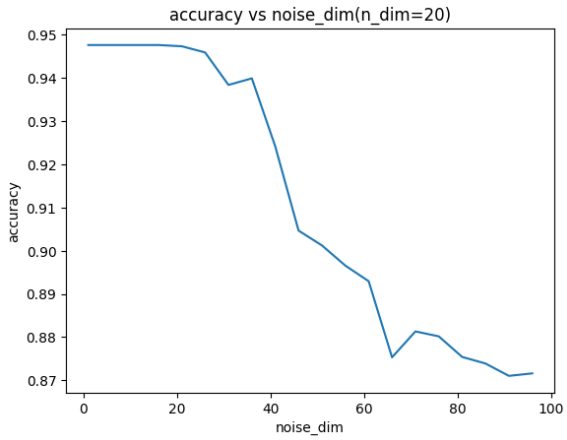

図 4 ノイズ特徴量の精度に対する影響

3.1.3 重要度に基づいたノイズ特徴量の除去

前節の実験から, データセットからノイズとなる特徴量を取り除くことが重要であると考えられる. ところで, IF はツリーベースの異常検知手法である. そこで, 同じくツリーベースの Randam Forest から特徴の重要度を参照すれば, ノイズとなる特徴量を取り除けるのではないかと考えた. 図 5 は, Random Forest で算出した特徴量の重要度を表している. このグラフは, ノイズ特徴量を判別できていることがわかる. そして, 実際にノイズ特徴量を取り除いた場合の精度を調査したところ, 精度は 0.872%から 0.945%まで向上した. この結果から, Feature Importance による特徴量選択手法は精度の向上に有効ではないかと考えた.

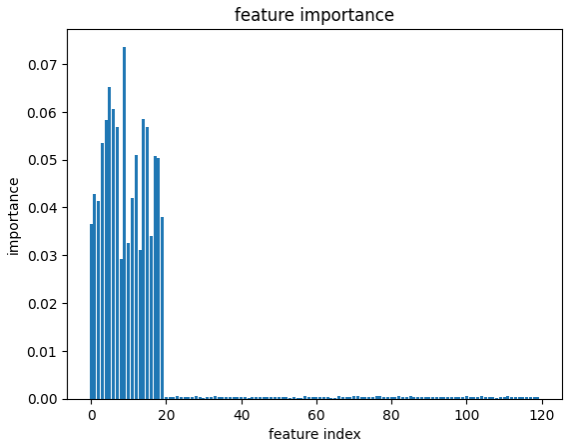

図 5 Random Forest で算出した特徴量の重要度

3.2 異常検知の境界線の問題

AbuAlghanam らが提案した IF を用いた異常検知手法 [1] では, 2 つのサブシステムが異常スコアを算出し, それが閾値を超えたデータを異常と判定する. そして, この 2 つの判定を表 1 のように組み合わせて最終的な判定を行っている. しかし, この手法は, 以下の 2 つの問題を抱えている.

- 各サブシステムの異常スコアの閾値を手動で設定する必要があること.
- 異常判定の境界線が垂直であるため, 異常スコアの分布によっては誤判定が多くなること.

実際に異常スコアの分布を調査したところ, 図 6 と図 7 で示されるように, 異常データと正常データの境界線は対角方向になっている.

図 6 UNSW-NB15 における訓練データの異常スコア分布

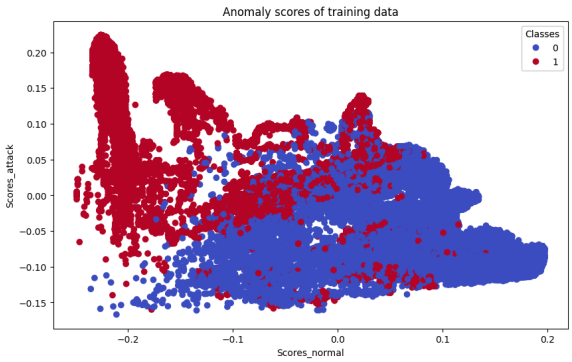

図 7 NSL-KDD における訓練データの異常スコア分布

このような場合, AbuAlghanam らの手法 [1] のように, 垂直に分割しては誤検知が多く発生してしまう問題がある. 図 8 と図 9 は, AbuAlghanam らの手法による異常判定結果を示しているが, 垂直に分割する手法では, 異常データが多く誤検知されていることがわかる.

この問題を解決するために, ロジスティック回帰を用いて異常判定の境界線を斜めに設定する手法を 4 章で提案する.

4. 提案手法

3 章で述べた, 以下の 2 つの問題

- ノイズ特徴量が精度に悪影響を及ぼす問題
- 異常検知の境界線の設定の問題

を解決するために, 以下の 2 つの提案を行った.

一つは, ノイズ特徴量を取り除くための特徴量選択手法の提案である. もう一つは, 異常検知の境界線が斜めに分

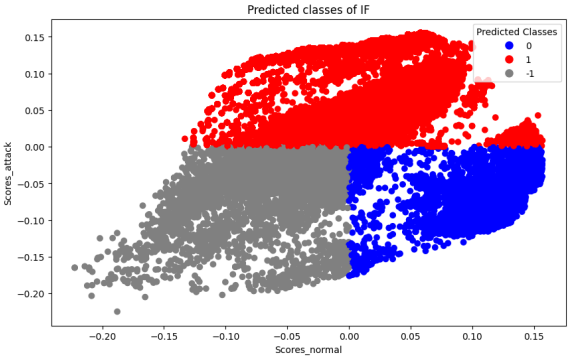

図 8 UNSW-NB15 における AbuAlghanam らの判定手法

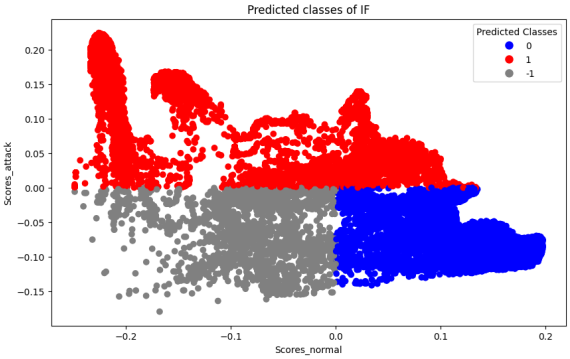

図 9 NSL-KDD における AbuAlghanam らの判定手法

布する場合にも適切な判定ができるよう, ロジスティック回帰を用いて境界を決定する手法の提案である.

4.1 特徴量選択手法の提案

本手法の疑似コードは表 2 の通りである. まず, Random Forest を用いて特徴量の重要度を算出する. 次に, 特徴量の重要度の上位 10%を選択し, 新しいデータセットを作成する. 実装には, scikit-learn のライブラリを用いた [4].

表 2 特徴量選択手法の擬似コード

```
# Load training data
    data, labels = load_data()
# Initialize Random Forest model
    model = RandomForest()
# Train the model
    model.fit(data, labels)
# Calculate feature importances
    feature_importances = model.feature_importances_
# Select top 10% features
    threshold = np.percentile(feature_importances, 90)
    selected_features = feature_importances > threshold
# Create a new dataset with selected features
    new_data = data[:, selected_features]
```

4.2 ロジスティック回帰を適用した異常判定手法

はじめに, 2 つのサブシステムで訓練データの異常スコアを算出する. 次に, これらの異常スコアと訓練データのラベルを入力としてロジスティック回帰をトレーニングする. そして, テストデータの異常スコアを入力としてロジスティック回帰を適用し, 異常判定を行う. 疑似コードは表 3 の通りである. ロジスティック回帰のパラメータは設定しておらず, scikit-learn のライブラリを用いて実装した.

表 3 ロジスティック回帰を適用した異常判定手法の擬似コード

```
# Calculate anomaly scores
    score1 = sub1.predict(train_data)
    score2 = sub2.predict(train_data)
# Prepare training data
    X_train = np.column_stack((score1, score2))
    y_train = labels
# Train logistic regression
    clf = LogisticRegression()
    clf.fit(X_train, y_train)
# Apply to test data
    test_score1 = sub1.predict(test_data)
    test_score2 = sub2.predict(test_data)
    X_test = np.column_stack((test_score1, test_score2))
    preds = clf.predict(X_test)
```

5. 結果と考察

5.1 比較するアルゴリズム

本研究では, 以下の 2 つの比較を行った. 特徴量選択手法の効果の比較を行うため, 特徴量選択を行わなかった場合と, Random Forest を用いた特徴量選択手法を用いた場合の結果を比較する. また, 判定の組み合わせアルゴリズムの比較には, AbuAlghanam らの 2 つのサブシステムの判定を組み合わせて検知を行う手法 [1] と, ロジスティック回帰を用いて判定を行う手法の結果を比較した.

(1) **特徴量エンジニアリングの比較**

- 特徴量選択なし
- Random Forest を用いた特徴量選択手法

(2) **判定の組み合わせアルゴリズムの比較**

- AbuAlghanam らの判定手法 [1]
- ロジスティック回帰を用いて判定

5.1.1 実験環境

実験は Mac Book Pro 2017 2.3GHz Intel Core i5, 8GB RAM で行った. また, 実験に用いたプログラムは Python3.10.4 で実装した. Isolation Forest や Random Forest の実装には, scikit-learn のライブラリを用いた [4].

5.1.2 データセット

本実験では, 以下の 2 つのデータセットを用いた. 表 4 にこれらのデータセットの概要を示す. 実験の際は, scikit-learn ライブラリの train-test-split 関数を用いて, これらのデータセットをトレーニングデータとテストデータに 7:3 に分割し評価を行った.

(1) **NSL-KDD**

異常検知や機械学習の分野で標準的なベンチマークデータセットとして広く利用されてきたKDDCUP99[6]の問題点を解決するために, Tavallaee らによって提案されたデータセットであり, データの冗長性や攻撃データの割合を調整したもの [7].

(2) **UNSW-NB15**

既存のデータセットの問題点を解決し, 現代のネットワークトラフィックを包括的に反映するために, Moustafa と Slay によって作成されたデータセット [3].

表 4 データセットの概要

No	NSL-KDD	UNSW-NB15
データ数	125,972	175,341
正常データ数	67,343	56,000
攻撃データ数	58,629	119,341
特徴量数	43	45
エンコード後の特徴量数	124	198

5.1.3 評価指標

本研究では, 以下の 2 つの評価指標を用いてモデルの性能を評価した.

(1) **Accuracy**

正確度を示し, 予測が実際のクラスと一致する割合を示す. Accuracy は以下の式で定義される.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

ここで, TP は True Positive, TN は True Negative, FP は False Positive, FN は False Negative を表す.

(2) **F1-score**

Precision と Recall の調和平均を示す. F1-score は以下の式で定義される.

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Precision と Recall はそれぞれ以下の式で定義される.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

5.2 結果

5.2.1 異常スコアの分布

はじめに, 2 つのデータセットに対してトレーニングデータの異常スコアの分布を調査した. 図 6 は UNSW-NB15 における異常スコアの分布を示し, 図 7 は NSL-KDD における異常スコアの分布を示す. 横軸は正常データで訓練された IF が算出した異常スコアを表し, 縦軸は攻撃データで訓練された IF が計算した異常スコアを表している. それぞれのゼロ点は異常スコアの閾値であり, 負の方向に大きくなるほど異常スコアが高く, 正の方向に大きくなるほど異常スコアが低いことを意味する. 各データ点の色はラベルを示しており, 赤が攻撃データ（1）, 青が正常データ（0）を表す.

図 6 をみると, UNSW-NB15 ではデータが対角線方向に分布しており, 2 つのサブシステムの判定が相反する場合が多いことがわかる. また, 右下に正常データが多く存在し, この領域の判定は適切に行われていることもわかる. 攻撃データと正常データの境界線が対角方向に位置しているため, 従来の垂直な分割に比べて, ロジスティック回帰を用いて斜めに分割することで精度が向上する可能性がある.

図 7 から, NSL-KDD では, 右下と左下にそれぞれ攻撃データと正常データが集中しており, 適切に行われている判定が多いことがわかる. 同様に, 2 つのデータの境界線が対角線方向に位置しているため, ロジスティック回帰を用いることで精度が向上する可能性がある.

表 5 UNSW-NB15 でのモデルの性能評価結果

Model	Accuracy	F1
特徴量選択なし, 閾値ベース	0.7506	0.7526
特徴量選択なし, 提案手法	0.8685	0.8666
特徴量選択あり, 閾値ベース	0.8216	0.8545
特徴量選択あり, 提案手法	0.9076	0.9047

表 6 NSLKDD でのモデルの性能評価結果

Model	Accuracy	F1
特徴量選択なし, 閾値ベース	0.7873	0.8171
特徴量選択なし, 提案手法	0.9370	0.9369
特徴量選択あり, 閾値ベース	0.8574	0.8898
特徴量選択あり, 提案手法	0.9491	0.9490

5.2.2 モデルの性能評価

UNSW-NB15 の結果は表 5,NSL-KDD の結果は表 6 の通りである. 表 5 を見ると, 特徴量の選択をすることで, 精度が 0.7506 から 0.8440 に向上したことがわかる. また, ロジスティック回帰を用いて判定を行うことで, 精度が 0.8440 から 0.9112 に向上したことがわかる. 同様に, 表 6 から, NSL-KDD でも, 特徴量の選択をすることで精度が 0.7873 から 0.8466 に向上し, ロジスティック回帰を用いて判定を行うことで精度が 0.8466 から 0.9452 に向上したことが確認できた.

5.3 考察

まず, ロジスティック回帰を導入することで, 異常スコアの分布が斜めに広がる場合により正確な境界を引くことが可能になった. 図 8 と図 9 はそれぞれ, 2 つのサブシステムの判定結果を組み合わせた従来の手法の判定結果を示しており, 図 10 と図 11 はロジスティック回帰を用いた提案手法の判定結果を示している. 図 8 と図 9 から, 2 つのサブシステムの判定を組み合わせた手法において, どれだけ閾値を適切に設定したとしても, 異常スコアの分布が斜めに

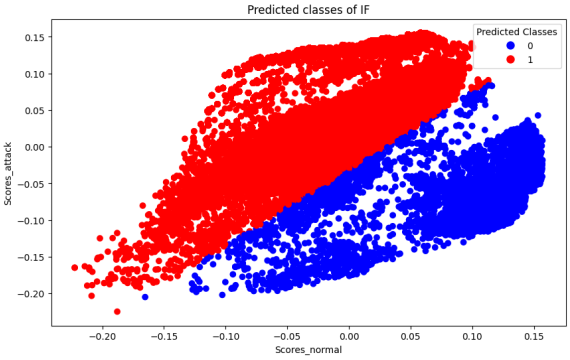

図 10　UNSW-NB15 における提案手法による結果

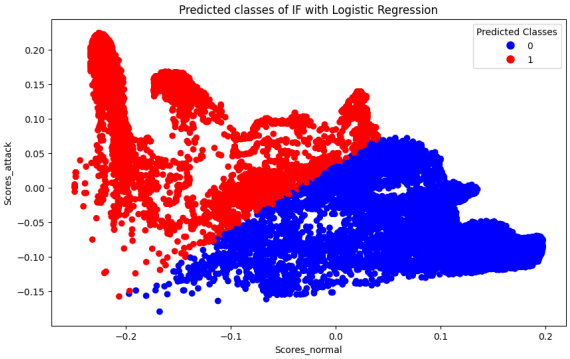

図 11　NSL-KDD における提案手法による結果

広がる場合には誤判定が多く発生することがわかる．一方，図 10 と図 11 からは，ロジスティック回帰を用いた手法が，斜めに広がる異常スコアの分布に対しても，より正確に判定できることが確認できる．従来の垂直な分割では誤判定が多発するケースにおいても，精度の向上が見込めることが実証された．

また，本アプローチが適用できる範囲についても考察が必要である．異常スコアの分布において，正常な通信と攻撃通信が直線的に分離できている場合，この手法は効果的である．しかし，攻撃通信と正常な通信が十分に分離されていない場合や，境界線が曲線状になっている場合には，判定精度が低くなる可能性がある．実際，図 6 を見ると，斜めに分布した攻撃通信の中に正常な通信が多く混ざっている．一方で図 7 では，概ね攻撃通信と正常な通信が分離され，お互いが混ざりあう部分が比較的小さい．これら 2 つのデータセットを比較すると，NSL-KDD は UNSW-NB15 よりも高い検知精度を示している．このように，異常スコアの分布によっては効果的でない場合があり，別の特徴量選択手法の採用や異なる機械学習アルゴリズムの適用など，さらなる工夫が必要となる可能性がある．

さらに，ロジスティック回帰を適用する際のコストや計算量に関するトレードオフも存在する．IF 単体での異常検知に比べ，ロジスティック回帰を用いることでモデル構築にかかる計算時間が増加することが懸念される．

以上より，提案手法は特定の条件下では高い効果を発揮することが確認されたが，全てのシナリオにおいて有効であるとは限らない．今後は，異常スコアの分布がさらに複雑な場合における改良や，計算効率の最適化について検討することが求められる．

6.　おわりに

本研究では，家庭環境でも高速に動作するような IDS の提案を目指すため，AbuAlghanam らが提案した IF を用いた IoT 環境向けの異常検知手法 [1] に注目し，その手法の改善を行なった．AbuAlghanam らの手法には，手動で閾値を設定する必要がある点や，判定の組み合わせ方法における精度の限界といった課題が存在していた．これらの課題を解決するためにロジスティック回帰を応用した判定アルゴリズムを導入した．この新しいアルゴリズムにより異常スコアの分布に応じた柔軟な境界設定が可能となり，AbuAlghanam らの判定組み合わせ手法に比べて，精度が大幅に向上したことが確認された．今後の課題として，より複雑な異常スコアの分布に対する適用性の検討や計算効率の最適化が求められる．

参考文献

[1] AbuAlghanam, O., Alazzam, H., Alhenawi, E., Qatawneh, M. and Adwan, O.: Fusion-based anomaly detection system using modified isolation forest for internet of things, *J. Ambient Intell. Humaniz. Comput.*, Vol. 14, No. 1, pp. 131–145 (2023).

[2] Liu, F. T., Ting, K. M. and Zhou, Z.-H.: Isolation Forest, *2008 Eighth IEEE International Conference on Data Mining*, IEEE, pp. 413–422 (2008).

[3] Moustafa, N. and Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), *2015 Military Communications and Information Systems Conference (MilCIS)*, IEEE, pp. 1–6 (2015).

[4] scikit-learn developers: scikit-learn: Machine Learning in Python, https://scikit-learn.org/stable/ (2024).

[5] Sinanović, H. and Mrdovic, S.: Analysis of Mirai malicious software, *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 1–5 (online), DOI: 10.23919/SOFTCOM.2017.8115504 (2017).

[6] Stolfo, S., Fan, W., Lee, W., Prodromidis, A. and Chan, P. K.: Cost-based modeling for fraud and intrusion detection: Results from the KDD99 cup, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, ACM Press, pp. 130–134 (1999).

[7] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A. A.: A detailed analysis of the KDD CUP 99 data set, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, pp. 1–6 (2009).

[8] Vailshery, L. S.: Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2023, with Forecasts from 2022 to 2030, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide (2022).