VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Product Manual of Intrusion detection System Using Machine Learning

By :- Aayush Verma(21BCE1226)

Sanket Agrawal(21BCE1195)

Srutayu Pandey(21BCE1168)

Introduction

- Overview of the Product

This product manual is designed to guide users through the entire process of setting up, configuring, using, and maintaining the Machine Learning-based Intrusion Detection System (IDS). This IDS employs cutting-edge machine learning techniques to analyze network traffic and identify potentially malicious activities or security threats in both real-time and offline modes. The system integrates multiple machine learning algorithms, including Random Forest, XGBoost, LightGBM, and CatBoost, to provide accurate and efficient intrusion detection. Each model is trained on the CICIDS2017 dataset, a comprehensive and widely recognized dataset for intrusion detection tasks.

The IDS is tailored for real-time monitoring of network traffic, enabling it to detect intrusions and anomalies as they occur, ensuring swift response and mitigation. In addition to real-time detection, the system supports batch processing of historical network logs, allowing users to analyze past network activities for potential threats that might have gone unnoticed during the real-time process. By leveraging multiple machine learning models, the system ensures high detection accuracy while minimizing false positives, thus reducing alert fatigue for security teams.

This product is designed to be highly customizable, providing users with the ability to fine-tune alert thresholds, retrain models with new datasets, and configure notifications for detected threats. Additionally, a web-based dashboard is provided for real-time monitoring and management of detected intrusions, making it easy for network administrators and security personnel to keep track of the system's performance and network health.

- Features of the System

The Intrusion Detection System comes equipped with several key features aimed at providing comprehensive network security. These features include:

1. Real-Time Monitoring: The system captures and analyzes network traffic in real-time, providing immediate detection of anomalies or potential threats.

2. Machine Learning Models: It leverages multiple pre-trained machine learning models (Random Forest, XGBoost, LightGBM, and CatBoost) to classify network traffic based on learned patterns from the CICIDS2017 dataset.

3. Batch Processing: In addition to real-time monitoring, the system allows for offline processing of historical network data to detect any anomalies that may have been missed during live monitoring.

4. Customizable Alert Thresholds: Administrators can set and adjust alert thresholds to balance detection accuracy and false positives based on the specific needs of the organization.

5. Dashboard Interface: A user-friendly web-based dashboard provides live visualization of network traffic, detected threats, and system performance metrics. It also offers options for managing system configurations and monitoring model performance.

6. Retraining and Model Updates: The system allows administrators to retrain the machine learning models using new datasets, enabling it to stay current with evolving network patterns and emerging threats.

7. Alerting and Notifications: The system generates detailed alerts for detected anomalies and supports integration with email or SMS notification systems to ensure real-time awareness of potential threats.

- Who Should Use This Manual

This manual is designed for use by a variety of stakeholders, primarily those involved in managing network security within an organization. These include:

1. System Administrators: Individuals responsible for maintaining the IT infrastructure and ensuring that network security systems are properly configured and operational.

2. Network Engineers: Professionals tasked with monitoring network traffic and ensuring that the system detects and prevents unauthorized access.

3.Cybersecurity Professionals: Security analysts and incident response teams who need to analyze detected intrusions, investigate security incidents, and mitigate threats.

4. Developers and Data Scientists: Those interested in customizing or extending the system, retraining models, or analyzing the performance of machine learning algorithms.

5. IT Managers: Decision-makers who oversee network security operations and are responsible for implementing effective intrusion detection solutions within their organizations.

System Overview

- What is the IDS?

An Intrusion Detection System (IDS) is a cybersecurity tool designed to monitor network or system activities for malicious behaviors or policy violations. The system analyzes network traffic and, using predefined rules or learned behavior patterns, flags activities that deviate from the expected norm. The IDS in this project is based on machine learning techniques, allowing it to identify both known and previously unseen threats through pattern recognition. By continuously learning from historical data and applying this knowledge to incoming network traffic, the IDS provides a dynamic, adaptable defense against intrusions, making it a critical component of modern network security infrastructure.

Traditional IDS systems rely heavily on signature-based detection, where predefined rules are used to flag malicious activity. While effective in some cases, this approach often struggles to detect new or evolving threats that don't match existing signatures. In contrast, a machine learning-based IDS learns from past data to detect anomalous patterns, even when specific signatures are not present. This makes it particularly effective in detecting zero-day attacks and advanced persistent threats (APTs).

- How the IDS Works

The IDS operates in several stages:

1. Data Collection: The IDS begins by collecting network traffic, which can be real-time data capturedfrom the live network or offline data from historical logs. This traffic includes packet-level details such as source and destination IP addresses, protocol types, and payload information.

2. Data Preprocessing: Once collected, the data undergoes preprocessing. This step involves cleaning the data to remove missing or erroneous values, normalizing the data to ensure consistency, and transforming it into a format suitable for model input. Preprocessing is crucial to ensure the accuracy and reliability of the machine learning models.

3. Feature Extraction: Relevant features are extracted from the preprocessed data, such as packet size, flow duration, and protocol types. These features serve as input for the machine learning models, which use them to identify patterns and detect anomalies.

4. Model Application: The system applies pre-trained machine learning models (Random Forest, XGBoost, LightGBM, CatBoost) to classify the network traffic as either benign or malicious. Each model operates on the extracted features, using its learned patterns to make predictions.

5. Real-Time Detection: In real-time mode, the system continuously monitors live traffic, applying the machine learning models to each packet or flow as it passes through the network. If a threat is detected, the system immediately generates an alert.

6. Batch Processing: In batch mode, the IDS processes previously collected network logs, allowing users to perform offline analysis and detect threats that may have been missed during real-time monitoring.

7. Alerting and Notification: When an anomaly or threat is detected, the system generates an alert. These alerts can be logged for further analysis or used to trigger notifications (email or SMS) to network administrators or security personnel.

8. Visualization and Reporting: The system includes a web-based dashboard that provides visualizations of detected intrusions, network traffic trends, and model performance. This allows administrators to quickly assess the system's health and take action if needed.

- System Architecture

The architecture of the IDS is designed to support both real-time and batch processing, ensuring flexibility in how it can be deployed and used. The system is composed of the following key components:

1. Data Collection Module: Captures network traffic from the live network or from offline logs. This module interfaces with network capture tools like Wireshark or tcpdump to collect packet-level data.

2. Data Preprocessing Module: Prepares the raw network traffic data for analysis by the machine learning models. This includes cleaning, normalizing, and transforming the data.

3. Feature Engineering: Extracts the necessary features from the network traffic, such as packet size, protocol type, flow duration, etc. These features are essential for accurate prediction by the machine learning models.

4. Machine Learning Models: The IDS uses four pre-trained machine learning models (Random Forest, XGBoost, LightGBM, CatBoost), each of which provides predictions based on the extracted features. The models are trained on the CICIDS2017 dataset, ensuring their ability to detect a wide range of network intrusions.

5. Real-Time Detection Engine: This engine continuously monitors incoming network traffic and applies the machine learning models to detect anomalies. It provides immediate feedback to administrators in the form of alerts.

6. Batch Processing Engine: Allows for offline analysis of historical network data, providing insights into network behavior over time.

7. Alerting System: Generates alerts whenever a potential intrusion is detected. These alerts can be configured to notify system administrators via email or SMS.

8. Web-Based Dashboard: The dashboard provides a user-friendly interface for monitoring the system, visualizing network traffic, and reviewing detected threats.

System Requirements

- Hardware Requirements

The IDS system requires certain hardware configurations to ensure smooth and efficient operation, especially when processing large volumes of network data in real-time. At a minimum, the system should be deployed on a machine with the following specifications:

1. Processor: A minimum of a quad-core processor (4-core CPU) is recommended to handle the computation required for real-time detection and machine learning inference. For larger network environments, an 8-core or higher CPU may be necessary to avoid performance bottlenecks.

2. Memory (RAM): At least 8GB of RAM is required for basic operation, especially when dealing with large datasets and real-time traffic. For environments with high traffic volumes, 16GB or more is recommended to ensure that data processing and model inference can occur without delays.

3. GPU: A GPU is not strictly necessary for small-scale training, but it can help accelerate the process.

- Recommended Hardware Configuration

- For optimal performance, especially when dealing with larger datasets or real-time monitoring, a system with an Intel Core i7 or AMD Ryzen 7 processor (6 or more cores), 16 GB or more of RAM, and an NVIDIA GPU with CUDA support (e.g., NVIDIA GTX 1660 or higher) is recommended.

- At least 20 GB of available storage space for data handling and model storage.

4. Storage: A minimum of 50GB of storage is needed for storing network logs, model files, and system logs. If the system will be processing large historical datasets, additional storage may be required.

5. Network Interface Card (NIC): A high-performance NIC is essential for capturing network traffic in real-time, especially in high-throughput environments. A Gigabit Ethernet NIC or higher is recommended.

- Software Requirements

The IDS system is built using Python and requires several software libraries and tools to function correctly. The following software components are necessary:

1. Python: The system requires Python 3.7 or higher for execution. Python is the core language used for building the IDS, and its libraries are essential for machine learning and data processing tasks.

2. Operating System: The IDS is compatible with major operating systems, including Linux (Ubuntu preferred), Windows, and macOS. For production environments, a Linux-based server (such as Ubuntu 20.04) is recommended for stability and performance.

3. Required Python Libraries: The following libraries are required for machine learning, data processing, and system functionality:

- numpy
- pandas
- scikit-learn
- xgboost
- lightgbm
- catboost
- matplotlib
- joblib
- river

These dependencies can be installed using the pip install -r requirements.txt command after setting up the environment.

- Networking Requirements

The system needs appropriate networking configurations to monitor live traffic and provide access to the web-based dashboard. Ensure the following:

1. Network Capture Permissions: The IDS needs to be deployed with permissions to capture network traffic, either through raw packet capture tools or by accessing network taps.

2. Dashboard Port Configuration: The web-based dashboard operates on port 5000 by default. Ensure that this port is open and accessible to allow administrators to view the dashboard.

3. Internet Access (Optional): For notifications (email/SMS), the system needs internet access to communicate with external mail servers or notification services.

Installation Guide

Step-by-Step Installation Instructions

1. Clone the Project Repository:

The first step is to download the project files from the repository. Use the following command to clone the repository:

```
git clone <repository_url>
cd <project_directory>
```

This command will copy all necessary files into your working directory and prepare you for the next steps.

2. Set Up the Python Environment:

Create and activate a Python virtual environment to keep dependencies isolated from your system. This is a recommended best practice for Python-based projects. Use the following commands to set up the environment:

```
python -m venv ids_env
source ids_env/bin/activateOn Windows use ids_env\Scripts\activate
```

The virtual environment ensures that all dependencies for the IDS system are managed within a controlled environment, preventing conflicts with other projects.

3. Install Dependencies:

The system requires several Python libraries for data processing and machine learning. Install these dependencies using the provided requirements.txt file:

```
pip install -r requirements.txt
```

This command will install all the required libraries, ensuring that the IDS can perform data preprocessing, feature extraction, and machine learning model inference.

4. Configuration

Before running the IDS, you need to configure it to suit your environment. The configuration file (config.yaml) contains essential parameters such as data paths, model save paths, and alert thresholds. Make sure to adjust these settings based on your requirements:

yaml

```
data_path: "data/CICIDS2017/"

model_save_path: "models/"

alert_threshold: 0.5
```

Adjust the alert threshold based on the desired sensitivity of the IDS. A lower threshold may result in more alerts, while a higher threshold will reduce alerts but might miss some threats.

Usage Instructions

1. **Running the IDS**

- Real-Time Monitoring:

To start real-time monitoring of network traffic, run the following command:

```
python real_time_detection.py
```

This will initiate the real-time detection engine, which will capture live network traffic, process it, and classify it using the machine learning models. The system will generate alerts for any detected intrusions or anomalies. The results of the classification and detected threats will be logged and displayed in the dashboard.

- Batch Processing (Offline Mode):

If you have previously collected network traffic logs or historical datasets, you can run the system in offline mode to analyze the data for any anomalies that might have been missed during real-time detection. Use the following command to process the offline data:

```
python offline_analysis.py --input data/offline_data.csv
```

This command processes the specified file and generates reports on any detected anomalies or intrusions. It is especially useful for post-incident analysis and for investigating network behavior over a specific period.

2. **Dashboard Usage**

- Launching the Dashboard:

The system includes a web-based dashboard that provides a visual interface for monitoring real-time network traffic and viewing detected threats. To launch the dashboard, use the following command:

```
python dashboard.py
```

Once the dashboard is running, open a web browser and navigate to http://localhost:5000 to access the interface. From here, you can view live traffic, review detected anomalies, and monitor the performance of the machine learning models. The dashboard also includes options for configuring system settings and managing alerts.

3. Managing Alerts and Notifications

Alerts generated by the system can be configured to trigger real-time notifications via email or SMS. These notifications help ensure that administrators are immediately aware of potential security threats. The alerting system can be customized in the config.yaml file, where you can specify the thresholds, email addresses, and notification settings:

```
yaml
alert_threshold: 0.5
email_notifications: true
email_recipient: "admin@company.com"
```

Alerts are also logged in the logs/ directory for further analysis and investigation.

4. Updating and Retraining Models

One of the key features of the IDS is the ability to update and retrain the machine learning models with new datasets. This ensures that the system remains effective as network traffic patterns evolve. To retrain the models, follow these steps:

1. Place the new dataset in the data/ directory.

2. Run the training script to update the machine learning models:

```
python train_models.py
```

The system will use the new data to train and update the models, improving their ability to detect emerging threats. This retraining process is essential for keeping the IDS up-to-date with current attack vectors.

Maintenance Guide

Regular maintenance is critical to ensure that the IDS operates efficiently and provides accurate detection. This section outlines the key maintenance tasks that administrators should perform on a regular basis.

1. Backup and Recovery

It is essential to regularly back up the system's critical components, including model files, configuration files, and logs. This ensures that you can quickly recover the system in case of any failures or data corruption. Use the following command to create a backup of the relevant files:

```
tar -czvf backup.tar.gz models/ logs/ config.yaml
```

Store the backup file in a secure location, and create a regular backup schedule to ensure that your data is protected.

2. System Updates

The IDS system may receive updates that improve its functionality, performance, or security. To update the system, pull the latest version from the repository and install any new dependencies:

```
git pull
pip install -r requirements.txt
```

Ensure that all dependencies are up-to-date and that the system is running the latest version of the machine learning models.

3. Monitoring and Logs

The IDS generates logs for both detected threats and system operations. It is important to review these logs regularly to ensure that the system is functioning correctly and to investigate any detected intrusions. Logs are stored in the logs/ directory, and administrators should set up a log rotation mechanism to prevent the log files from consuming too much disk space over time.

Troubleshooting

This section provides a guide to common issues that users may encounter when using the IDS and offers solutions to resolve them.

Common Issues and Solutions

1. **Issue: The dashboard is not loading.**

Solution: Ensure that the dashboard service is running on port 5000 and that there are no conflicts with other services using the same port. You can also check the log files for any errors related to the dashboard service.

2. **Issue: No real-time detection alerts are being generated.**

Solution: Verify that the packet capture service is running and has the necessary permissions to capture live network traffic. Ensure that the alert threshold is configured correctly, and check the system logs for any errors.

3. **Issue: The system is running slowly during batch processing.**

Solution: This could be due to insufficient hardware resources. Check the system's CPU and memory usage, and consider upgrading the hardware or optimizing the data preprocessing pipeline.

Conclusion

The Machine Learning-based Intrusion Detection System (IDS) offers a powerful and flexible solution for network security. By leveraging advanced machine learning techniques, the system provides real-time monitoring, anomaly detection, and post-incident analysis. With its customizable alerting system, web-based dashboard, and support for model retraining, the IDS is well-suited for organizations looking to enhance their cybersecurity defenses.

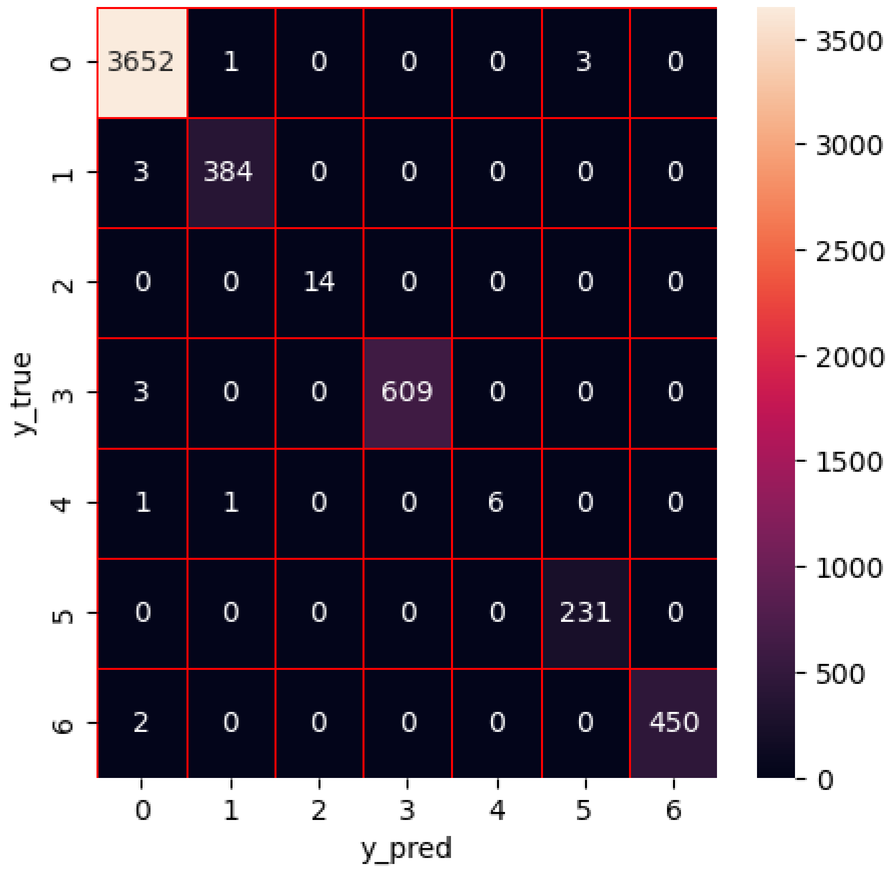
Results

Recall of LightGBM: 0.9973880597014926

Average F1 of LightGBM: 0.9973623367605119

F1 of LightGBM for each type of attack:

```
[0.99822332 0.99353169 1. 0.9975430.85714286 0.99354839 0.99778271]
```



Classification Report:

```
precision recallf1-scoresupport


 01.00 1.00 1.00 3656

 10.99 0.99 0.99387

 21.00 1.00 1.0014

 31.00 1.00 1.00612

 41.00 0.75 0.86 8

 50.99 1.00 0.99231

 61.00 1.00 1.00452
```

accuracy1.00 5360

macro avg1.00 0.96 0.98 5360

weighted avg1.00 1.00 1.00 5360

Accuracy of XGBoost:0.9970149253731343

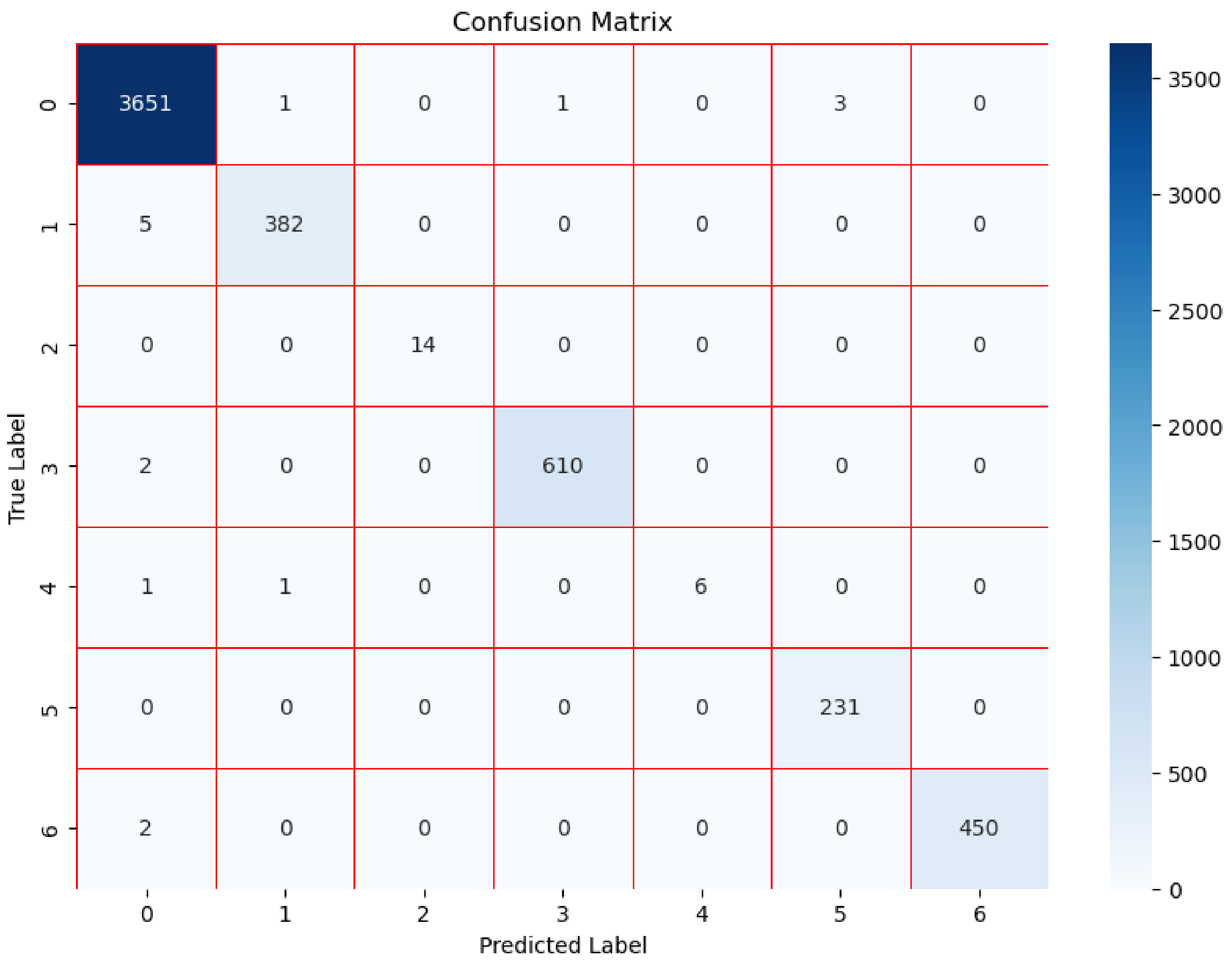
Precision of XGBoost:0.9970214286096803

Recall of XGBoost:0.9970149253731343

Average F1 of XGBoost:0.9969878512805174

F1 of XGBoost for each type of attack:

[0.99794998 0.99092088 1. 0.99754702 0.85714286 0.99354839 0.99778271]



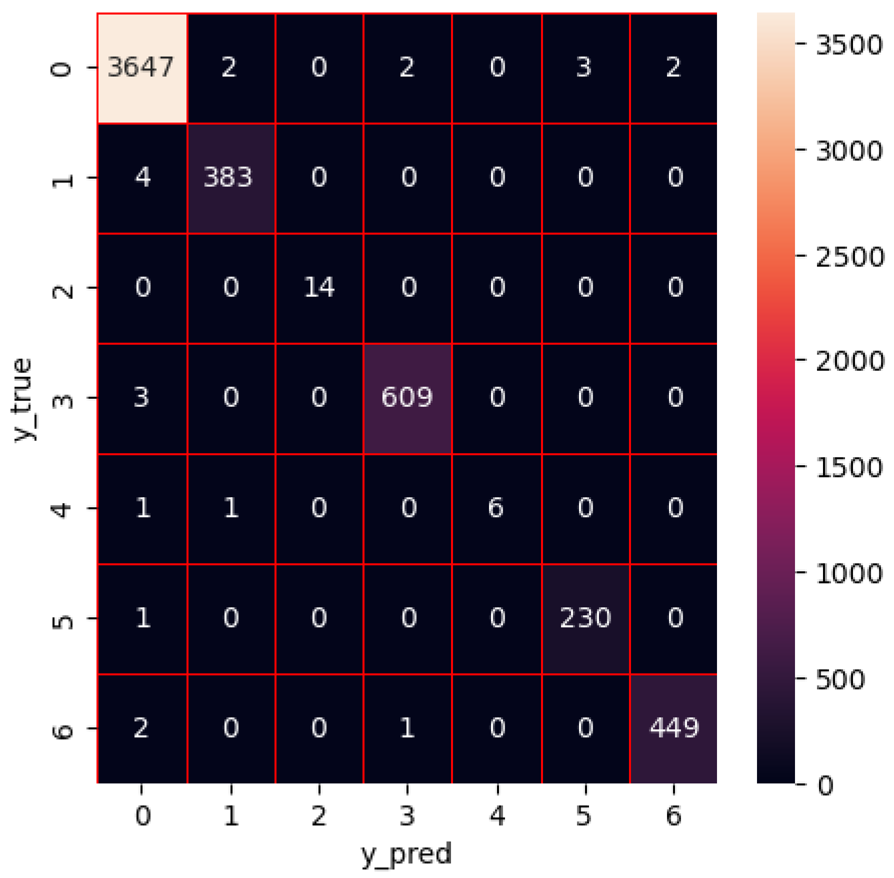
True \ Predicted	0	1	2	3	4	5	6
0	3651	1	0	1	0	3	0
1	5	382	0	0	0	0	0
2	0	0	14	0	0	0	0
3	2	0	0	610	0	0	0
4	1	1	0	0	6	0	0
5	0	0	0	0	0	231	0
6	2	0	0	0	0	0	450

precision recallf1-scoresupport

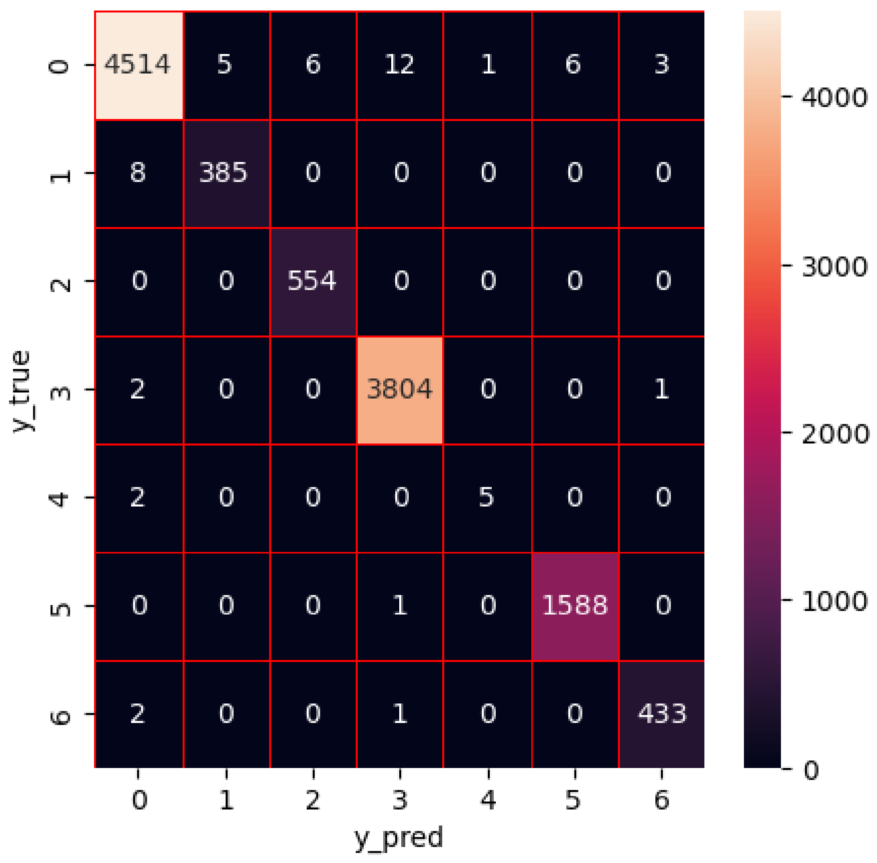
01.00 1.00 1.00 3656

10.99 0.99 0.99387

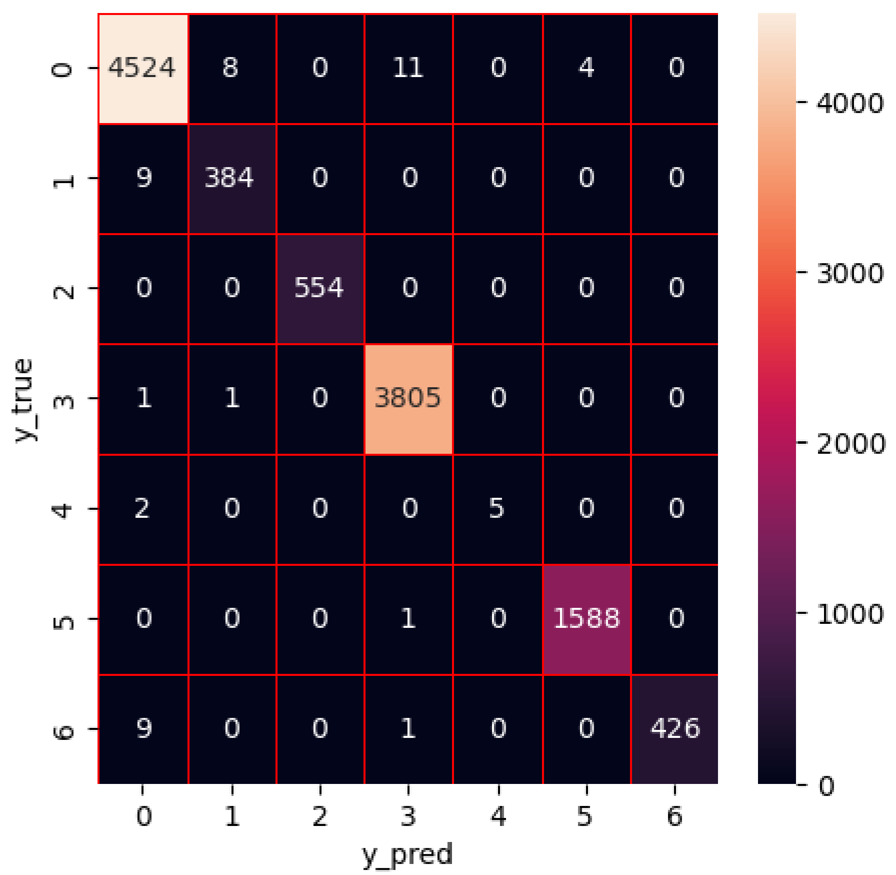
```
 21.00 1.00 1.0014
 31.00 1.00 1.00612
 41.00 0.75 0.86 8
 50.99 1.00 0.99231
 61.00 0.99 0.99452

accuracy1.00 5360
macro avg1.00 0.96 0.98 5360
weighted avg1.00 1.00 1.00 5360

Accuracy of CatBoost: 0.9958955223880597
Precision of CatBoost: 0.995899171395122
Recall of CatBoost: 0.9958955223880597
Average F1 of CatBoost: 0.9958696271764219
F1 of CatBoost for each type of attack:
[0.99726552 0.99094437 1. 0.99509804 0.85714286 0.99137931 0.9944629]
```



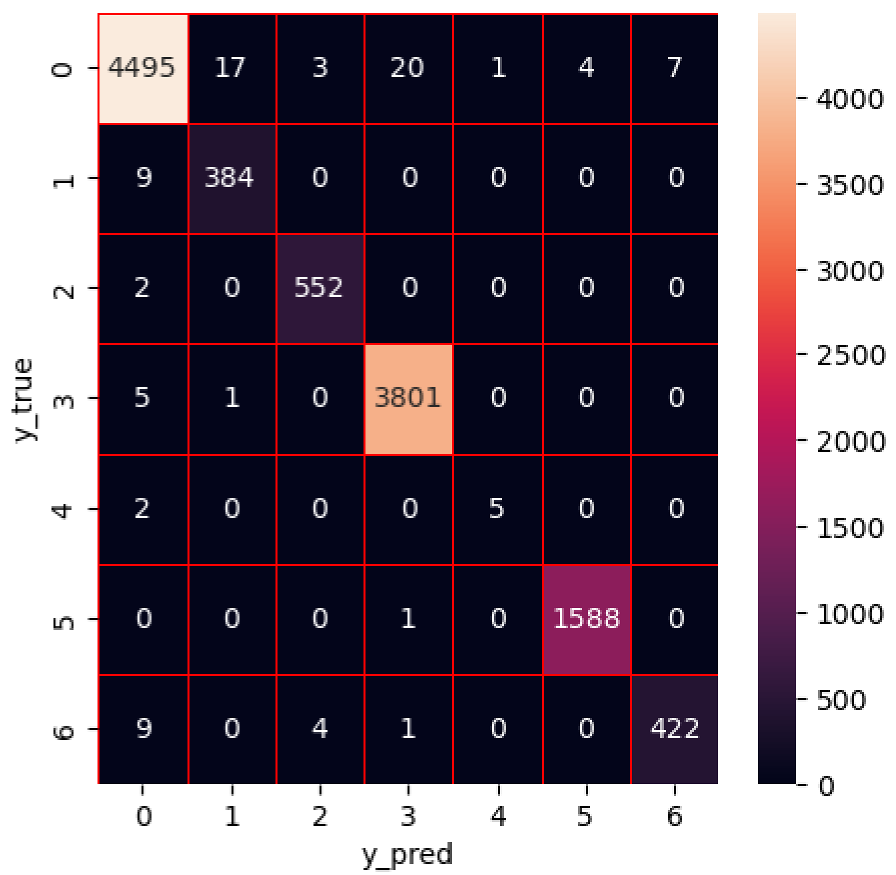
```
Accuracy of DT: 0.9955881055325156
Precision of DT: 0.9955765217572947
Recall of DT: 0.9955881055325156
F1-score of DT: 0.9955747319751295
 precision recallf1-scoresupport

 01.00 0.99 0.99 4547
 10.99 0.98 0.98393
 20.99 1.00 0.99554
 31.00 1.00 1.00 3807
 40.83 0.71 0.77 7
 51.00 1.00 1.00 1589
 60.99 0.99 0.99436

accuracy1.0011333
macro avg0.97 0.95 0.9611333
weighted avg1.00 1.00 1.0011333
```



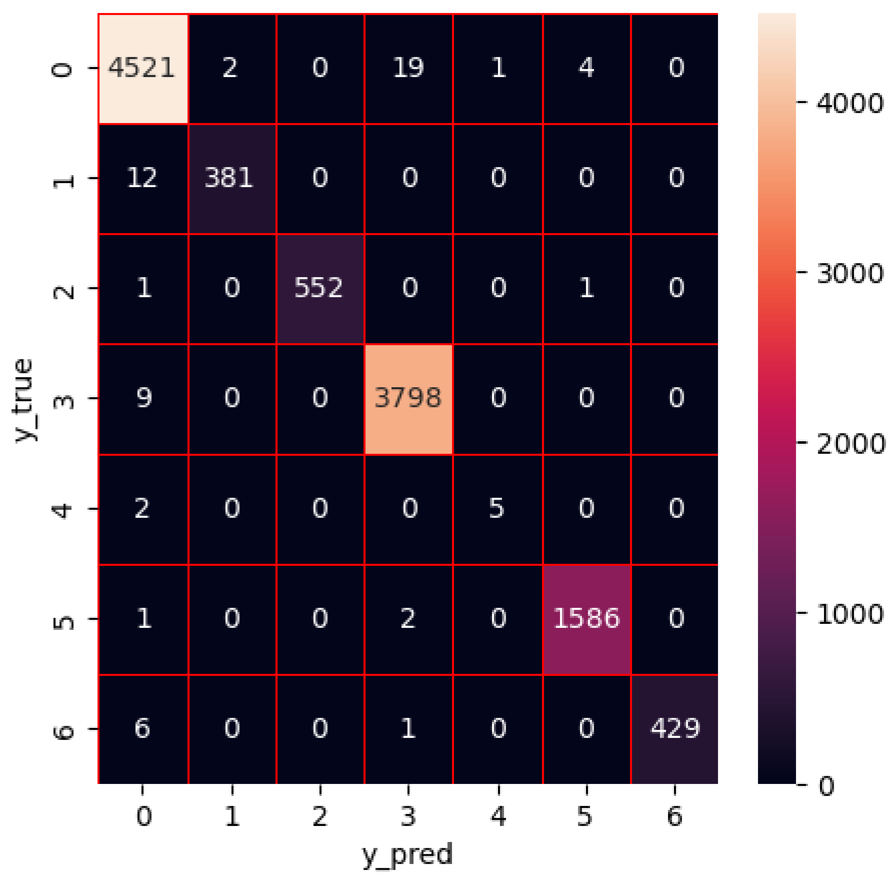
```
Accuracy of RF: 0.9958528192005647

Precision of RF: 0.9958559737870673

Recall of RF: 0.9958528192005647

F1-score of RF: 0.9958337304063318

 precision recallf1-scoresupport


 01.00 0.99 1.00 4547

 10.98 0.98 0.98393

 21.00 1.00 1.00554

 31.00 1.00 1.00 3807

 41.00 0.71 0.83 7

 51.00 1.00 1.00 1589

 61.00 0.98 0.99436


 accuracy1.0011333

 macro avg1.00 0.95 0.9711333

 weighted avg1.00 1.00 1.0011333
```



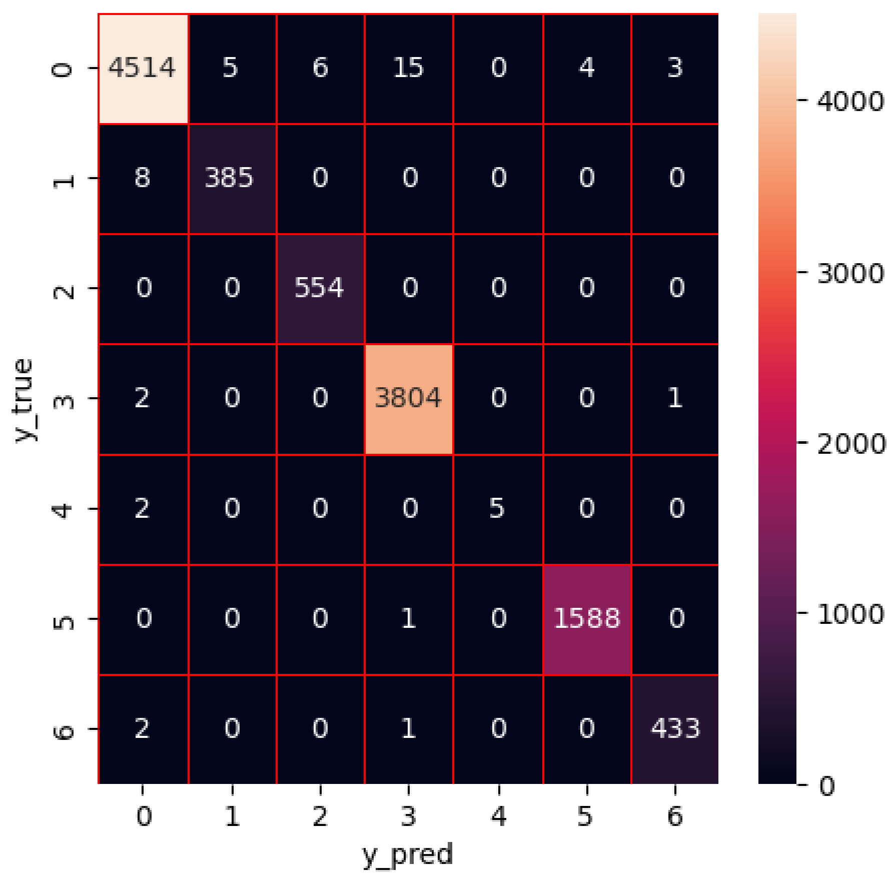
```
Accuracy of ET: 0.9924115415159269

Precision of ET: 0.9924234582936363

Recall of ET: 0.9924115415159269

F1-score of ET: 0.9924023222656908

precision recallf1-scoresupport


 00.99 0.99 0.99 4547

 10.96 0.98 0.97393

 20.99 1.00 0.99554

 30.99 1.00 1.00 3807

 40.83 0.71 0.77 7

 51.00 1.00 1.00 1589

 60.98 0.97 0.98436


 accuracy0.9911333

 macro avg0.96 0.95 0.9611333

 weighted avg0.99 0.99 0.9911333
```



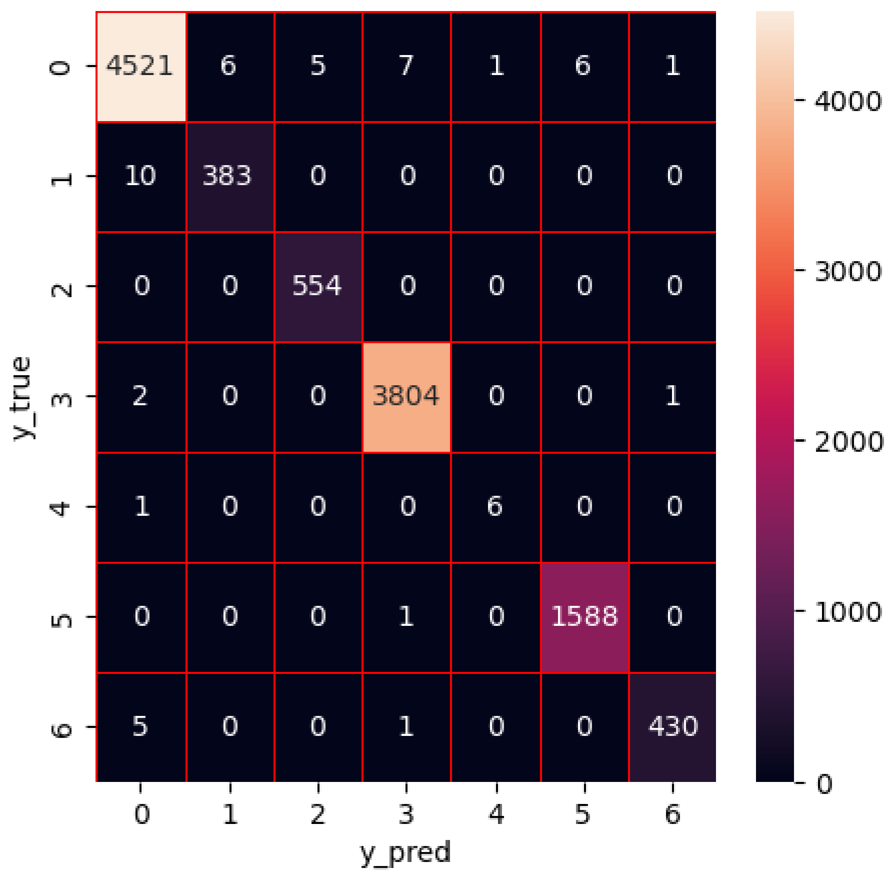
```
Accuracy of XGBoost: 0.9946174887496692

 Precision of XGBoost: 0.9946083401962289

 Recall of XGBoost: 0.9946174887496692

 F1-score of XGBoost: 0.9946006241300753

  precision recallf1-scoresupport


  00.99 0.99 0.99 4547

  10.99 0.97 0.98393

  21.00 1.00 1.00554

  30.99 1.00 1.00 3807

  40.83 0.71 0.77 7

  51.00 1.00 1.00 1589

  61.00 0.98 0.99436


  accuracy0.9911333

 macro avg0.97 0.95 0.9611333

 weighted avg0.99 0.99 0.9911333
```



```
Accuracy of Stacking: 0.9955881055325156

 Precision of Stacking: 0.9955921695013356

 Recall of Stacking: 0.9955881055325156

 F1-score of Stacking: 0.9955704673390741

  precision recallf1-scoresupport


  01.00 0.99 0.99 4547

  10.99 0.98 0.98393

  20.99 1.00 0.99554

  31.00 1.00 1.00 3807

  41.00 0.71 0.83 7

  51.00 1.00 1.00 1589

  60.99 0.99 0.99436


  accuracy1.0011333

 macro avg0.99 0.95 0.9711333

 weighted avg1.00 1.00 1.0011333
```

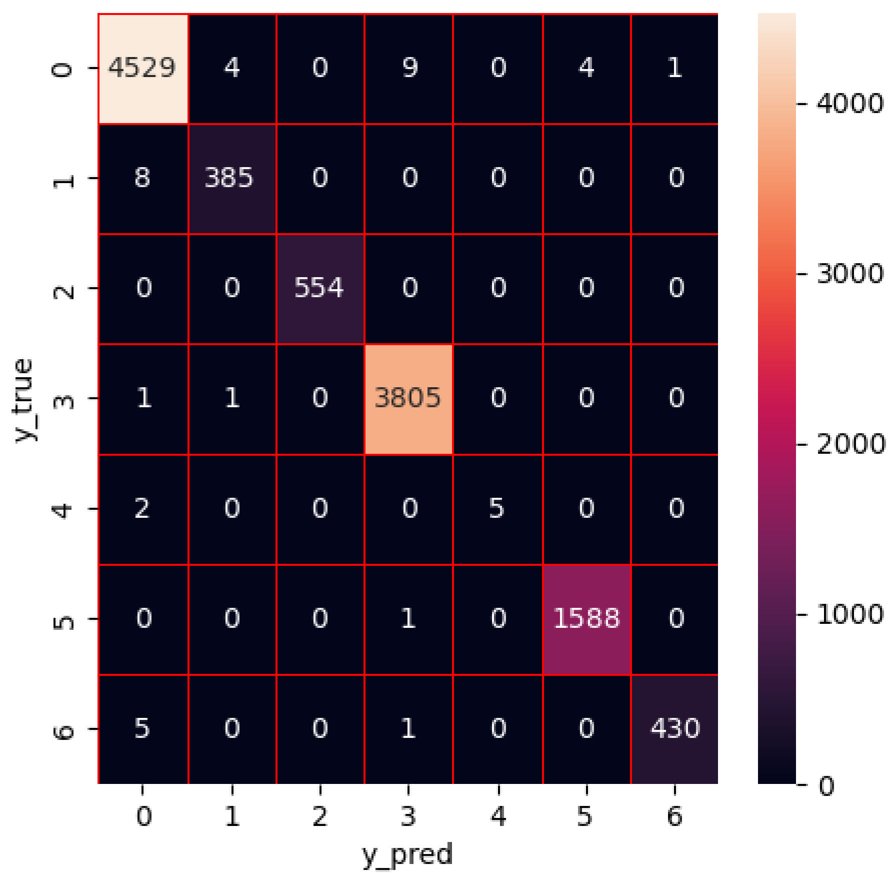


```
Accuracy of DT: 0.9958528192005647

 Precision of DT: 0.9958497983894136

 Recall of DT: 0.9958528192005647

 F1-score of DT: 0.9958477657212498

  precision recallf1-scoresupport


  01.00 0.99 1.00 4547

  10.98 0.97 0.98393

  20.99 1.00 1.00554

  31.00 1.00 1.00 3807

  40.86 0.86 0.86 7

  51.00 1.00 1.00 1589

  61.00 0.99 0.99436


  accuracy1.0011333

 macro avg0.97 0.97 0.9711333

 weighted avg1.00 1.00 1.0011333
```

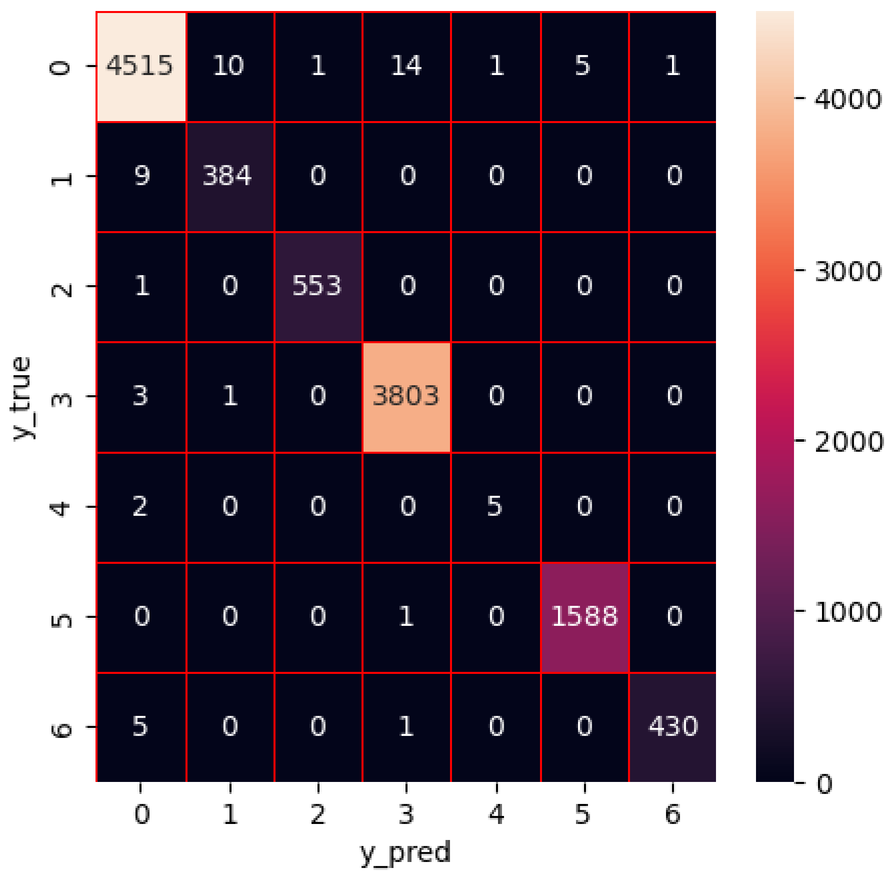


	0	1	2	3	4	5	6
0	4521	6	5	7	1	6	1
1	10	383	0	0	0	0	0
2	0	0	554	0	0	0	0
3	2	0	0	3804	0	0	1
4	1	0	0	0	6	0	0
5	0	0	0	1	0	1588	0
6	5	0	0	1	0	0	430

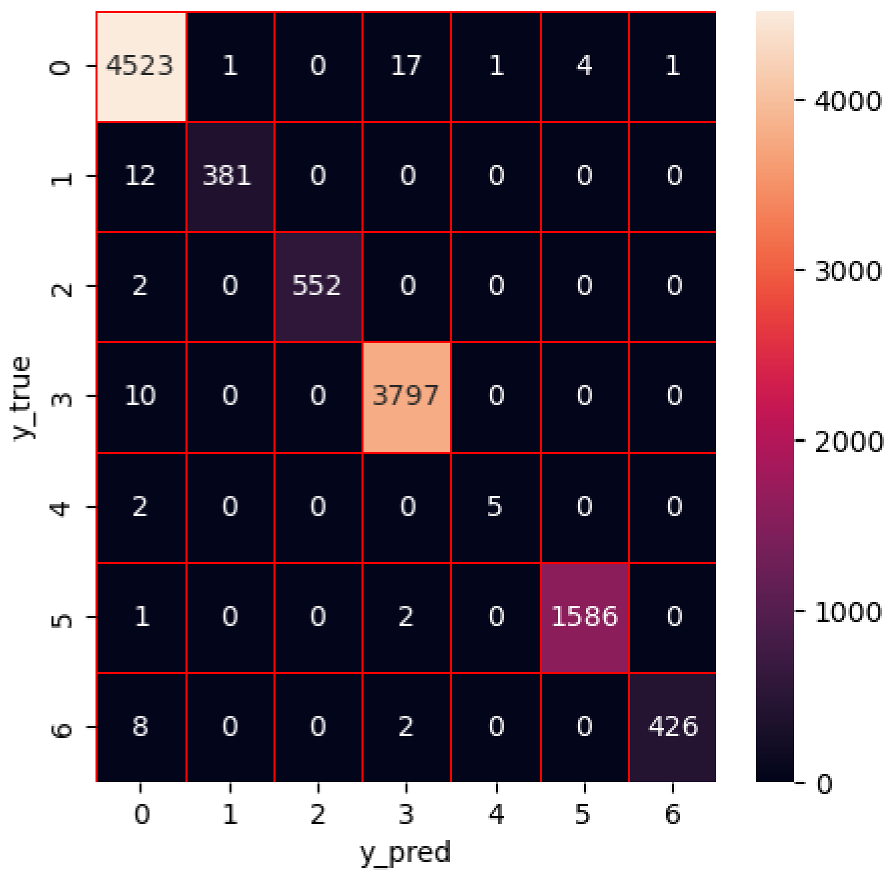
```
Accuracy of RF: 0.9967351980940616
 Precision of RF: 0.9967331137310563
 Recall of RF: 0.9967351980940616
 F1-score of RF: 0.9967170684652156
  precision recallf1-scoresupport


  01.00 1.00 1.00 4547
  10.99 0.98 0.98393
  21.00 1.00 1.00554
  31.00 1.00 1.00 3807
  41.00 0.71 0.83 7
  51.00 1.00 1.00 1589
  61.00 0.99 0.99436


  accuracy1.0011333
 macro avg1.00 0.95 0.9711333
 weighted avg1.00 1.00 1.0011333
```



```
Accuracy of ET: 0.9951469160857672

Precision of ET: 0.9951369749189738

Recall of ET: 0.9951469160857672

F1-score of ET: 0.995135883238205

 precision recallf1-scoresupport


 01.00 0.99 0.99 4547

 10.97 0.98 0.97393

 21.00 1.00 1.00554

 31.00 1.00 1.00 3807

 40.83 0.71 0.77 7

 51.00 1.00 1.00 1589

 61.00 0.99 0.99436


 accuracy1.0011333

macro avg0.97 0.95 0.9611333

weighted avg1.00 1.00 1.0011333
```

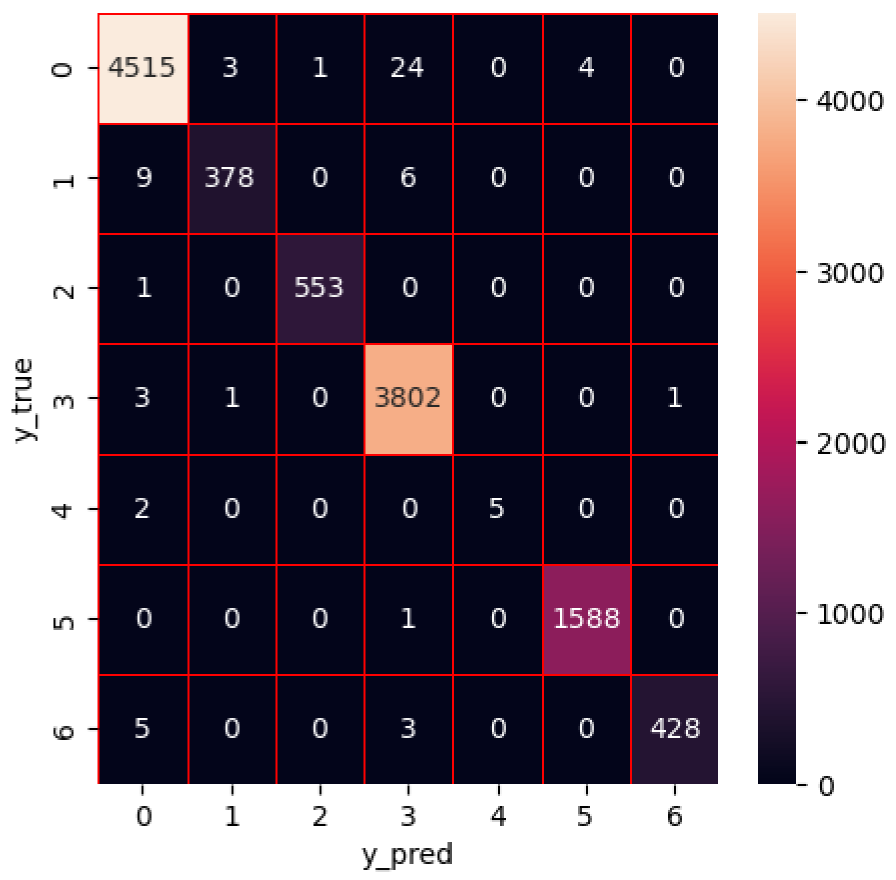


```
Accuracy of XGBoost: 0.9944410129709698
Precision of XGBoost: 0.9944349197665435
Recall of XGBoost: 0.9944410129709698
F1-score of XGBoost: 0.9944226852852549
 precision recallf1-scoresupport


 00.99 0.99 0.99 4547
 11.00 0.97 0.98393
 21.00 1.00 1.00554
 30.99 1.00 1.00 3807
 40.83 0.71 0.77 7
 51.00 1.00 1.00 1589
 61.00 0.98 0.99436


 accuracy0.9911333
macro avg0.97 0.95 0.9611333
weighted avg0.99 0.99 0.9911333
```



	0	1	2	3	4	5	6
0	4523	1	0	17	1	4	1
1	12	381	0	0	0	0	0
2	2	0	552	0	0	0	0
3	10	0	0	3797	0	0	0
4	2	0	0	0	5	0	0
5	1	0	0	2	0	1586	0
6	8	0	0	2	0	0	426

```
Accuracy of Stacking: 0.99435277508162

Precision of Stacking: 0.9943598477268112

Recall of Stacking: 0.99435277508162

F1-score of Stacking: 0.9943266630011871

 precision recallf1-scoresupport


 01.00 0.99 0.99 4547

 10.99 0.96 0.98393

 21.00 1.00 1.00554

 30.99 1.00 0.99 3807

 41.00 0.71 0.83 7

 51.00 1.00 1.00 1589

 61.00 0.98 0.99436


 accuracy0.9911333

macro avg1.00 0.95 0.9711333

weighted avg0.99 0.99 0.9911333
```



y_true \ y_pred	0	1	2	3	4	5	6
0	4515	3	1	24	0	4	0
1	9	378	0	6	0	0	0
2	1	0	553	0	0	0	0
3	3	1	0	3802	0	0	1
4	2	0	0	0	5	0	0
5	0	0	0	1	0	1588	0
6	5	0	0	3	0	0	428

Appendix

Glossary

- Bagging
  An ensemble technique that improves model accuracy by averaging multiple models trained on random data subsets.
- CatBoost
  A gradient boosting library optimized for categorical features.
- Classification Report
  A summary of model performance, showing precision, recall, and f1-score.
- Confusion Matrix
  A table showing actual vs. predicted labels, broken down into TP, TN, FP, and FN.
- Cross-Validation
  A technique for evaluating model performance by training on subsets of data.
- Decision Tree
  A model that splits data into branches based on feature values.
- F1 Score
  The harmonic mean of precision and recall, useful for imbalanced datasets.
- Feature Importance
  A ranking of how much each feature contributes to model predictions.
- Gradient Boosting
  A technique where each new model corrects errors made by the previous one (e.g., XGBoost, LightGBM, CatBoost).
- Imbalanced Dataset
  A dataset where classes are not equally represented (e.g., more benign traffic than attacks).
- Intrusion Detection System (IDS)
  A system designed to detect unauthorized access or attacks on networks.
- LightGBM
  A fast, efficient gradient boosting framework for large datasets.
- Overfitting
  When a model learns noise instead of patterns, leading to poor generalization.
- Precision
  The ratio of true positives to all predicted positives.
- Random Forest
  An ensemble method using multiple decision trees to improve accuracy and reduce overfitting.
- Recall
  The ratio of true positives to all actual positives, critical in intrusion detection.
- ROC Curve
  A plot of true positive rate vs. false positive rate for different classification thresholds.
- SMOTE
  An oversampling technique for handling imbalanced datasets by generating synthetic samples.
- Supervised Learning
  Training a model on labeled data where the correct output is known.
- XGBoost
  A powerful gradient boosting algorithm known for performance and scalability.