electronics

MDPI

Article

Applying Multi-CLASS Support Vector Machines: One-vs.-One vs. One-vs.-All on the UWF-ZeekDataFall22 Dataset

Rocio Krebs [1], Sikha S. Bagui [1,*], Dustin Mink [2] and Subhash C. Bagui [3]

[1] Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA; rk50@students.uwf.edu
[2] Department of Cybersecurity, University of West Florida, Pensacola, FL 32514, USA; dmink@uwf.edu
[3] Department of Mathematics and Statistics, University of West Florida, Pensacola, FL 32514, USA; sbagui@uwf.edu
* Correspondence: bagui@uwf.edu

Abstract: This study investigates the technical challenges of applying Support Vector Machines (SVM) for multi-class classification in network intrusion detection using the UWF-ZeekDataFall22 dataset, which is labeled based on the MITRE ATT&CK framework. A key challenge lies in handling imbalanced classes and complex attack patterns, which are inherent in intrusion detection data. This work highlights the difficulties in implementing SVMs for multi-class classification, particularly with One-vs.-One (OvO) and One-vs.-All (OvA) methods, including scalability issues due to the large volume of network traffic logs and the tendency of SVMs to be sensitive to noisy data and class imbalances. SMOTE was used to address class imbalances, while preprocessing techniques were applied to improve feature selection and reduce noise in the data. The unique structure of network traffic data, with overlapping patterns between attack vectors, posed significant challenges in achieving accurate classification. Our model reached an accuracy of over 90% with OvO and over 80% with OvA, demonstrating that despite these challenges, multi-class SVMs can be effectively applied to complex intrusion detection tasks when combined with appropriate balancing and preprocessing techniques.

Keywords: multi-class classification; SVM; machine learning; network log analysis; imbalanced datasets; supervised learning

Citation: Krebs, R.; Bagui, S.S.; Mink, D.; Bagui, S.C. Applying Multi-Class Support Vector Machines: One-vs.-One vs. One-vs.-All on the UWF-ZeekDataFall22 Dataset. *Electronics* **2024**, *13*, 3916. https://doi.org/10.3390/electronics13193916

Academic Editor:Wajeb Gharibi

Received: 26 August 2024
Revised: 26 September 2024
Accepted: 30 September 2024
Published: 3 October 2024

1. Introduction

To maintain the security and integrity of information systems, it is necessary to effectively detect and classify cyber attacks on networks. Classifying cyber attacks on network logs requires developing sophisticated techniques to identify a thread and correctly specify the kind of threat. Traditional methods of intrusion detection rely on correct cyber threat classification. Machine learning, particularly Support Vector Machines (SVMs), is a powerful tool that identifies network threats and anomalies in large datasets.

This research explores the use of SVMs for the multi-class classification of cyber attacks using the UWF-ZeekDataFall22 dataset [1]. This dataset, provided by the University of West Florida, comprises comprehensive network log data labeled according to the MITRE ATT&CK framework, a globally recognized knowledge base of adversary tactics and techniques [2]. The dataset aligns with real-world cyber threat scenarios, making it an ideal candidate for evaluating machine learning models.

Multi-class classification with SVMs is achievable using the following strategies: One-vs.-One (OvO) and One-vs.-All (OvA). The OvO strategy involves training a separate classifier for each pair of classes, resulting in many binary classifiers that collectively handle the multi-class problem. This method offers more precise decision boundaries but at the cost of increased computational complexity. Contrarily, the OvA strategy simplifies the multi-class problem by training a single classifier per class, distinguishing one class from

all others. OvA is computationally less intensive, but its performance is low when trained with imbalanced datasets.

The main goal of this study is to perform a comparative analysis of the OvO and OvA strategies in classifying attack tactics on network logs. By leveraging the UWF-ZeekDataFall22 dataset labeled with the help of the MITRE ATT&CK framework [3], we intend to evaluate these strategies' effectiveness in accurately classifying various types of cyber attacks. We will use the following performance metrics: accuracy, precision, recall, F1-score, and confusion matrix, provided by Scikit-learn [4], to assess and compare the models.

This research contributes to cybersecurity by providing an overview of the strengths and limitations of different SVM-based multi-class classification strategies. The findings will inform the development of more robust and efficient intrusion detection systems, ultimately enhancing organizations' ability to defend against sophisticated cyber threats.

The rest of this paper is organized as follows: Section 2 reviews related works, focusing on approaches to multi-class classification and handling imbalanced datasets. Section 3 introduces the UWF-ZeekDataFall22 dataset and its key characteristics. Section 4 describes the preprocessing steps used to improve model performance. Section 5 details the implementation of the multi-class SVM algorithms, specifically the One-vs.-One (OvO) and One-vs.-All (OvA) strategies. Section 6 presents the experimental results, including performance metrics and comparisons. Section 7 discusses the findings and their implications. Finally, Section 8 concludes the paper and suggests directions for future work.

2. Related Works

Multi-class classification using SVMs has been extensively studied. Table 1 provides an overview of various approaches to addressing multi-class problems and handling imbalanced datasets.

However, an emerging direction in Intrusion Detection Systems (IDS) research focuses on detecting unknown threats through unsupervised learning techniques. Unlike supervised methods like SVM, which rely on labeled data for training, unsupervised methods are designed to identify novel or previously unseen attacks without the need for predefined attack labels.

Mirsky et al. (2018) propose Kitsune, an ensemble of autoencoders, for online network intrusion detection [5]. This unsupervised method uses a collection of autoencoders to learn the normal behavior of network traffic. When an anomaly arises, Kitsune can detect deviations from learned patterns, making it effective for identifying previously unknown threats. This is a significant advantage over traditional SVM-based approaches, which require labeled data for training and may struggle with novel attack types. Kitsune's ability to adapt to new attacks in real time demonstrates the power of unsupervised learning for zero-day attack detection.

Similarly, Bovenzi et al. (2023) explore deep-learning-based anomaly detection in Internet of Things (IoT) environments [6]. Their comparison of various deep learning architectures highlights the robustness of unsupervised models in detecting anomalies in network traffic. Specifically, their study reveals that these models can outperform traditional classifiers in terms of robustness and generalizability when faced with previously unseen data. The use of deep learning techniques, such as autoencoders and Generative Adversarial Networks (GANs), allows for a more flexible and scalable approach to IDS, addressing the challenges posed by dynamic and evolving cyber threats.

The CMSVM algorithm proposed by Shi Dong et al. [7] uses active learning to assign weights dynamically to applications. This algorithm can manage imbalanced datasets, which is crucial for security and network monitoring applications. Compared to traditional methods, CMSVM reduces computation costs, improves classification accuracy, and solves the imbalance problem in network traffic identification. This approach benefits applications using dynamic unknown port numbers, masquerading, and encryption techniques.

Table 1. Summary of Related Works.

Study	Technique	Focus/Objective	Key Contribution
Shi Dong [7]	CMSVM	Active learning to manage imbalanced datasets	Improved classification accuracy and reduced computation costs for network traffic identification
Chih-Wei Hsu and Chih-Jen Lin [8]	SVM (OvA, OvO, DAGSVM)	Multi-class classification using SVM extensions	Comparison of different SVM-based approaches for computational efficiency and classifier performance
Bagui et al. [9]	BSMOTE	Resampling techniques for imbalanced network intrusion datasets	Optimized classification rates for rare attack identification
Kamil et al. [10]	Deep Learning	Intrusion detection using convolutional neural networks (CNN)	Enhanced detection of complex attack patterns using CNN architectures
Roy and Singh [11]	Random Forest	Multi-class classification for anomaly detection in network traffic	Utilized ensemble learning to improve detection accuracy in large-scale datasets
Mirsky et al. [5]	Autoencoders	Online detection of unknown attacks in real time	Developed Kitsune ensemble to detect zero-day attacks in network traffic without labeled data
Bovenzi et al. [6]	Deep Learning (Unsupervised)	Anomaly detection in IoT environments	Comparative analysis showing the robustness of unsupervised deep learning models in detecting unknown threats

According to Hsu and Lin [8], Support Vector Machines can be extended with the following methods: one-against-all, one-against-one, and directed acyclic graph SVM (DAGSVM). One-Against-All (OvA) trains a separate binary classifier for each class against all others, resulting in a number of classifiers equal to the number of classes. One-against-one (OvO) trains a binary classifier for each pair of classes, resulting in multiple classifiers for the pairs. Directed Acyclic Graph SVM (DAGSVM) uses a directed acyclic graph to structure the binary classifiers. During prediction, the graph is traversed from the root to a leaf node, reducing the number of comparisons needed and enhancing computational efficiency.

Additionally, Bagui et al. (2023) focus on identifying rare attacks in imbalanced network intrusion datasets by exploring different ratios of oversampled to undersampled data, finding that random undersampling before splitting yields better classification rates, while undersampling after oversampling with BSMOTE allows for lower ratios of oversampled data [9].

Kamil et al. [10] present a deep learning approach using convolutional neural networks (CNN) for intrusion detection. Their work focuses on enhancing the detection of complex attack patterns, showing that CNN architectures can effectively capture intricate features in network traffic data, leading to improved classification performance.

Roy and Singh [11] explore the use of Random Forest for multi-class classification in anomaly detection within network traffic. Their research demonstrates how ensemble learning techniques can be utilized to improve detection accuracy in large-scale datasets, making Random Forest a viable alternative to traditional SVM-based methods for certain types of network intrusion detection tasks.

Together, these studies underscore the versatility and effectiveness of multi-class SVMs in various applications, particularly in network security. They highlight the importance of combining SVMs with active learning to manage large datasets efficiently and the critical

role of kernel selection in optimizing classifier performance. The continuous development and application of these techniques are essential for advancing the field of network traffic and log file classification, providing robust solutions for cybersecurity challenges. Hence, this paper provides a comparison of binary and multi-class SVMs in classifying attack tactics in the newly created UWF-ZeekDataFall22 dataset.

3. Dataset

UWF-ZeekDataFall22 contains Zeek Conn logs collected from the Cyber Range at the University of West Florida (UWF). The dataset was labeled using the MITRE ATT&CK framework [3]. The dataset contains 700,340 logs with eleven categories. Of the eleven categories, ten are common tactics used in network cyberattacks.

In addition to containing a wide range of logs, the dataset represents various stages of network traffic, which are useful in detecting specific types of malicious behavior. This makes the dataset highly valuable for understanding and classifying cyberattacks, providing insights for network security applications. To provide a more comprehensive understanding of the dataset, we performed a detailed statistical analysis of its contents, summarized in Tables 2 and 3.

Table 2. Feature overview.

Feature	Description
Duration	The connection time in seconds.
Orig_bytes	The number of bytes sent by the originator.
Resp_bytes	The number of bytes sent by the responder.
Orig_pkts	The number of packets sent by the originator.
Resp_pkts	The number of packets sent by the responder.
Proto	The protocol used (TCP, UDP, etc.).
Service	The service requested (HTTP, FTP, etc.).
Conn_state	The connection state.
Label_tactic	The label corresponding to the MITRE ATT&CK tactic category.

Table 3. Descriptive statistics for numeric features.

Feature	Mean	Median	Std Dev	Min	Max
Duration	0.751 s	0.000016 s	10.83 s	0.000001 s	3560.86 s
Orig_bytes	123.58	102.00	1180.84	0	218,820
Resp_bytes	110.23	0.00	10,702.38	0	2,174,312
Orig_pkts	1.94	2.00	24.68	0	10,040
Resp_pkts	0.60	0.00	24.79	0	10,040
Src_port_zeek	47,435.10	47,125.00	9047.25	3	65,535
Dest_port_zeek	13,395.93	53.00	19,938.01	0	65,535
Missed_bytes	0.13	0.00	39.76	0	19,090
Orig_ip_bytes	133.43	134.00	2328.26	0	959,736
Resp_ip_bytes	89.77	0.00	11,678.85	0	4,870,648

These values reflect the skewed nature of network traffic, where a small subset of connections accounts for much larger or smaller values, significantly influencing the mean.

As observed in Table 4, tactics like Initial Access and Execution dominate the dataset, while categories such as Collection and Command and Control are underrepresented. This imbalance is critical when analyzing the dataset, as it may introduce challenges in obtaining accuracy in our models.

Table 4. Distribution of label_tactic categories.

Tactic	Count	Percentage
Resource Development	262,408	61.45%
None	117,242	27.46%
Reconnaissance	28,344	14.29%
Discovery	16,025	11.43%
Defense Evasion	2894	2.86%
Execution	29	1.43%
Initial Access	18	0.76%
Command and Control	14	0.57%
Lateral Movement	10	0.14%
Persistence	5	0.14%
Collection	1	0.14%

Figure 1 presents a tabular presentation of the eleven categories identified in the dataset. The categories are found in the column label_tactic. It can be noted that the dataset contains imbalanced classes. Our research will concentrate on the following categories: Resource Development, Reconnaissance, Discovery, and Defense Evasion. These categories contain a reasonable number of samples that can be used for effective model training.

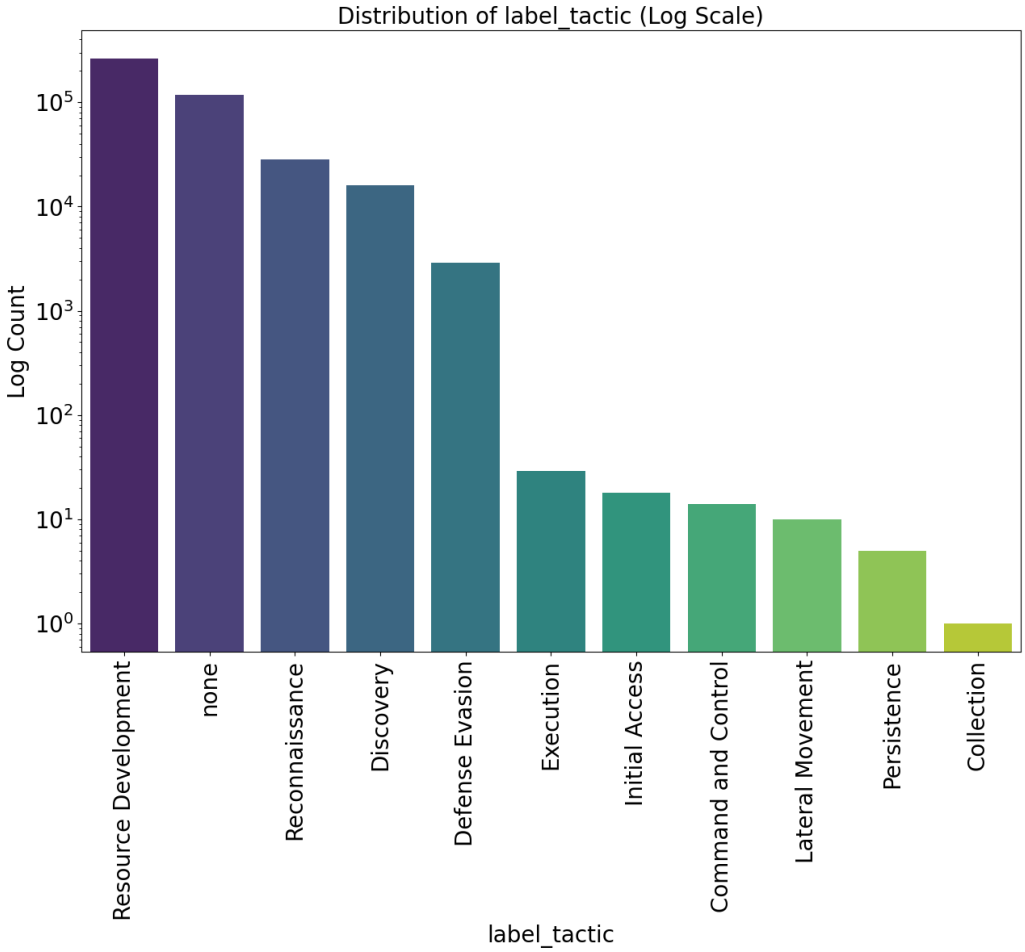

Figure 1. Distribution of label_tactic.

4. Preprocessing

Effective data preprocessing is crucial for ensuring the success of tactic classification. In this work, this process involved cleaning the data, reducing dimensionality, handling missing values, and balancing the dataset to achieve optimal performance when training

the Support Vector Machine Classifiers. The data in this dataset were in multiple Parquet files, which were loaded into dataframes using the Pandas library [12].

Duplicate rows were removed to prevent bias and overfitting in the model. To address missing data, we applied a backfilling method, propagating the next valid observation backward to fill the gaps. This approach helps maintain the dataset's continuity without introducing additional bias.

The `label_binary` column contained string values 'True' and 'False', which were converted to integers (1 and 0, respectively). This conversion was necessary to make the data suitable for machine learning algorithms, which typically require numerical input.

The tactics (or attack categories) with extremely few samples were dropped from the dataset. These categories are as follows: 'Execution', 'Initial Access', 'Command and Control', 'Lateral Movement', 'Persistence', and 'Collection'. By focusing on the most relevant categories, we aimed to improve the model's performance in the key classes of interest.

To reduce dimensionality and noise, several non-essential columns were dropped. The Random Forest classifier was used to perform feature selection. The Random Forest classifier was trained on the preprocessed dataset to calculate feature importance [13]. Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes of the individual trees [14]. One of the advantages of using Random Forest is its ability to estimate feature importance, which helps in identifying the most relevant features for our classification task. Features with an importance score below 0.05 were discarded, as they were considered less significant to the model's performance. This step reduced the dimensionality of the dataset and enhanced the model's interpretability and accuracy. Specifically, columns such as `duration`, `orig_bytes`, `resp_bytes`, `service`, `uid`, `ts`, `datetime`, `label_technique`, `community_id`, `src_ip_zeek`, and `dest_ip_zeek` were removed.

The selected features were scaled using StandardScaler to standardize the data distribution [15]. Categorical columns such as `conn_state`, `proto`, `history`, and `label_tactic` were encoded into numerical values using label encoding.

Stratified sampling was performed to ensure that each class in the dataset was proportionally represented [16]. This technique helps maintain the representative samples, which are important for the validity of the training and testing phases.

There was still class imbalance; therefore, the Synthetic Minority Over-Sampling Technique (SMOTE) [16] was employed. SMOTE generates synthetic samples for the minority classes by identifying the k-nearest neighbors and creating new samples along the line segments between the original samples and their neighbors. This technique helps balance the class distribution and improve the model's generalization ability across different classes.

After preprocessing, we performed k-fold cross-validation to evaluate the model's performance [17]. Additionally, manual hyperparameter tuning was conducted for the SVM classifier to optimize its parameters, further enhancing the model's accuracy and robustness. By carefully executing these preprocessing steps, we ensured that the data were in an optimal state for training the SVM classifier, leading to an improved performance in tactic classification.

Figure 2 presents the final distribution of the data after preprocessing and resampling. The dataset contains five distinct classes, which are the primary focus of our classification task. These classes are `Resource Development`, `None`, `Reconnaissance`, `Discovery`, and `Defense Evasion`. Each class represents a different type of network activity that is critical for intrusion detection tasks in cybersecurity. None is non-attack data.

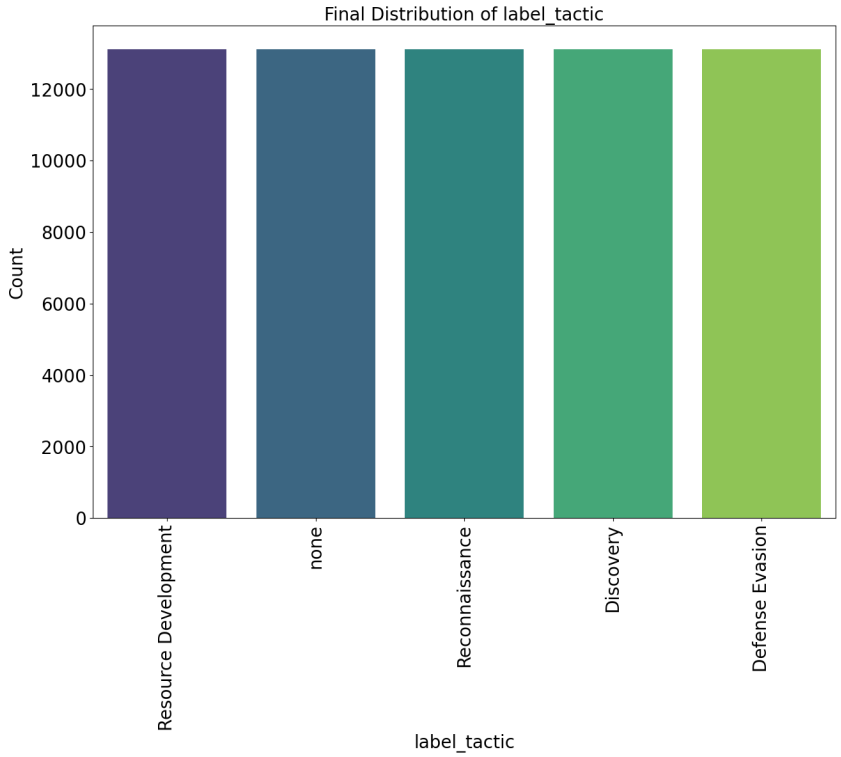

Figure 2. Final distribution of label_tatic after balancing, reduced to the most critical classes for intrusion detection.

5. Multi-Class SVM Algorithms

5.1. Suport Vector Machines (SVM)

Support vector machine (SVM) is a supervised machine learning algorithm, specially designed for binary classification. It was initially introduced as a support-vector network by Corinna Cortes and Vladimir Vapnik [18]; SVMs aim to find the optimal hyperplane that will successfully separate data into two different classes while maximizing the margin between the two classes. Mathematically, SVM is defined as an optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^2 + C \sum_{i=1}^{n} \xi_i, \tag{1}$$

subject to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \ldots, n \tag{2}$$

where C represents the regularization factor, w is the weight vector defining the orientation of the hyperplane, and b is the hyperplane's intercept.

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} a_i y_i K(x_i, x) + b \right) \tag{3}$$

where N is the number of support vectors, a_i are the Lagrange multipliers, b is the bias term, and $K(x_i, x)$ is the kernel function that computes the similarity between feature vectors x_i in a high-dimensional space [19].

After discussing the basic concepts behind Support Vector Machines (SVM), we can now look at how the algorithm is implemented. The procedure is outlined in Algorithm 1, which illustrates the key steps involved in training and using an SVM model.

Algorithm 1 Support vector machine (SVM) procedure.

```
procedure INIT
    lr ← 0.001
    λ ← 0.01
    n_iters ← 1000
    w ← None
    b ← None
end procedure
procedure FIT(X, y)
    n_samples, n_features ← shape(X)
    w ← zeros(n_features)
    b ← 0
    for _ ← 1 to n_iters do
        for idx, x_i in enumerate(X) do
            condition ← y[idx] · (dot(x_i, w) − b) ≥ 1
            if condition then
                w ← w − lr · (2 · λ · w)
            else
                w ← w − lr · (2 · λ · w − dot(x_i, y[idx]))
                b ← b − lr · y[idx]
            end if
        end for
    end for
end procedure
procedure PREDICT(X)
    approx ← dot(X, w) − b
    return sign(approx)
end procedure
```

We aimed to find the optimal hyperplane with Support Vector Machines (SVM) to maximize the margin between two classes. Our algorithm follows these steps:

5.1.1. Initialization

The SVM model is initialized with a learning rate (α), a regularization parameter (λ), and the number of iterations for training. The weights ($\mathbf{w}$) and bias (b) are initialized to zero.

5.1.2. Training the Model

The training process involves iteratively updating the weights and bias using gradient descent. For each sample i, the margin is calculated as follows:

$$\text{margin}_i = y_i(\mathbf{w}^\top \mathbf{x}_i - b) \tag{4}$$

where y_i is the true label of sample i and $\mathbf{x}_i$ is the feature vector of sample i.

If the sample is correctly classified ($\text{margin}_i \geq 1$), the weights are updated as follows:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha(2\lambda\mathbf{w}) \tag{5}$$

If the sample is misclassified ($\text{margin}_i < 1$), the weights and bias are updated as follows:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha(2\lambda\mathbf{w} - y_i\mathbf{x}_i) \tag{6}$$

$$b \leftarrow b + \alpha y_i \tag{7}$$

5.1.3. Making Predictions

The prediction for a new sample $\mathbf{x}$ is made by calculating the decision function:

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} - b \tag{8}$$

and taking the sign of the result as follows:

$$\hat{y} = \text{sign}(f(\mathbf{x})) \tag{9}$$

5.2. One-vs.-One (OvO) Approach

This approach allows us to perform multi-class classification using SVMs. According to Liu, Bi, and Fan [20], an n-class classification problem can be represented as a binary classification problem. Multiple binary classifiers are trained for pairwise comparisons between different classes in this approach.

The optimization problem for this approach can be formulated as follows:

$$\min_{w_{ij}, b_{ij}, \xi_{ij}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij}^2 + C \sum_{i=1}^{N} \sum_{j=i+1}^{N} \sum_{k=1}^{n} \xi_{ij}^{(k)} \tag{10}$$

subject to:

$$y_i(w_{ij} \cdot x_i + b_{ij}) \geq 1 - \xi_{ij}^{(k)}, \quad \text{for the } k\text{th sample of class } i \text{ vs. } j, \tag{11}$$

$$y_j(w_{ij} \cdot x_j + b_{ij}) \geq 1 - \xi_{ij}^{(k)}, \quad \text{for the } k\text{th sample of class } j \text{ vs. } i, \tag{12}$$

$$\xi_{ij}^{(k)} \geq 0, \quad \text{for } i = 1, 2, \ldots, N, \quad j = i+1, \ldots, N, \quad k = 1, 2, \ldots, n \tag{13}$$

Algorithm 2 outlines the implementation of the One-vs.-One (OvO) strategy for multi-class SVM classification, where a binary SVM is trained for each pair of classes:

Algorithm 2 Multi-class SVM OvO

```
procedure INIT
    lr ← 0.001
    λ ← 0.01
    n_iters ← 1000
    classifiers ← dict()
end procedure
procedure FIT(X, y)
    classes ← unique(y)
    for i ← 1 to len(classes) do
        for j ← i + 1 to len(classes) do
            class_i, class_j ← classes[i], classes[j]
            X_pair ← X[(y == class_i) ∨ (y == class_j)]
            y_pair ← y[(y == class_i) ∨ (y == class_j)]
            y_pair ← where(y_pair == class_i, 1, −1)
            svm ← SVM()
            svm.Fit(X_pair, y_pair)
            classifiers[(class_i, class_j)] ← svm
        end for
    end for
end procedure
procedure PREDICT(X)
    votes ← zeros(X.shape[0], len(unique(list(classifiers.keys()))))
    for (class_i, class_j), svm in classifiers.items() do
        preds ← svm.Predict(X)
        votes[:, class_i] ← votes[:, class_i] + (preds == 1)
        votes[:, class_j] ← votes[:, class_j] + (preds == −1)
    end for
    return argmax(votes, axis = 1)
end procedure
```

5.2.1. Initialization

The OvO SVM model is initialized with a learning rate, a regularization parameter, and the number of iterations for training. It also initializes a set of binary classifiers for each pair of classes.

5.2.2. Training the Model

A binary SVM classifier is trained for each pair of classes (i, j). The training samples are selected from the two classes, and their labels are converted to ± 1 (1 for class i and -1 for class j). The SVM classifier is trained using the selected samples and labels.

5.2.3. Making Predictions

To predict the class of a new sample, each binary classifier votes for one of the two classes it was trained on. The class with the most votes is chosen as the final prediction:

$$\hat{y} = \arg\max_{k} \sum_{(i,j)\in \text{pairs}} \mathbf{1}_{\hat{y}_{ij}=k} \tag{14}$$

5.3. One-vs.-All (OvA) Approach

The One-vs.-All (OvA) approach helps us solve multi-classification problems with the binary classifier SVM.

$$\min_{w_i, b_i, \xi_i} \frac{1}{2} \sum_{i=1}^{N} w_i^2 + C \sum_{i=1}^{N} \sum_{j\neq i}^{N} \sum_{k=1}^{n} \xi_i^{(k)} \tag{15}$$

subject to:

$$y_i(w_i \cdot x_i + b_i) \geq 1 - \xi_i^{(k)}, \quad \text{for } k\text{th of class } i, \tag{16}$$

$$\xi_i^{(k)} \geq 0, \quad \text{for } i = 1, 2, \ldots, N, \quad k = 1, 2, \ldots, n \tag{17}$$

Algorithm 3 outlines the implementation of the One-vs.-All (OvA) strategy for multi-class SVM classification, where a separate binary SVM is trained for each class against all other classes.

Algorithm 3 Multi-class SVM OvA

```
procedure INIT
    lr ← 0.001
    λ ← 0.01
    n_iters ← 1000
    classifiers ← list()
end procedure
procedure FIT(X, y)
    classes ← unique(y)
    for cls in classes do
        binary_y ← where(y == cls, 1, −1)
        svm ← SVM()
        svm.Fit(X, binary_y)
        classifiers.append(svm)
    end for
end procedure
procedure PREDICT(X)
    predictions ← zeros(X.shape[0], len(classifiers))
    for i, svm in enumerate(classifiers) do
        predictions[:, i] ← svm.Predict(X)
    end for
    return argmax(predictions, axis = 1)
end procedure
```

The One-vs.-All (OvA) approach involves training a binary SVM classifier for each class against all other classes.

5.3.1. Initialization

The OvA SVM model is initialized with a learning rate, regularization parameter, and the number of iterations for training. It also initializes a set of binary classifiers, one for each class.

5.3.2. Training the Model

A binary SVM classifier is trained for each class k. The training samples are labeled as 1 if they belong to class k and -1 otherwise. The SVM classifier is trained using these binary labels.

5.3.3. Making Predictions

To predict the class of a new sample, each binary classifier outputs a score indicating the likelihood of the sample belonging to the respective class. The class with the highest score is chosen as the final prediction:

$$\hat{y} = \arg\max_k f_k(\mathbf{x}) \tag{18}$$

where $f_k(\mathbf{x})$ is the decision function for class k.

6. Results

The performance of the Support Vector Machine (SVM) models was evaluated using multiple metrics, including accuracy, precision, recall, and F1-score. Below is a brief description of each metric:

- Accuracy: The ratio of correctly predicted instances to the total instances. This measures the overall effectiveness of the classifier.
- Precision: The proportion of true positives (correctly predicted positive cases) out of all predicted positives (true positives + false positives). Precision indicates how reliable the classifier's positive predictions are.
- F1-Score: The harmonic mean of precision and recall, providing a balance between the two metrics. This is useful when dealing with imbalanced datasets.
- Confusion Matrix: A table that describes the performance of the classifier by showing the counts of true positives, true negatives, false positives, and false negatives for each class.

1. Cross-Validation Accuracy

A five-fold cross-validation was used to assess the model performance and its consistency across different data splits. The accuracy scores obtained from each fold are shown in Table 5.

Table 5. Cross-Validation Accuracy Scores.

Fold	Accuracy
Fold 1	0.9679
Fold 2	0.9668
Fold 3	0.9664
Fold 4	0.9667
Fold 5	0.9633
Mean Accuracy	0.9662

These results indicate high consistency and reliability in the model's performance, 284 with a mean cross-validation accuracy of 0.96622 and a standard deviation of approximately

285, 0.0017. The standard deviation indicates that the model's performance is stable across 286 different subsets of the data. The resulting accuracy scores reflect the effectiveness of our 287 preprocessing steps, feature selection, and handling of class imbalance.

2. Hyperparameter Tuning

Hyperparameter tuning was manually conducted for both the One-vs.-One (OvO) and One-vs.-All (OvA) SVM classifiers. The same optimal parameters were identified for both classifiers, as shown below:

Learning rate: 0.01

Regularization parameter ($\lambda\lambda$): 0.01

Number of iterations: 2000

These parameters were chosen to ensure a balance between model complexity and the ability to generalize to unseen data. Proper tuning helped improve classification accuracy and prevent overfitting.

3. Performance Comparison Between OvO and OvA SVM Models

The performance metrics of the OvO and OvA SVM models are summarized in Table 6. The results show that the OvO model outperformed the OvA model in terms of accuracy and F1-score, with both models showing similar precision.

Table 6. Performance Metrics for OvO and OvA SVM Models.

Metric	OvO	OvA
Accuracy	0.8284	0.7715
Precision	0.8578	0.8582
Recall	0.8264	0.7689
F1-Score	0.7823	0.7295

The OvO model achieved an accuracy of 82.84% and an F1 Score of 78.23%, while the OvA model recorded an accuracy of 77.15% and an F1 Score of 72.95%. Although the precision was similar for both models, the OvO model showed better recall, indicating superior performance in identifying positive instances.

4. Confusion Matrices

Figures 3 and 4 display the confusion matrices for the OvO and OvA models. The OvO model demonstrated a stronger ability to differentiate between similar classes, such as 'Reconnaissance' and 'Discovery', whereas the OvA model showed more misclassifications.

The OvO model's confusion matrix (Figure 3) shows fewer misclassifications across classes, while the OvA model (Figure 4) exhibited more confusion between similar classes, particularly between 'Defense Evasion' and 'Reconnaissance'.

While the OvA classifier achieved a slightly lower accuracy of 77.15% compared to the OvO classifier, it maintained a high precision of 85.82%, reflecting strong reliability in its positive predictions. The recall was 76.89%, indicating a robust ability to identify positive instances. The F1-score of 72.95% suggests a balanced performance between precision and recall, though slightly lower than that of the OvO classifier.

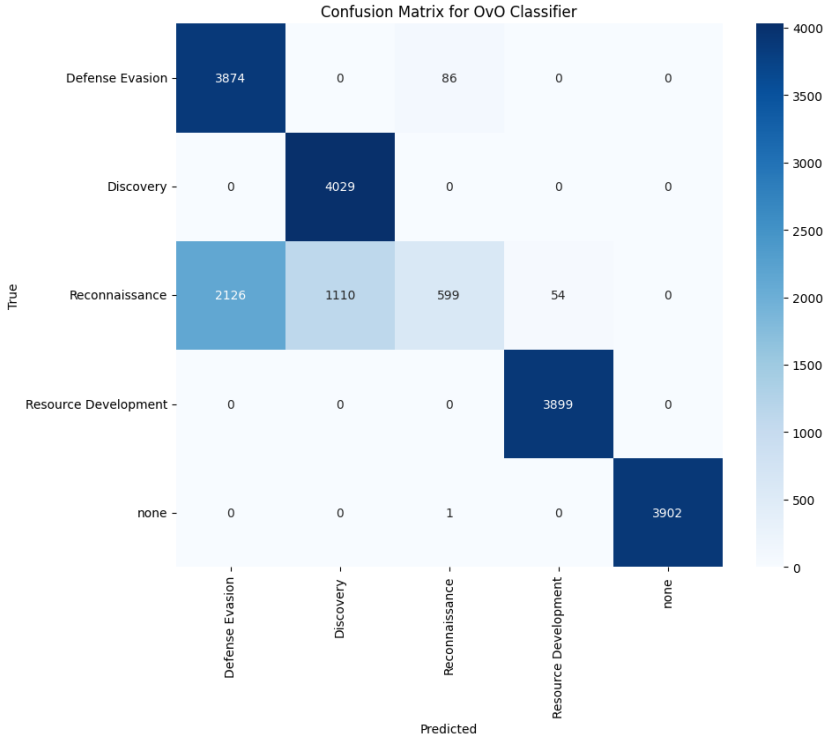

Figure 3. Confusion Matrix for OvO.

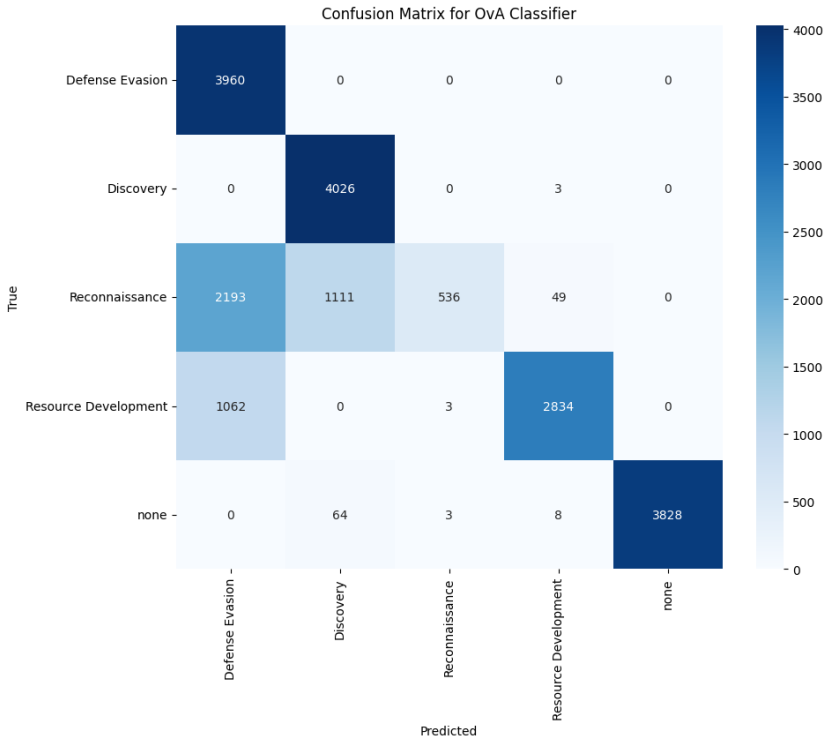

Figure 4. Confusion Matrix for OVA.

7. Discussion

The results of previous studies using the binary SVM classifier on this dataset have not been very consistent [3] and the multi-class SVM has not been applied to this dataset before. This study shows that SVM is effective for multi-class classification in network

intrusion detection, particularly when handling complex, imbalanced datasets like UWF-ZeekDataFall22. However, several challenges were identified.

A major challenge was the class imbalance, with tactics like 'Initial Access' and 'Command and Control' underrepresented. SMOTE helped improve performance for these minority classes, though it introduced some overfitting.

The OvO method performed better overall, with higher F1-scores and fewer misclassifications, while the OvA approach, though computationally cheaper, struggled more with class imbalance and similar classes, such as 'Reconnaissance' and 'Discovery'.

Overlapping patterns in network traffic data made it difficult to distinguish between attack types like 'Defense Evasion' and 'Discovery'. This suggests that more advanced feature engineering or time-series analysis could enhance classification.

Scalability is another challenge, especially for large datasets. The computational cost of training multiple binary classifiers in OvO is high. Future work could explore distributed computing frameworks like Apache Spark to improve scalability.

Despite noise-reduction efforts, residual noise in the network traffic logs likely impacted classification. More advanced noise-reduction techniques could further improve model robustness.

In summary, while SVMs are powerful for multi-class classification in intrusion detection, addressing class imbalance, overlapping data patterns, and scalability will be critical for future research. Ensemble learning or deep learning could provide further improvements for handling the complexities of network security.

8. Conclusions

The results demonstrate the efficacy of this study's preprocessing pipeline, feature selection, and class imbalance handling techniques. The OvO classifier outperformed the OvA classifier in terms of overall accuracy and F1-score, suggesting that the OvO approach may be more suitable for the multi-class classification of network log data, even after balancing the classes. Both classifiers, however, exhibited high precision, indicating strong reliability in positive predictions.

The consistent and high cross-validation accuracy highlight the robustness and reliability of the preprocessing and feature selection pipeline. The minimal variance in accuracy scores across different folds indicates that the model generalizes well to unseen data, providing insight into the possible outcomes if it is applied to real-world scenarios.

These findings provide a strong foundation for further enhancements and applications in network log analysis and cyberattack detection that requires multi-class classification. Future work could explore advanced techniques such as ensemble learning, deep learning approaches, or the further optimization of hyperparameters to improve classifier performance further.

Author Contributions: Conceptualization, R.K. and S.S.B.; methodology, R.K.; software, R.K.; validation, S.S.B., D.M. and S.C.B.; formal analysis, R.K.; investigation, R.K.; resources, S.S.B., D.M. and S.C.B.; data curation, R.K., S.S.B., D.M. and S.C.B.; writing—original draft preparation, R.K.; writing—review and editing, S.S.B., D.M. and S.C.B.; visualization, R.K.; supervision, S.S.B.; project administration, S.S.B., D.M. and S.C.B.; funding acquisition, S.S.B., D.M. and S.C.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Centers of Academic Excellence in Cybersecurity, NCAE-C-002: Cyber Research Innovation Grant Program, Grant Number: H98230-21-1-0170. This work was partially supported by the Askew Institute at The University of West Florida.

Data Availability Statement: The datasets are available at https://datasets.uwf.edu/ (accessed on 20 August 2023).

Acknowledgments: We would also like to thank the Askew Institute at University of West Florida for partially supporting this grant.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
OVO	One versus One
OVA	One Versus All
SVM	Support Vector Machines
UWF	University of West Florida
GAN	Generative Adversial Networks
IDS	Intrusion Detection Systems
CNN	Convolutional Neural Network
DAGSVM	Directed Acrylic Graph SVM
SMOTE	Synthetic Minority Over-sampling Technique
BSMOTE	Boderline SMOTE Algorithm

References

1. UWF-ZeekData22 Dataset. Available online: https://datasets.uwf.edu/ (accessed on 10 March 2024).
2. MITRE ATT&CK. Available online: https://attack.mitre.org/ (accessed on 10 March 2024).
3. Bagui, S.S.; Mink, D.; Bagui, S.C.; Madhyala, P.; Uppal, N.; McElroy, T.; Plenkers, R.; Elam, M.; Prayaga, S. Introducing the UWF-ZeekDataFall22 Dataset to Classify Attack Tactics from Zeek Conn Logs Using Spark's Machine Learning in a Big Data Framework. *Electronics* **2023**, *12*, 5039. [CrossRef]
4. Scikit-Learn. Available online: https://scikit-learn.org/ (accessed on 10 June 2024).
5. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv* **2018**, arXiv:1802.09089.
6. Bovenzi, G.; Aceto, G.; Ciuonzo, D.; Pescape, A. Network anomaly detection methods in IoT environments via deep learning: A fair comparison of performance and robustness. *Comput. Secur.* **2023**, *128*, 103167. [CrossRef]
7. Dong, S. Multi class SVM algorithm with active learning for network traffic classification. *Expert Syst. Appl.* **2021**, *176*, 114885. [CrossRef]
8. Hsu, C.-W.; Lin, C.-J. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Netw.* **2002**, *13*, 415–425. [PubMed]
9. Bagui, S.S.; Mink, D.; Bagui, S.C.; Subramaniam, S.; Wallace, D. Resampling Imbalanced Network Intrusion Datasets To Identify Rare Attacks. *Future Internet* **2023**, *15*, 130. [CrossRef]
10. Kamil, W.F.; Mohammed, I.J. Deep learning model for intrusion detection system utilizing convolution neural network. *Open Eng.* **2023**, 13, 20220403. [CrossRef]
11. Roy, A. and Singh, K.J. Multi-classification of UNSW-NB15 Dataset for Network Anomaly Detection System. In *Proceedings of International Conference on Communication and Computational Technologies, Jaipur, India, 30–31 August 2019*; Algorithms for Intelligent Systems; Springer: Singapore, 2021. [CrossRef]
12. Pandas-Parquet. Available online: https://pandas.pydata.org/docs/reference/api/pandas.read_parquet.html (accessed on 10 March 2024).
13. Feature Importances with a Forest of Trees. Available online: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html (accessed on 15 July 2024).
14. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
15. Preprocessing Data. Available online: https://scikit-learn.org/stable/modules/preprocessing.html (accessed on 20 April 2024).
16. Imbalanced-Learn Documentation. Available online: https://imbalanced-learn.org/stable/index.html (accessed on 25 July 2024).
17. Cross-Validation. Available online: https://scikit-learn.org/stable/modules/cross_validation.html (accessed on 29 July 2024).
18. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
19. Shawe-Taylor, J.; Cristianini, N. *Kernel Methods for Pattern Analysis*; Cambridge University Press: Cambridge, UK, 2004.
20. Liu, Y.; Bi, J.-W.; Fan, Z.-P. A method for multi-class sentiment classification based on an improved one-vs.-one (OVO) strategy and the support vector machine (SVM) algorithm. *Inf. Sci.* **2017**, *394–395*, 38–52. [CrossRef]