Shining New Light on Useful Features for Network Intrusion Detection Algorithms

Heather Lawrence, Uchenna Ezeobi, Gedare Bloom, Yanyan Zhuang
University of Colorado Colorado Springs
1420 Austin Bluffs Pkwy, Colorado Springs, CO, 80918, USA
hlawrenc@uccs.edu, uezeobi@uccs.edu, gbloom@uccs.edu, yzhuang@uccs.edu

***Abstract*—Network intrusion detection systems (NIDS) today must quickly provide visibility into anomalous behavior on a growing amount of data. Meanwhile different data models have evolved over time, each providing a different set of features to classify attacks. Defenders have limited time to retrain classifiers, while the scale of data and feature mismatch between data models can affect the ability to periodically retrain. Much work has focused on classification accuracy yet feature selection is a key part of machine learning that, when optimized, reduces the training time and can increase accuracy by removing poorly performing features that introduce noise. With a larger feature space, the pursuit of more features is not as valuable as selecting *better* features. In this paper, we use an ensemble approach of filter methods to rank features followed by a voting technique to select a subset of features. We evaluate our approach using three datasets to show that, across datasets and network topologies, similar features have a trivial effect on classifier accuracy after removal. Our approach identifies poorly performing features to remove in a classifier-agnostic manner that can significantly save time for periodic retraining of production NIDS.**

***Index Terms*—Network Intrusion Detection Systems, NIDS, Ensemble Feature Selection**

I. Introduction

Network intrusion detection systems (NIDS) leverage signature-based [1], specification-based [2]–[4], and anomaly-based techniques [5] to identify malicious traffic. Signature-based techniques are effective against known attacks but are unable to detect novel (zero-day) attacks. Specification-based techniques can detect novel attacks but require a subject matter expert to create a model of legitimate behavior. Anomaly-based techniques can also detect novel attacks but, being based on statistical machine learning classifier models, are prone to false positives that cause problems for usability like alert fatigue [6], [7]. In addition, anomaly-based NIDS face practical challenges in deploying state-of-the-art classifiers on modern networks and achieving reproducible results [8]. In an attempt to balance advantages and disadvantages, modern security solutions tend to use a hybrid approach with both signature- and anomaly-based techniques [9], [10] to detect suspicious activity.

This work is supported in part by NSF grants OAC-2115134, OAC-2001789, OAC-1920462, CNS-2046705, and Colorado State Bill 18-086.

Anomaly-based NIDS research relies on datasets to improve state-of-the-art classification techniques [11]. An evolving network traffic landscape not only requires that these datasets are created using rigorous metrics [12], but also causes datasets to become quickly outdated based on the protocols and their versions in use, traffic profiles, and attacks included. Hence, in evaluating anomaly-based NIDS, dataset selection plays a large role in reported performance. A mismatch between a chosen dataset and target environment undermines the validity of decisions made based on experimental results derived from that dataset. Further, feature selection in NIDS can be challenging due to an imbalance in the amount of attack and benign data as attack data can be harder to capture in bulk.

Datasets are primarily stored in two ways: in raw data (packet captures, or pcaps) or in a key-value store (a network flow, or netflow). In packet captures, the unencrypted portion of the network traffic provides more data to distill into features for machine learning and facilitates more custom use cases. However, this also requires that researchers architect their own network traffic parser to distill raw traffic into machine learning features or to rely on prior work [13], [14]. Prior work for packet capture parsers have extracted 27 fields [14] and 80 fields [13]. A network flow, or netflow, more formally defined as a set of packets or frames passing an *observation point* in the network during a specific time interval, describes network communication from a top-down conversational level. Netflows provide metadata about the conversation—such as the source and destination IP address—in lieu of the raw data, and therefore the original data cannot be restored [15]. Prior work for netflows has increased the feature space from 18 fields [16] to 491 fields [17]. The undetermined compatibility and efficacy of features across these approaches adds difficulty in evaluating and comparing experimental results. The challenges are particularly salient when approaches evaluate combined datasets to cover more attack patterns.

Regardless of the network capture data model, netflows or pcaps, network data follows a non-stationary distribution [14]. Anomaly detection models are considered quasi-stationary in that machine learning models can

978-1-6654-3161-3/22/$31.00 ©2022 IEEE

require retraining—daily [18], when a new attack is discovered [19], or on-demand [20]. These requirements place a constraint on model retraining time regardless of the data model or size of the feature space. Retraining time is defined as the amount of time spent training and testing classification algorithms using the k-fold cross-validation approach. As shown with the explosion of the feature space facilitated by the maturity of the netflow protocol to 491 mineable features [16], [17], there appears to be prioritization to increase the number of features over the need to select better features.

Effective feature selection leads to classifiers that require less memory and are faster to train and test, reduce feature extraction costs, and lead to better generalization [21]. While there is prior work in feature selection research that applies filter methods to determine impact on core metrics, the premise of our work attempts to show a pattern of features that are poorly associated with the label across common datasets. A feature's poor performance may be due describing a network artifact with little correlation to the label.

In this paper, we approach the feature selection process from a classifier-agnostic perspective based on correlation among common feature selection algorithms to identify such patterns across three different datasets and network topologies. We make the following contributions:

- **A classifier-agnostic approach for feature selection**. We introduce an ensemble approach of filter methods to rank 65 features followed by a voting technique to select a subset of features. Our approach does not rely on, or make explicit assumptions about, the classifier that will utilize the selected features.
- **Evaluation of ensemble feature selection voting**. We evaluate our approach using three different datasets with respect to classifier performance and training time. We show that, across datasets and network topologies, removal of similar features has a trivial effect on accuracy.
- **An analysis of the feature selection efficacy across commonly-used machine learning classification techniques**. We show the classifier-agnostic nature of our approach by applying it to typical classifiers. We show that classifier performance converges between 30 to 50 features, depending on the dataset. An ensemble of filter feature selection methods with an aggressive feature elimination threshold achieved the best classifier performance while substantially reducing the training time.

The remainder of this paper is organized as follows. Section II gives an overview of approaches for feature selection. In Section III we provide the background regarding the data models for feature extraction and the datasets we use. Section IV presents our approach to construct a classifier-agnostic feature elimination method, and our experimental results are described in Section V. Finally, Section VI concludes.

II. Related Work

The closely related work involve approaches that reduce the feature space prior to training and tuning detection algorithms. Three primary approaches are used to eliminate features that are not relevant for classification: wrapper, embedded, and filter methods. We briefly review each of these methods before we discuss ensembles.

In **wrapper methods**, the classifier is *wrapped* in an algorithm that searches the feature space for a subset of features that yield the highest classifier performance based on optimization of a predictor function [22]. Wrapper methods include genetic algorithms with logistic regression [23], [24] and differential evolution with neural networks [25]. A key drawback of wrapper methods is that solving the optimization problem is time-consuming to the detriment of our goal of identifying a feature set for daily re-training of NIDS in production environments. These methods are not recommended for high dimensionality datasets due to their computational complexity [26].

Filter methods rank features and select the highly-ranked features for the classifier. Ranking is often based on statistical techniques applied to a feature to determine its correlation with the label or outcome. Common approaches include mutual information (MI) or information gain [27]–[29] and analysis of variance (ANOVA) [30]. Filter methods are computationally efficient depending on the time complexity of the filter used, but have low precision and may fail to find linear correlations between features [30].

Embedded methods seek to minimize the computation time required to reclassify the optimal subsets generated in wrapper methods by combining both filter and wrapper approaches in a two-stage process [31]. They embed feature selection as part of the training process without splitting the data into training and testing sets. Selvakumar and Muneeswaran use mutual information, a filter technique, prior to using a meta heuristic firefly algorithm as a wrapper [32]. Kasongo and Sun used Extreme Gradient Boosting (XGBoost) to select a subset features from the UNSW-NB15 dataset in 2020 [33]. Unfortunately, embedded methods still require an iterative convergence of an optimization step in the wrapper, thus the time complexity remains high.

Ensemble methods combine multiple filters or filters with embedded approaches. Krishnaveni et al. [34] introduced a univariate ensemble feature selection method using majority voting across three datasets. This ensemble included MI, gain-ratio, Chi-squared, symmetric uncertainty, and relief. The authors tested effectiveness on the accuracy, detection rate, and false positive rate of support vector machine, naïve Bayes, logistic regression, and decision tree. The reduction of build time and test time was not reported. Seijo-Parso et al. [35] presented an ensemble of filters and embedded methods tested using a support vector machine classifier and different thresholds

to determine performance of the feature selection methods across diverse datasets outside of the NIDS domain.

Comparing and aggregating the results of heterogeneous classifiers into a single signal is traditionally approached through voting or stacking. Malhotra and Sharma [36] conducted an empirical study to determine the threshold of features on the Apache Click dataset and found that only 33-50% of the features were necessary to yield reasonable results. Seijo-Pardo et al [37] proposed a programmatically chosen threshold based on a formula on various datasets and found that automatic means were faster than a numeric threshold cutoff after evaluating feature selection on SVM. They further showed that Fisher's discriminant ratio was effective at reducing the number of features without significantly affecting performance [38]. A thorough exploration of ensemble approaches can be found in [39] and [40].

Many of the aforementioned works rely on KDD99 and NSL-KDD datasets. These datasets do not have the latest attack techniques and thus these results may not be representative of new protocols or attack techniques. Due to the time constraints involved in daily retraining of NIDS data, we chose to pursue a filter-based approach. Our ensemble of filter methods, however, employs a *voting algorithm* in addition to statistical tests to determine which features are the most useful. We apply our approach across multiple datasets and reveal features that do not improve classifier performance across samples, network topologies, and captured attacks.

III. Background

In this section, we provide background information regarding the data models that can be used to extract features from network traffic (Section III-A) and our selection criteria for datasets as well as the ones we chose in this work (Section III-B).

A. Data Models

Supervised machine learning approaches for NIDS require structured and labeled data. Choosing a data model to capture a dataset such as raw packet captures or a (specific) network flow version incurs tradeoffs. Network flows can save on space and processing, but as the flows provide metadata about a network connection, feature visibility can be lost as the original traffic cannot be reconstructed [41]. Packet captures, or pcaps, retain this visibility and provide a richer feature domain, but are larger to store and take longer to process. Pcap data, as they are in raw form, must be distilled into usable machine learning features [13], [42] and labeled prior to training a supervised classifier. By contrast, network flows began as a proprietary standard and the features they provide have evolved over time [43]. What began as a 5-tuple data object providing IP addresses, port numbers, and protocol, evolved to 18 fields in version 5 [43]. As network flow protocols matured, the number of fields expanded to the 491 fields available in RFC 7011 [17].

Other packet capture distillation methods include CICFlowMeter [13], formerly known as ISCXFlowMeter, and Joy [14]. CICFlowMeter is an open source tool that generates bidirectional network flows from raw packet capture data. It distills 83 features into a CSV file and requires (manual) labeling after distillation. Joy [14], by contrast, performs feature extraction by distilling packet captures into JSON or IPFIX. The number of features extracted is dependent on user configuration.

B. Dataset Selection Criteria and NIDS Datasets

To select appropriate datasets, we used the following selection criteria. First, as netflows can irreversibly convert a pcap, we prioritized datasets in the packet capture format that were publicly available. Second, we chose multiple datasets to capture different types of attack traffic over a range of network topologies instead of focusing in a single domain like solely SSH attacks [44] or botnet traffic [45]. Further, the datasets only contained actual traffic and not traffic generated by an algorithm. Lastly, the dataset had to be labeled, or provide adequate documentation to manually label, to provide the ground truth. In the following we describe the three datasets we chose for this work using these criteria.

1) CTU-13: Czech Technical University (CTU) released a dataset in 2011 [45] comprised of thirteen network traffic captures focused on detecting botnet traffic totalling 1.9GB. The dataset includes an edited packet capture, a labeled network flow, and documentation regarding the capture timeline. The preprocessed netflow form is provided in addition to the raw data. The raw data was processed using CICFlowMeter and labeled according to the dataset documentation that included the source of malicious traffic by IP. Malicious traffic included click fraud, port scans, fast flux, and author-controlled malware. The network used to generate traffic consisted of virtualized computers running Windows XP SP2, what the botnet malware could run on at the time, on a Linux Debian host bridged into the university network. The final set contains both the traffic from the virtualized computers and the university router, though some of the traffic was removed due to privacy concerns.

2) CICIDS17: The Canadian Institute for Cybersecurity (CIC) released the CICIDS2017 dataset [13] which includes a variety of attacks including password brute forcing, a heartbleed exploit, botnet traffic, traffic floods resulting in denial of service and distributed denial of service, a web server SQL injection, cross site scripting, and an infiltration. These attacks are recorded on a diverse network consisting of Windows and Ubuntu hosts, a firewall, several switches, and using both Windows 8.1 and Kali as attacking nodes. With the array of attacks, Sharafaldin et al. [13] provided a more general dataset available in both raw packet capture and pre-

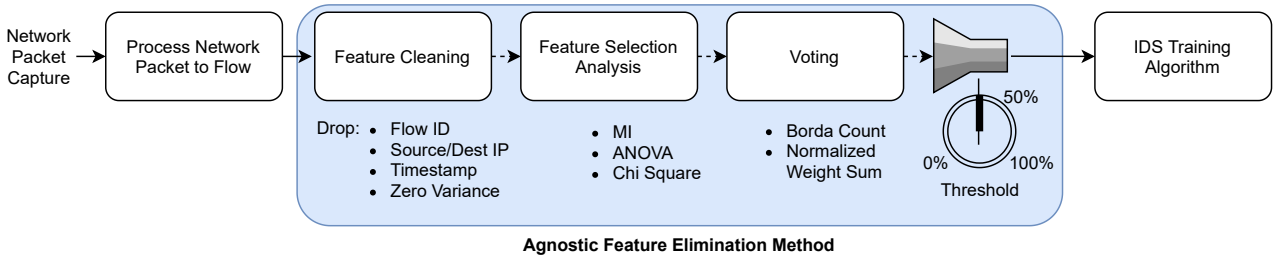


Fig. 1: Feature elimination pipeline between packet capture and classification.

processed by CICFlowMeter [46]. It covers five days of documented events resulting in 51.1GB of data. With the 80 features provided by CICFlowMeter, they utilized RandomForestRegressor, part of the scikit-learn library, to select prominent features per attack prior to training a standard set of classification algorithms.

3) CUPID: Like CICIDS17, the CUPID dataset[1] is a more general dataset providing a variety of attack types including webcrawling, reconnaissance techniques like ARP and nmap, web attacks like SQL injection, Layer 2 attacks, botnet traffic, and 10 pcaps generated from human operators. It was generated on an isolated test network consisting of Windows and Ubuntu hosts, a firewall, a switch, and Kali provided an attacking node. It is provided both in the pcap format and processed by CICFlowMeter comprising approximately 50 GB of data.

IV. Design

We now present our approach to construct a classifier-agnostic feature elimination method. We use a pipeline model, shown in Figure 1, to structure our approach and in the following explain each stage subsequent to the network packet capture. Note that this research does not take into account adversarial attacks, like data poisoning, instead prioritizing on patterns revealed by feature selection methods. Features that could easily be used to introduce noise and affect the accuracy of the classifier, like the timestamp or the IP address, were removed prior to training.

A. Process Network Packets to Flows

We use the netflow output from CICFlowMeter [13] as a common baseline for distilling features to feed our classifier-agnostic feature elimination method. We gathered the CTU-13 and CUPID datasets from their respective repositories in the packet capture format and processed the raw network data using CICFlowMeter [13]. CTU-13 provides adequate documentation to label the processed data. The CICIDS17 dataset is already available in the CICFlowMeter format. This approach facilitates analysis of the same 83 features over the three datasets.

[1]https://www.cupid.directory/

B. Feature Cleaning

Several features were eliminated prior to performing the feature selection process. The Flow ID, Source IP, Destination IP, and Timestamp were dropped. The Flow ID is a categorical feature that uniquely identifies each conversation. Using it would require one-hot encoding which would greatly increase the feature space. The IP address can be spoofed by an attacker and is also a categorical feature, so using the source or destination IP address could add noise to the dataset while also increasing the feature space. The timestamp was dropped as individual sensors, like IDSs or honeypots, could not be shown to be synchronized across devices nor datasets and obtaining simple temporal relationships requires additional heuristics [47].

Finally, features with zero variance in any dataset, as shown in Table I, were removed from all datasets. For the CICIDS17 dataset, redundant features were also removed (i.e., “Fwd Header Length”) and rows containing “Infinity” and “NaN” were dropped [48]. As a result of this feature cleaning, we are left with the same 65 features over three datasets. Features that could not be analyzed across all datasets were dropped. For example, if the active standard deviation was a zero-variance feature in CTU-13 and CUPID, it was also dropped from CICIDS17 as it was not possible to compare the feature across the datasets. While it is possible that removing specific features from a dataset may affect classifier performance, the features were removed for feature parity across datasets. Several features available natively from CICFlowMeter processing were not available in the processed CICIDS files and were removed from the other datasets to maintain feature parity including the source port, the active series (active minimum, active maximum, active mean, and active standard deviation), and several flag counts.

C. Feature Selection Analysis

After dropping the fixed features (known worthless and zero-variance), we use an ensemble of filter techniques—ANOVA, Chi-square, and MI—to determine dependencies and relationships between features. Prior to applying each filter we applied min-max normalization on each feature to rescale into $[0, 1]$.

TABLE I: Zero variance features in each dataset.

Feature Name	Dataset
Backward Avg Bulk Rate	CICIDS17
Backward Avg Packets	CICIDS17
Active Std	CTU13, CUPID
Active Mean	CTU13, CUPID
Active Max	CTU13, CUPID
Active Min	CTU13, CUPID
Forward URG Flag	CTU13, CUPID
URG Flag Count	CTU13, CUPID
Subflow Backward Packets	CTU13, CUPID
Forward Avg Bulk Rate	CICIDS17, CTU13, CUPID
Backward Avg Bytes/Bulk	CICIDS17, CTU13, CUPID
Forward Avg Bytes/Bulk	CICIDS17, CTU13, CUPID
Forward Avg Packets/Bulk	CICIDS17, CTU13, CUPID
Backward PSH Flag	CICIDS17, CTU13, CUPID
Backward URG Flag	CICIDS17, CTU13, CUPID

ANOVA statistically determines if two or more features are different by comparing the variance to estimate a linear degree of dependence. Features are ranked higher if they show a weak linear degree of dependence [49]. Chi-square evaluates the independence of two features by measuring dependence between each non-negative feature and class based on the class labels [50]. Mutual Information (MI) measures the statistical dependence between two random features. It is equal to zero if and only if the two features are independent and higher values correlate to dependence [27].

D. Voting

Since an ensemble of multiple filter methods has been shown to perform better than a single filter method [34], we explored two techniques—Borda count and min-max normalized weight summation—to combine the rankings of the three filter methods.

1) Borda Count: Borda count is a scoring rule used in a voting system that gives candidates points according to their voting rank position [51]. With a Borda count rule, each filter selects candidate features with the highest scores based on the statistical weights. $m - j$ points are given to a candidate that is ranked jth position, where m is the total number of features (65 in our case); the points awarded according to rank are $m - 1$, $m - 2$, ..., 0. The scores given to each feature are summed across the three filter methods and rank-ordered.

2) Normalized Weights Sum: In this approach, we normalized the filter method weights to [0,1] to distribute the weights uniformly. We sum the weights for each feature across the three filters and rank-order the features by their sums.

E. Thresholding

To determine the number of features to eliminate a threshold must be identified. Rather than using a metric to determine a quantitative threshold, we chose to use the notion of *diminishing returns* to identify a threshold rank above which adding more features does not improve classifier performance. Our approach is inspired by the theoretical approach of Mario et al. [52] but uses an empirical method applied during the training phase. Our approach finds the feature selection threshold by modeling how feature selection affects one classifier's performance, and then applying that threshold on future feature selection iterations with other classifiers. Classifier performance converges after a set number of features and the addition of more features does not improve performance but adds to training time.

F. NIDS Training Algorithms

We use the decision tree, random forest, and k-Nearest Neighbors (kNN) classifiers to determine the efficacy of our approach. The decision tree classifier attempts to predict a discrete target value from a set of observations. More specifically, we used the Classification And Regression Trees (CART) algorithm, which differs from traditional decision trees by constructing binary trees using the feature and threshold that yield the largest mutual information at each node. The random forest classifier consists of a set of individual decision trees. Each individual tree attempts to determine a class. The class with the most votes from the individual trees wins. The kNN classifier determines class membership based on a plurality vote of the input's neighbors.

The tree-based models use the raw values of the selected features, while the kNN classifier uses the $[0, 1]$ normalized features. We chose these algorithms because they are commonly used in NIDS [53], but we anticipate that other classifiers could be used similarly.

V. Experiments

In this section, we present the experiments and results obtained from the process described in Section IV.

A. Experimental Setup

Our experiments were implemented in Python 3.8.1 on a Microsoft Azure compute node, 4 cores, 28 GB RAM, 56GB disk.

B. Trivial Features

After performing our classifier-agnostic feature elimination technique, we mapped feature ranking as shown in Figure 2. This annotated heatmap reflects the ranking of each feature available across all three datasets after processing by CICFlowMeter.

The feature ranking in Figure 2 is indicated by the number and color where poor performers are darkly shaded with a numerical rank closer to 65 and excellent performers are indicated by a numerical rank closer to 0 and are lighter in shade. Notice that, when sorted by the median of the rankings, certain features that are poorly ranked across multiple analyses sink to the bottom of the diagram. Consider the `total_fwd_packets` and the `total_length_of_fwd_packet`, and their `bwd` counterparts, that are represented towards the bottom of Figure 2. These features indicate the number of packets

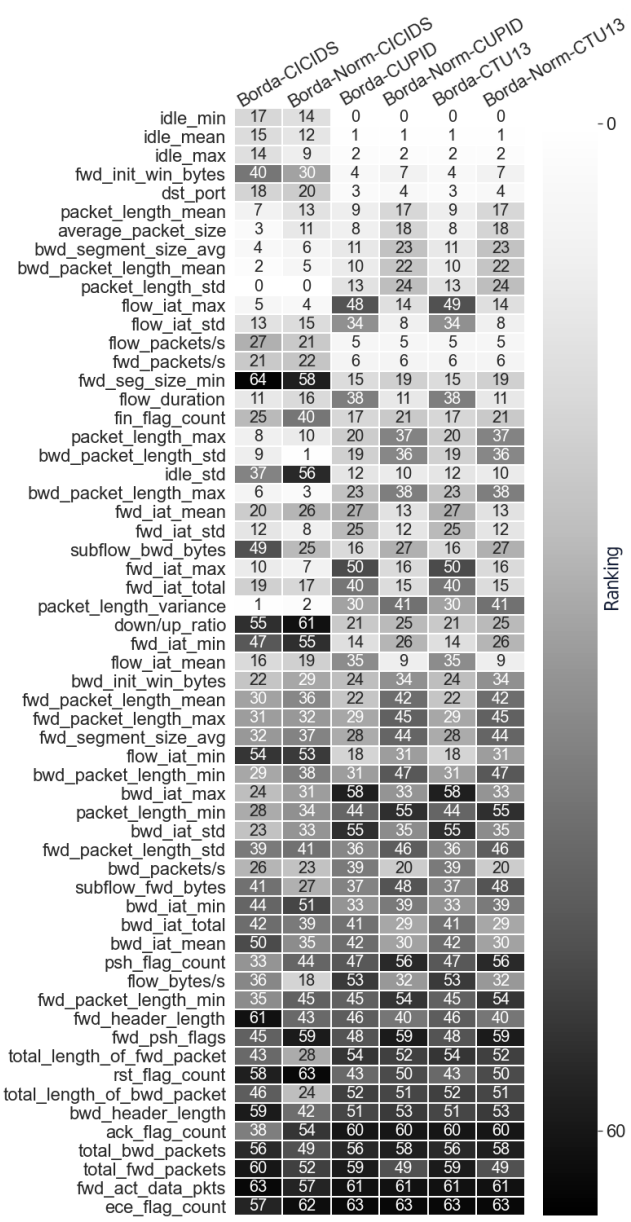


Fig. 2: Ranked features across datasets, sorted by median. A smaller value indicates a better ranking.

in a flow and the size of the packets to and from a host but perform poorly across the datasets measured with no ranking better than 24, where ranking is determined by each feature selection method. We theorize that the number of packets or their size is not highly correlated to either a benign or malicious event and is thus poorly rated. We suggest that poor performers may be due to fundamental network processes that do not correlate highly with a benign or malicious event and, despite different network topologies and datasets, that they may not be useful features for NIDS in general.

C. Classifier Performance Impact

Feature selection is intended to remove noise or bias for a more robust classifier, but overly aggressive feature elimination can greatly reduce classifier performance. We designed an experiment to evaluate the impact of our feature elimination method on classifier performance. In this experiment, we processed the three datasets—CICIDS17, CTU-13, and CUPID—through the pipeline and measured the classifier performance using accuracy, precision, recall, and F-1 score. Due to the imbalanced nature of all three NIDS datasets, cross validation is necessary to reduce possible bias in the classifiers; we used 5-fold cross-validation that was repeated two times.

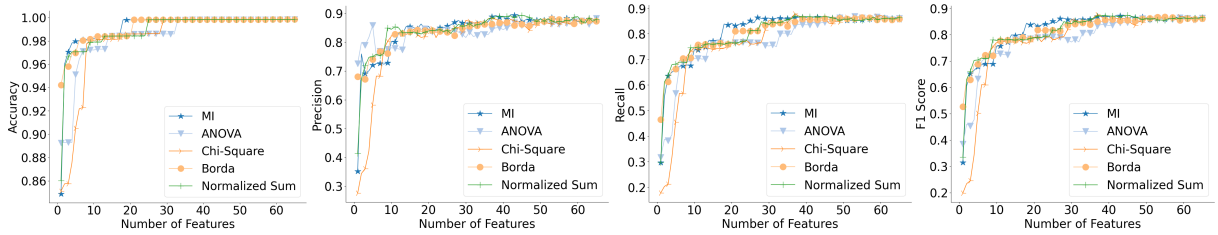
Figures 3, 4, and 5 show the classifier performance for the Decision Tree classifier as the number of features selected increases based on the individual results of the three filter methods—MI, ANOVA, and Chi-square—and the two ensemble approaches—Borda count and min-max normalization. Classifier performance in general converges between 30 and 50 features, depending on the dataset. Eliminating 10-20 features results in the same classifier performance as keeping all features in these datasets. In addition, no individual filter method performs best across the datasets, but the ensemble approach consistently achieves the best classifier performance with more aggressive feature elimination thresholds.

Table II shows the average re-training time for one shuffle (of the cross-validation) and classifier performance on the CTU-13 dataset for the three classifiers at the following selected points in the feature elimination thresholds: the full 65 features, the top 45 features as ranked in Figure 2, and the top 10 features determined by the Borda count and the min-max normalization approaches.

Tables III and IV show the same for the CICIDS17 and CUPID datasets. As seen in each table, the classifier accuracy trained with the top 10 features is trivially affected when compared to using the entire feature set, for all three datasets on some of the classifiers. Notably, kNN exhibited poor performance in precision and recall scores for the CICIDS17 dataset, while the remaining classifier performance results were reasonable. Furthermore, the re-training time decreased between 35% and 96%, depending on the dataset and the classification algorithm. Overall, it makes little sense to use the full feature set, as a careful selection of the feature pool as a subset is faster to train on, while minimally affecting performance.

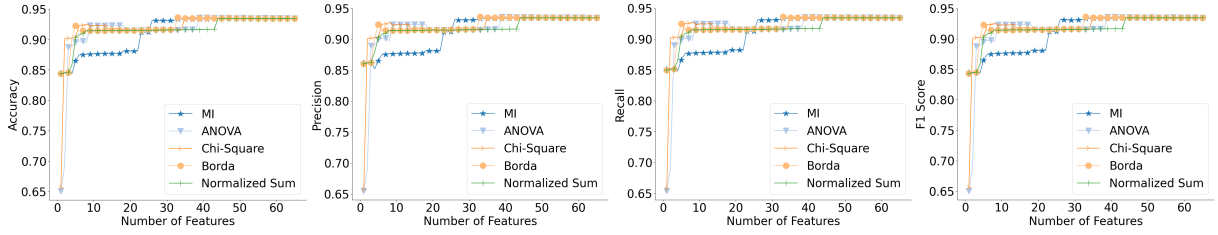
D. Re-training Time Impact

One of the goals of our approach for feature elimination is to improve re-training time, as there are fewer operations required to train the classifier, while maintaining performance. From Tables II to IV we observe that the re-training time decreases when the number of features is reduced. To determine the extent of the effect of feature selection on re-training time, we modeled re-training time across the datasets as the feature count decreases as



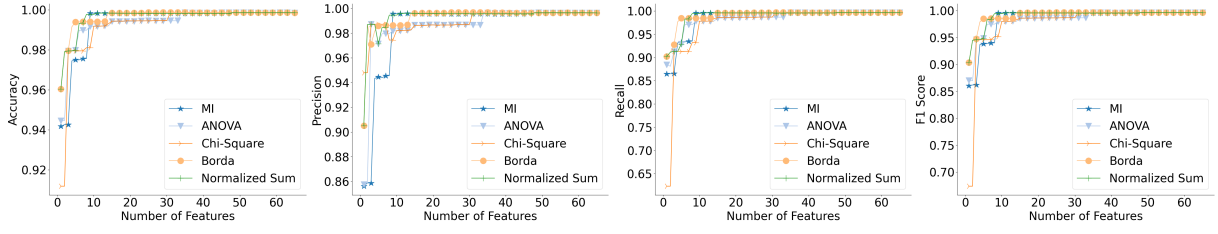
(a) Accuracy (b) Precision (c) Recall (d) F-1 Score

Fig. 3: Decision Tree classifier performance for CICIDS17 across feature selection approaches.



(a) Accuracy (b) Precision (c) Recall (d) F-1 Score

Fig. 4: Decision Tree classifier performance for CTU-13 across feature selection approaches.



(a) Accuracy (b) Precision (c) Recall (d) F-1 Score

Fig. 5: Decision Tree classifier performance for CUPID across feature selection approaches.

TABLE II: CTU-13 performance in terms of average re-training time (Time) in seconds, accuracy (Acc.), precision (Prec.), recall (Recl.), and F-1 score.

Classifier	**Full feature set**					**Top 45 features**					**Top 10 feat. (Borda)**					**Top 10 feat. (Normalized)**				
	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1
Random Forest	76.9	.937	.938	.939	.937	62.4	.906	.908	.908	.906	32.7	.928	.930	.931	.928	40.1	.930	.932	.933	.930
kNN	379.1	.886	.892	.890	.886	345.8	.854	.862	.858	.854	30.0	.912	.916	.915	.912	23.9	.914	.918	.917	.914
Decision Tree	9.9	.922	.922	.922	.922	7.2	.893	.893	.893	.893	1.5	.919	.919	.920	.919	2.1	.919	.918	.920	.919

TABLE III: CICIDS17 performance in terms of average re-training time (Time) in seconds, accuracy (Acc.), precision (Prec.), recall (Recl.), and F-1 score.

Classifier	**Full feature set**					**Top 45 features**					**Top 10 feat. (Borda)**					**Top 10 feat. (Normalized)**				
	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1
Random Forest	1265.7	.999	.998	.998	.998	1044.4	.999	.998	.998	.998	675.8	.973	.980	.934	.955	609.9	.974	.981	.937	.957
kNN	35463.9	.989	.990	.977	.983	32093.4	.987	.988	.970	.979	504.6	.975	.981	.940	.959	3302.9	.965	.971	.919	.942
Decision Tree	147.0	.999	.998	.998	.998	104.8	.999	.998	.998	.998	20.7	.973	.980	.934	.955	18.0	.974	.981	.936	.957

TABLE IV: CUPID performance in terms of average re-training time (Time) in seconds, accuracy (Acc.), precision (Prec.), recall (Recl.), and F-1 score.

Classifier	**Full feature set**					**Top 45 features**					**Top 10 feat. (Borda)**					**Top 10 feat. (Normalized)**				
	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1	Time	Acc.	Prec.	Recl.	F-1
Random Forest	425.4	.999	.997	.996	.997	418.4	.997	.994	.991	.992	321.9	.995	.989	.986	.987	180.5	.998	.997	.995	.996
kNN	8153.3	.992	.990	.973	.981	7950.7	.992	.989	.971	.980	254.1	.993	.990	.974	.982	2340.0	.993	.990	.976	.983
Decision Tree	33.5	.998	.996	.996	.996	21.9	.998	.996	.996	.996	9.0	.994	.986	.985	.985	5.1	.998	.995	.994	.995

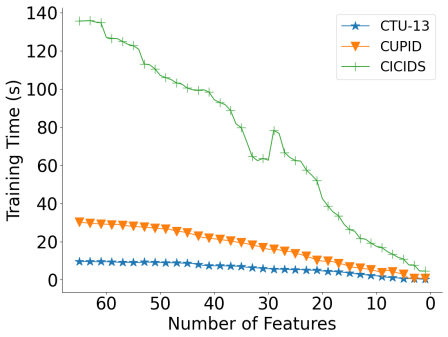


Fig. 6: Average re-training time (Decision Tree) vs Number of top K features using normalized feature selection.

shown in Figure 6 (using the MI feature selection, but the re-training time results generalize across feature selection approaches).

This figure reflects a *linear decrease* in average (per-fold) re-training time across the three datasets. Each dataset does not appear to follow the same trend for re-training time, indicating that the impact of feature elimination on re-training time varies by dataset. CICIDS17 has a much larger slope than CUPID or CTU-13, which we attribute to its use of multinomial labels as opposed to the binary labels used in CUPID and CTU-13. Note that the expected re-training time may differ depending on the method used, e.g., 5-fold cross-validation may have an expected total re-training time of 5 times the average re-training times shown, while simpler approaches such as an 80/20 validation would have re-training time close to the average (since we used a repeated 5-fold approach, each shuffle is roughly one 80/20 split).

E. Summary of Results

We evaluated our approach on three different network intrusion datasets (CICIDS17, CTU-13, and CUPID) using an ensemble of three filter techniques. Filter selection is intended to decrease re-training time while maintaining classifier performance—accuracy, precision, recall, and F-1 score. Across the datasets analyzed, we found that re-training time depends on multiple parameters of the dataset, e.g., size, number of labels, and complexity of flows, but that in all cases the re-training time increases linearly (with varying slopes) as more features are selected for classification.

After removing zero variance and redundant features, we ranked the remaining features based on the filter methods and two ensemble methods—Borda count and min-max normalization. We found that several features, like the number of packets sent or the packet length, perform poorly on average regardless of the dataset and the attacks captured in those datasets and may not be as useful because the time between network flows does not correlate strongly to either benign or malicious traffic. Features that generally rank strongly across datasets, like the destination port, may be useful as the destination port is indicative of the service being attacked at that port. As the number of available features to mine increases with protocol maturity there may be features that generally rank poorly as they are artifacts of the underlying network architecture but not indicative of traffic behavior.

We analyzed the effect on performance across a spectrum of feature elimination from using all features down to one feature with the lowest ranked feature removed at each step. We found that classifier performance plateaus between 30 to 50 features, depending on the dataset, while re-training time continues to grow linearly, which motivates a simple threshold approach to eliminate features that do not contribute to classifier performance past the plateau point.

VI. Conclusion

In this paper we have shown that a classifier-agnostic ensemble approach can be used to rank and eliminate features that contribute minimal value to classifier performance in order to reduce training time. With an increasing amount of data to select features and train and test on, even a modest reduction to training time can result in substantial cost savings and operational efficiency especially in production NIDS with daily packet capture data in the terabyte range. As this work solely focused on evaluating the feature space provided in the CICFlowMeter data model, future work could compare NIDS data across different data models (CICFlowMeter, Joy, IPFIX) or non-enterprise domains [54] to reveal relevant characteristics across models.

References

[1] J. Beale, *Snort 2.1 Intrusion Detection, Second Edition*. Syngress Publishing, 2004.
[2] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou, "Specification-based anomaly detection: a new approach for detecting network intrusions." ACM, 2002, pp. 265–274.
[3] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1484–1494, 2020.
[4] Y. Zhuang, E. Gessiou, S. Portzer, F. Fund, M. Muhammad, I. Beschastnikh, and J. Cappos, "Netcheck: Network diagnoses from blackbox traces," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI'14)*, 2014.
[5] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom, "Survey of automotive controller area network intrusion detection systems," *IEEE Design Test*, vol. 36, no. 6, pp. 48–55, 2019.
[6] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 3, pp. 186–205, 2000.
[7] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *Network and Distributed Systems Security Symposium*, 2019.
[8] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516–524, 2010.
[9] A. Countermeasures, "Real intelligence threat analytics (rita)," https://github.com/activecm/rita, 2020.
[10] Zeek, "Zeek network security monitor," https://github.com/zeek/zeek, 2020.

[11] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.

[12] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, 2012.

[13] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." 2018.

[14] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, 2017.

[15] J. Mao, Y. Hu, D. Jiang, T. Wei, and F. Shen, "Cbfs: A clustering-based feature selection mechanism for network anomaly detection," *IEEE Access*, vol. 8, pp. 116 216–116 225, 2020.

[16] J. Quittek, "Information model for ip flow information export," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 5102, January 2008. [Online]. Available: https://tools.ietf.org/html/rfc5102

[17] B. Claise, "Specification of the ip flow information export (ipfix) protocol for the exchange of flow information," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 7011, September 2013.

[18] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis." in *Ndss*, 2011, pp. 1–17.

[19] J. V. Hansen, P. B. Lowry, R. D. Meservy, and D. M. McDonald, "Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection," *Decision Support Systems*, vol. 43, no. 4, pp. 1362–1374, 2007.

[20] F. Jemili, M. Zaghdoud, and M. B. Ahmed, "A framework for an adaptive intrusion detection system using bayesian network," in *2007 IEEE Intelligence and Security Informatics*. IEEE, 2007.

[21] Z. Xu, G. Huang, K. Q. Weinberger, and A. X. Zheng, "Gradient boosted feature selection," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 522–531.

[22] H. Liu and H. Motoda, *Computational methods of feature selection*. CRC Press, 2007.

[23] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *computers & security*, vol. 70, pp. 255–277, 2017.

[24] ——, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Computer Networks*, vol. 172, p. 107183, 2020.

[25] F. H. Almasoudy, W. L. Al-Yaseen, and A. K. Idrees, "Differential evolution wrapper feature selection for intrusion detection system," *Procedia Computer Science*, vol. 167, pp. 1230–1239, 2020.

[26] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Computational Statistics & Data Analysis*, vol. 143, p. 106839, 2020.

[27] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184–1199, 2011.

[28] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE transactions on computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[29] S. M. Kasongo and Y. Sun, "A deep learning method with filter based feature engineering for wireless intrusion detection system," *IEEE Access*, vol. 7, pp. 38 597–38 607, 2019.

[30] M. A. Siddiqi and W. Pak, "Optimizing filter-based feature selection method flow for intrusion detection system," *Electronics*, vol. 9, no. 12, 2020.

[31] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[32] S. B and M. K, "Firefly algorithm based feature selection for network intrusion detection," *Computers & security*, vol. 81, pp. 148–155, 2019.

[33] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset," *Journal of Big Data*, vol. 7, no. 1, pp. 1–20, 2020.

[34] S. Krishnaveni, S. Sivamohan, S. Sridhar, and S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing," *Cluster Computing*, pp. 1–19, 2021.

[35] B. Seijo-Pardo, I. Porto-Díaz, V. Bolón-Canedo, and A. Alonso-Betanzos, "Ensemble feature selection: homogeneous and heterogeneous approaches," *Knowledge-Based Systems*, vol. 118, 2017.

[36] R. Malhotra and A. Sharma, "Threshold benchmarking for feature ranking techniques," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 2, pp. 1063–1070, 2021.

[37] B. Seijo-Pardo, V. Bolón-Canedo, and A. Alonso-Betanzos, "On developing an automatic threshold applied to feature selection ensembles," *Information Fusion*, vol. 45, pp. 227–245, 2019.

[38] ——, "Testing different ensemble configurations for feature selection," *Neural Processing Letters*, vol. 46, no. 3, 2017.

[39] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Information Fusion*, vol. 52, pp. 1–12, 2019.

[40] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Computer Science Review*, vol. 39, p. 100357, 2021.

[41] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[42] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM workshop on artificial intelligence and security*. ACM, 2016, pp. 35–46.

[43] O. Santos, *Cisco NetFlow Versions and Features*. Cisco Press, 2016.

[44] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "Ssh compromise detection using netflow/ipfix," *ACM SIGCOMM computer communication review*, vol. 44, no. 5, pp. 20–26, 2014.

[45] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[46] G. Draper-Gil., A. H. Lashkari., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC. SciTePress, 2016, pp. 407–414.

[47] T. C. Ristenpart, "Time stamp synchronization of distributed sensor logs: impossibility results and approximation algorithms," Ph.D. dissertation, University of California, Davis, 2005.

[48] G. Farahani, "Feature selection based on cross-correlation for the intrusion detection system," *Security and Communication Networks*, vol. 2020, 2020.

[49] A. Binbusayyis and T. Vaiyapuri, "Comprehensive analysis and recommendation of feature evaluation measures for intrusion detection," *Heliyon*, vol. 6, no. 7, 2020.

[50] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class svm," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 4, pp. 462–472, 2017.

[51] M. Diss, A. Tlidi, and E. Kamwa, "On some k-scoring rules for committee elections: agreement and condorcet principle," *Revue d'economie politique*, vol. 130, no. 5, pp. 699–725, 2020.

[52] M. Beraha, A. M. Metelli, M. Papini, A. Tirinzoni, and M. Restelli, "Feature selection via mutual information: New theoretical insights," *CoRR*, vol. abs/1907.07384, 2019.

[53] I. Aiyanyo, H. Samuel, and H. Lim, "A systematic review of defensive and offensive cybersecurity with machine learning," *Applied Sciences (Switzerland)*, vol. 10, no. 17, Sep. 2020.

[54] G. Bloom, B. Alsulami, E. Nwafor, and I. C. Bertolotti, "Design patterns for the industrial internet of things," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–10.