VOTING-BASED ENSEMBLE MODEL FOR NETWORK ANOMALY DETECTION

Tzu-Hsin Yang, Yu-Tai Lin, Chao-Lun Wu, Chih-Yu Wang

Research Center for InformationTechnology Innovation, Academia Sinica, Taipei, Taiwan, R.O.C.
{tzuhsin, ytlin1993, allenwu, cywang}@citi.sinica.edu.tw

ABSTRACT

Network anomaly detection (NAD) aims to capture potential abnormal behaviors by observing traffic data over a period of time. Machine learning has become a promising approach for the NAD problem. In this work, we propose a machine learning framework based on XGBoost and deep neural networks to classify normal traffic and anomalous traffic. Data-driven feature engineering and post-processing are further proposed to improve the performance of the models. The experiment results suggest the proposed model can achieve 94% for F1 measure in the macro average of five labels on real-world traffic data.

1. INTRODUCTION

The Internet and online services have become a critical part of daily life for a majority of human beings. The growing trend is still ongoing as the total number of internet users is projected to grow from 3.9 billion in 2018 to 5.3 billion by 2023 [1]. However, cyber attacks that target the Internet users and the network are also growing rapidly along with the growing demands on the Internet. Accordingly, network security and network anomaly detection (NAD) becomes a critical issue that needs to be tackle. Many researchers studied this problem based on a popular dataset NSL-KDD[2].

Traditionally, NAD resort to signature-based (or rule-based) methodologies to detect anomalies. Signature-based detection is based on predefined signatures representing known patterns of network anomalies and making rules to detect network traffic that contains the identical signatures [3]. However, such methodologies rely on manual approaches to detect anomalies patterns, which may not catch up with the rapid growth and evolution of cyber attacks recently.

In recent years, machine learning (ML) solutions such as XGBoost[4] and deep neural networks have become ubiquitous in various areas such as computer vision, natural language processing, etc. ML-based methods are potentially perfect choices for NAD. Several ML methods have been applied to cybersecurity intrusion detection [5]. For instance, Duffield et al.[6] exploited correlations between packet and flow-level information via an ML approach to associate packet-level alarms with a feature vector derived from flow records on the same traffic. Gaddam et al.[7] presented a method combining K-means clustering and ID3 decision tree on several anomaly detection tasks. Chen et al.[8] utilized XGBoost for DDOS detection in self defined networks. Shone et al.[9] constructed a deep autoencoder(NDAE) for unsupervised feature learning and using stacked NDAEs for intrusion detection. Ran et al.[10] provided a semi-supervised learning framework to learn from anomaly traffic. Yan et al.[11] designed an LSTM-based model with autoencoders for intrusion detection. Devan et al.[12] introduced an XGBoost-DNN model to detect cyber attacks; they used XGBoost for feature extractions and DNN for classification. Ma et al.[13] combined reinforcement learning with SMOTE algorithm to address the imbalanced class problem in the anomaly detection dataset. The previous works focus on model design. In contrast, we find that feature engineering and post-process are also critical for NAD.

As the participants of the ZYELL-NCTU NAD Challenge competition, we present an ensemble approach employing XGBoost and deep neural networks to detect network anomalies. According to the analysis on the received training dataset from the competition, we present the important insights and the corresponding feature engineering and post-processing techniques. The main contributions of this paper can be summarized as follows:

- We compose an ensemble approach employing XGBoost and deep neural networks to detect network anomalies.
- We compose a voting method in post-processing to improve prediction accuracy.
- We discover various patterns regarding different types of attack and craft additional features to represent them.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the attacks listed in the data and discuss some discoveries we found in the data. In Section 3, we elaborate our proposed method as well as feature engineering. Performance and experimental results are shown in Section 4. And finally, we conclude this paper in Section 5.

2. ATTACK ANALYSIS

The attacks listed in the training dataset, which are also the target of this competition, are - DDOS-Smurf, Probing-IP Sweep, Probing-Port Sweep, and Probing-Nmap Sweep. Since the goal is to detect attacks in network traffic, we will briefly introduce attacks mentioned above in the following sub-section for completeness. Furthermore, we will also present some attack analysis regarding the dataset in the following sub-section.

2.1. Attack Description

- **Probing-IP Sweep**: The attacker sends ICMP packets to various hosts to identify accessible hosts through IP network scanning. The goal of the attacker is to understand critical elements of the target network, such as vulnerable areas.
- **Probing-Port Sweep**: Similar to Probing-IP Sweep, Probing-Port Sweep also sends ICMP packets to victim hosts. However, Probing-Port Sweep scans various ports to identify the services in use on a specific port.
- **Probing-Nmap Sweep**: Probing-Nmap Sweep is using Nmap - a Linux tool to find active host IP in a network, discover open ports and services and detect vulnerabilities. The attacker automates this process through customized Nmap scripts.
- **DDOS-Smurf**: A DDOS-smurf attack sends a slew of ICMP Echo request packets in order to sabotage devices under the target network. Unlike similar attacks such as ping flood, DDOS-smurf targets the network's broadcast address that allows packets to be sent to all hosts in a particular network.

2.2. Attack Analysis

We analyze the training dataset to identify useful patterns related to the attacks' characteristics that could be exploited in feature engineering for the proposed ML model.

Firstly, We discovered that the Probing-IP sweep consists of an interesting pattern in the "dst IP" feature; most of them only differ from one another in the last octet but be the same in the first three octets. The behavior makes sense to us since the Probing-IP sweep scans hosts in the particular network. We also found that for DDOS-smurf attack, all the "dst IP" are ended with 255. That discovery also makes sense to us because DDOS-smurf is targeting broadcast addresses in order to enlarge the number of victims. Furthermore, we found that Probing-Nmap sweep's destination IP all starts with "172.24" in the first two octets. However, since the amount of data regarding the Probing-Nmap sweep is significantly smaller than other attacks in the dataset, further studies are needed to confirm whether this pattern would apply to general cases.

3. METHODOLOGY

Next, we will present our proposed ML method, which consists of feature engineering, ML model, and post-processing. Feature engineering aims to extract useful information beforehand based on the attack analysis. Post-processing aims to refine the prediction outcomes of multiple ML models to boost the prediction performance.

3.1. Feature Engineering

We first introduce how we process features before modeling. We used raw features of "duration," "in (bytes)," "out (bytes)," and all "cnt" features. Plus, we transformed some features in order to make them more meaningful for modeling. The details are as follows.

3.1.1. Encoding for Categorical Features

There are categorical features, "app" and "proto" in the feature set. Since they are not meaningful in their original format, we have to process them with one-hot encoders. After encoding, each dimension represents one specific type of application or protocol.

3.1.2. Time, IPs and Ports

When digging into data, we discovered that although time series of traffic data are significant for anomaly detection, the specific time is not important as both normal and anomaly traffic can happen at any time. Therefore, we don't use actual time as the input of the model. Instead, we use a time-series relationship in post-processing. We will illustrate how we process the output of the model in Section 3.3.

For IPs, we originally assumed that they are useful features because, as mentioned in Section 2.2, the attack contains patterns that relate to IP addresses. However, this assumption holds only if the collected traffic in both training and testing datasets belong to the same network area. To verify this assumption, we check the IP addresses co-exist in both the training and testing dataset. We firstly binarize all source and destination IPs and then use Principal Component Analysis [14] (PCA) to reduce dimensions. Then, we visualize the latent vectors to see if the IPs in the training set are similar to testing sets. It turned out that they are not similar, which suggests the assumption is invalid. Therefore, we remove features of IPs. Note that although the features of IPs are not included as model inputs, they are still useful in post-processing as we will illustrate later.

Regarding ports, although some applications or protocols correspond to a common port, ports can be a random number in many cases. Thus, in order to prevent features of ports from affecting model performance due to its randomness, we removed these features as well.

3.1.3. Additional Features

As mentioned in Section 2.2, we discovered each attack contains some interesting patterns. According to these patterns, we craft additional features indicating these patterns and found that some of these additional features could further boost the prediction performance in training sets. We list the most effective features in the following.

- **inner_src, inner_dst**: We discovered that all the destination IPs of Probing-Nmap start with "172.24". Thus, we think the features of Inner IPs may help to detect Probing-Nmap. We consider the IP as inner IP if it belongs to the IPv4 address ranges reserved for a private network.
- **dst_ip_end_with_255**: Since we discovered that the destination IPs of DDOS-smurf data are all ended with 255, we wondered whether adding a feature indicating the destination IP of data ended with 255 would improve performance in predicting DDOS-smurf. The experiment results in Section 4.2.5 show that the performance improves dramatically in predicting DDOS-smurf. However, since the broadcast address on different environment settings is not always ended 255, this feature may lead to overfitting.
- **is_sweep**: We discovered that, for Probing-Port sweep and Probing-IP sweep, the destination IP often differs only in the last octet from the consecutive data with the same label in the neighboring. Therefore, we calculate the difference of "src" and "dst" between each row. If "src" is the same as the previous row and "dst" is the same in the first three octets but different in the last octet. Then we assign the value of "is_sweep" in this case is 1. Otherwise, it is 0. We also add this feature to train the model. The experiment result shows this will improve the detection of Probing-IP sweep, Probing-Port sweep, and DDOS-smurf. For probing attacks, the improvement is reasonable. The improvement of DDOS-smurf comes from the fact that DDOS-smurf always attacks the same broadcast addresses for a while. Thus, it will not trigger the "is_sweep" feature. Nevertheless, the destination IP of sweeping attacks would not necessarily differ in the last octet. The attacker could also target multiple networks at the same time, and the attack record might vary a lot in destination IP. Thus, this feature may also lead to overfitting.

3.2. Ensemble method

3.2.1. XGBoost Classifier

XGBoost [4] is a scalable boosting framework, which is widely used in both industry and academia. We chose this model as our main framework. To tackle the considered multi-classification problem, we use softmax as the learning objective. Softmax normalizes the outputs and gives a probability of each category $\hat{y}_i$. Softmax is defined in (1), where z_i is the original outputs of the model.

$$\hat{y}_i = \frac{e^{z_i}}{\sum_k^C e^{z_k}} \tag{1}$$

The loss function of XGBoost is categorical cross-entropy loss shown in (2). C is the number of the category, which is 5 in our task. y_i is the ground truth of traffic. The value is 1 if traffic belongs to its corresponding category; otherwise, it is 0. $\hat{y}_i$ is the prediction probability of the proposed model.

$$Loss = -\sum_i^C y_i log(\hat{y}_i) \tag{2}$$

The XGBoost's learning parameters are shown below. The learning rate is 0.1. The maximum depth of a tree is 6. The minimum child weight is 1. L2 regularization is used on weights. The optimization algorithm follows the XGBoost optimization algorithm [4] and using early stop for the validation and the test. The early stop round is set to 10.

3.2.2. Deep Neural Networks

Recently, deep learning gradually becomes a popular technique widely used in many research fields such as signal processing, computer vision, and speech recognition. Deep learning-based models have several benefits for data mining. We designed a deep neural network to detect anomalous traffic. There are 7 layers (ReLU-Dropout*3+Softmax) in total in our proposed neural network model. The structure of the proposed method contains three fully connected ReLU[15] layers. Each layer has 2048 neurons. ReLU is effective for optimizing networks. ReLu activation function is defined in (3).

$$ReLU(\boldsymbol{x}) = max(0, \boldsymbol{w}^T \boldsymbol{x} + b) \tag{3}$$

After each ReLU layer, a dropout[16] layer with a dropout rate of 0.5 are added. Dropout is a regularization method for neural networks. The output layer has 5 softmax(1) neurons that predict the probability of each traffic category.

We use categorical cross entropy loss (2) for optimization, and the optimizer is Adam[17]. Note that XGBoost and neural networks do not share the same loss function. They are trained separately, and then their prediction results are combined. Details will be described in 3.2.3.

The learning and inference of neural networks follow the below procedure. Before training the proposed model, we normalized the preprocessed data that processed with feature engineering mentioned in Section 3.1. We use z-score to rescale these features into a distribution with mean value 0.5 and standard deviation value 0.5. In the training process, we train the model with early stop techniques, setting the patience count to 10. The batch size is 128. The optimizer's parameters are the learning rate $\alpha = 0.001$, the exponential decay rates $\beta_1 = 0.9$, $\beta_2 = 0.999$ and the small constant avoiding dividing by zero $\epsilon = 10^{-7}$. We utilized 1 GPU(GTX 1080

Ti) to accelerate the training and inference process. In the inference process, we chose the category with the highest probability to represent the traffic. And then, we also performed post-processing to increase the performance. The procedure and the network architecture are shown in Fig. 1

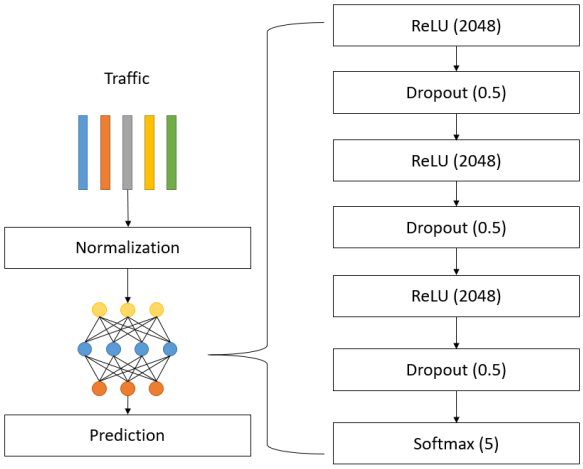


Fig. 1. Neural Network Workflow

3.2.3. Model Ensembling

Finally, we utilize ensemble methods to integrate prediction results from XGBoost and neural networks. We found out XGBoost and neural networks are capable of predicting different types of anomalous traffic. Specifically, we discovered that XGBoost made better predictions in most classes except Probing-IP sweep. In XGBoost prediction evaluation, the F1 score of Probing-IP sweep was 80%, while in neural networks prediction evaluation, the F1 score of Probing-IP sweep was 85%. Therefore, we use neural networks to predict Probing-IP sweep, and use XGBoost to predict other kinds of attacks and Normal to generate the final result.

3.3. Voting Method In Post-Processing

By studying the given dataset, we discovered that the consecutive data with the same source IP should have the same label in a certain time range, but our predicted result may exist in some cases that counter to this phenomenon. Thus, we compose a voting post-processing method to adjust the prediction result. For starters, the model predicts the probabilities of each label with given data. Then, the voting procedure would sum all output probabilities within the same consecutive data and assign all the involved cases with the highest candidate label. In this way, we expect those cases which do not have an apparent tendency can be assigned the correct label with their neighboring rows.

The rule that we define the data are consecutive is whether they are in the same hour. That is, if two traffic happened in the same hour and with the same source IP, we view them as consecutive data. Thus, they would get involved in the same voting session. For splitting time, we simply reference the"Time" feature in the data and separate time period for every hour. We have to note that we also discovered when we used different time periods to define consecutive data, the result would be slightly different. Specifically, we get a better F1 score for DDOS-smurf class when using "same minute" to define consecutive data. We argue that it is because the traffic of DDOS-smurf with the same source IP doesn't last as long as the traffic of Probing sweep, so the performance of it doesn't benefit from a longer time range. Therefore, when using "same hour" to define consecutive data, very few DDOS-smurf data get involved. Therefore, we compose a combined voting method that uses "same minute" setting for DDOS-smurf, and uses "same hour" setting for all the others.

Besides the fixed time splitting, we further propose a dynamic time splitting method by defining two traffic in the same group if their source IPs are the same and the time difference between two traffic are less than one hour. The result shows the improvement in predicting testing data. Please refer to Section 4.3.1 for details.

4. EXPERIMENTS

In this section, we will introduce the dataset and how we split the dataset into training and validation datasets to conduct our experiments in sub-section 4.1. Then, we will present our experimental results and discussion in sub-section 4.2. Finally, we will introduce the model explainer in sub-section 4.2.2.

4.1. Dataset

The data are time-series network traffic records captured by ZYELL's firewall. Each network traffic record is a network connection session and labeled as either normal or an attack type.

By analyzing the training dataset, we discovered that the training dataset collected from different dates distribute considerably different. For example, the distribution on 12/03 for Normal, Probing-IP sweep, Probing-Port sweep, Probing-Nmap, DDOS-smurf are 91.1%, 6.4%, 2.4%, 0%, 0% respectively, while the distribution on 12/10 for Normal, Probing-IP sweep, Probing-Port sweep, Probing-Nmap, DDOS-smurf are 97.9%, 1.8%, 0.2%, 0%, 0% respectively. It shows the labels of data are imbalanced in different datasets. Also, we discover that data labeled Probing-Nmap sweep are too few for training models. Due to the label imbalance problem, we need to split the data into training and validation sets carefully instead of using one data to predict the other one directly.

On the other hand, since most occurrences have more than one connection, each connection usually relates to its previous and subsequent cases. Therefore, when generating training and validation sets, we should not shuffle the raw dataset

directly. Otherwise, the model may peek at traffic data which also appears in the validation set. Then, the validation result is not convincing anymore. In this training stage, we take the dataset on 12/03, and partial dataset on 12/10 as the training set, and take the dataset on 12/16 and the partial dataset on 12/10 as the validation set. We carefully split the data to make sure they have similar label distribution as well as not having data within the same occurrence.

Moreover, we discovered that the data labeled Normal consist up to 95% of training data. This severe data imbalance would make the model overfit to Normal data and will always return Normal in prediction. Consequently, we downsample some Normal data and keep Normal data being 1.3 times the size of data labeled Probing-IP sweep.

4.2. Experiment Result

4.2.1. Confusion Matrix

We show the confusion matrix of the predictions made by XGBoost in Table 1. We found that DDOS-smurf cases and Probing-IP sweep cases are tended to be predicted as Normal, while Probing-Port sweep cases are tended to be predicted as Probing-IP sweep.

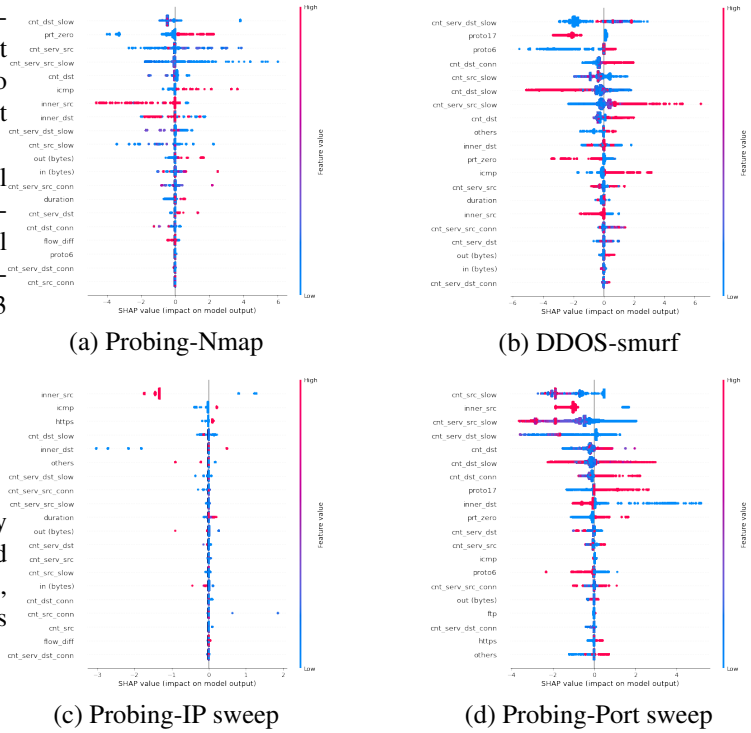
Table 1. Confusion Matrix

	DDOS	Normal	IP sweep	Nmap sweep	Port sweep
DDOS	102	931	0	0	0
Normal	3068	3161023	0	0	285
IP sweep	2	11001	48537	117	242
Nmap	0	0	0	95	14
Port sweep	1	0	130	0	8513

4.2.2. XGBoost Evaluation - SHAP Values

In Fig. 2, we plot the SHapley Additive exPlanations (SHAP) values summary [18]. We should note that the SHAP values summary plots only show how models learn from given features without considering the effect of the proposed voting procedure. However, we can still find some insights that explain the prediction of the models.

First, according to the plot of DDOS-smurf, "prt_zero," which is defined as 1 when both source and destination port are zero, and "ICMP", which is defined as 1 when the case's "app" is "ICMP," are two significant features for predictions as we can see the red points and blue points are separated clearly. In addition, according to the plot of Probing-IP sweep, "cnt_dst_slow" is an important feature. For the same source IP address, when the number of unique destination IP addresses inside the network is bigger, it is more likely to be a Probing-IP sweep behavior. For the Probing-Nmap sweep, since there are no significant features of predicting the Probing-Nmap sweep, the points are gathered at zero lines. Finally, for Probing-Port sweep, "proto6" and "proto17" are significant features, which help to distinguish from Probing-Port sweep and Probing-IP sweep.



(a) Probing-Nmap (b) DDOS-smurf

(c) Probing-IP sweep (d) Probing-Port sweep

Fig. 2. SHAP values of attack class

4.2.3. Performance Across Models

We compare performance across models, i.e., XGBoost, neural network (NN), and our proposed ensemble method in this sub-section. Note that in this experiment, we use the voting method with "minute" as the time range. Also, note that the evaluation matrices used in the experiments, i.e., F beta score and Criteria are provided by the contest committee.

Table 2 shows the prediction performance for each model applies to validation dataset. We can see that the ensemble method outperforms XGBoost and NN method. Since the ensemble method chooses the better prediction results from XGBoost or NN base on their prediction performance in each label, the performance result shown in Table 2 is plausible. Table 3 shows the performance of our proposed ensemble method across labels. We can see that performance on either DDOS-smurf or Probing-Nmap sweep is poor. We think the reason is that the data number of DDOS-smurf and Probing-Nmap sweep, especially for Probing-Nmap sweep, are far fewer than other labels. Also, by digging into DDOS-smurf, we found that the model is easily confused DDOS-smurf for Normal data, i.e., the model will easily miss-predict DDOS-smurf data to Normal. This phenomenon also happens in XGBoost and NN in our experiment.

Table 2. Performance Across Models

	XGBoost	NN	ensemble
F beta score	0.6666	0.6750	0.6958
Criteria	0.6408	0.6512	0.6631

Table 3. Ensemble Method Performance Across Labels

	Precision	Recall	F1 score	data counts
DDOS	0.0344	0.1007	0.0513	1033
Normal	0.9995	0.9931	0.9963	3164376
IP sweep	0.7592	0.9814	0.8561	59899
Nmap sweep	0.2290	0.8716	0.3626	109
Port sweep	0.9018	0.9337	0.9173	8644

However, we found that the NN's performance is not stable after conducting several tests. It is possible that NN is sensitive to data downsampling. Thus, NN is easier to overfit training data, even if there are only small differences of downsampling data between each test. We still select XGBoost as our main model for the following experiments.

4.2.4. Time range in voting method

In this experiment, we present how the voting method boosts the performance in prediction as well as how time range settings affect different types of attacks. Table 4 shows the results of applying our proposed voting method. Note that column "voting combined" indicates the final time range combination across different kinds of attacks we apply that mentioned in Section 3.3, and column "voting dynamic" indicates the refined time splitting approach.

Table 4. F1 score with different time range

	original	voting min	voting hour	voting combined	voting dynamic
DDOS	0.0546	0.0562	0.0115	0.0560	0.0134
Normal	0.9960	0.9961	0.9976	0.9976	0.9976
IP sweep	0.8081	0.8067	0.8952	0.8942	0.8942
Nmap sweep	0.3626	0.6620	0.4204	0.5901	0.6985
Port sweep	0.9173	0.9415	0.9549	0.9620	0.9582
Criteria	0.6408	0.6554	0.6595	0.6688	0.6857

As we can see in Table 4, the performance is significantly boosted in all kinds of attacks, especially for Probing-IP sweep, Probing-Port sweep, and Probing-Nmap sweep. Specifically, the XGBoost model originally was easily confusing Probing-IP sweep for Normal and Probing-Port sweep, and easily confusing Probing-Nmap sweep for Probing-IP sweep and Probing-Port sweep. However, the major portion of confusing data is the small portion of the consecutive data with the same source IP. Thus, by applying the voting post-processing procedure, the incorrect result would be correct and consequently improve the prediction accuracy. As for time range choosing, the result shows that Probing-IP sweep and Probing-Port sweep perform better in prediction when choosing hour as time range. Typically, Probing-IP sweep and Probing-Port sweep continue for a long time, usually exceed 1 min; thus, using "hour" as time range in Probing-IP and Probing-Port sweep would make the voting fairer. For "voting dynamic", although it performs worse in predicting DDOS-smurf, this could be resolved by adding "dst_ip_end_with_255" feature. On the other hand, "voting dynamic" perform better in predicting Probing-Nmap sweep.

4.2.5. Performance with additional feature

In this sub-section, we present the experiment results in feature engineering. The base model we use in this experiment is XGBoost with voting post-processing.

Table 5 shows the F1 score for each label across different combination of feature engineering mentioned in Section 3.1. The result shows that performance further boosts in predicting DDOS-smurf, IP sweep, and Port sweep by adding additional features "is_sweep" and "dst_ip_end_with_255". For adding "is_sweep", the experiment result shows that the performance improves not only in predicting IP sweep and Port sweep but also in predicting Nmap sweep. We believe that the model could learn the sweeping behavior referring to this feature and, thus, could separate sweeping kinds of attacks from others well in prediction. For adding "dst_ip_end_with_255", the model could filter out DDOS-smurf attacks by this feature since it could strongly represent the DDOS-smurf behavior.

Table 5. F1 score for different additional feature

	voting	voting + is_sweep	voting + is_sweep + dst_end_with_ip_255
DDOS	0.0546	0.1677	0.9918
Normal	0.9960	0.9973	0.9973
IP sweep	0.8081	0.8755	0.8670
Nmap sweep	0.3626	0.6352	0.6352
Port sweep	0.9173	0.9201	0.9191
Criteria	0.6408	0.6873	0.8083

4.3. Testing Results

4.3.1. Submission Results

Table 6 shows the 7 testing results we get after submitting our models. In the previous experiments, the results show that our proposed ensemble method and additional features improve the performance.

For 3rd submission (ensemble method with voting_min), we think the reason why it performs worse than 1st one (XGBoost + voting_min) is that the ensemble method always uses NN's prediction result in the Probing-IP sweep label. If NN is overfitting to a training set that performs badly in the testing set, it will also bring the ensemble method into the abyss

Table 6. Testing Results

	Combination	Criteria
1st submit	xgboost + voting_min	0.5476
2nd submit	xgboost + voting_min + is_255	0.5876
3rd submit	ensemble + voting_min	0.4982
4th submit	xgboost + voting_min + is_sweep	0.5003
5th submit	xgboost + voting_combined	0.6060
6th submit	xgboost + voting_combined + is_255	0.6469
7th submit	ensemble + voting_combined + is_255	0.5887
8th submit	xgboost + voting_dynamic + is_255	0.6494

of overfitting. To address this issue, we add dropout layers to avoid overfitting. However, although, the 7th submission shows that the setback reduces comparing to 6th submission than 3rd submission comparing to 1st submission, the ensemble method still can't avoid overfitting issue.

For 2nd and 6th submissions, we discover that the testing result improves comparing to 1st and 5th submissions. We believe that the DDOS-smurf attack in the testing data is also likely to target the broadcast address end with 255. However, we should notice that, as mentioned in Section 3.1, the broadcast addresses are not always ended with 255. As for 4th submissions, recalling that as Section 3.1 have said, the sweeping behavior may randomly choose a different network to attack.

For 8th submissions, we refined our voting method as mentioned in Section 3.3. The result shows that the criteria improve around 0.003 in comparison with the 6th submission. We believe that the new time splitting paradigm in the voting method performs better in grouping consecutive data.

4.3.2. Grand Challenge Testing Result

We compared two testing results, that is, our 1st submission and 5th submission, respectively. The only difference between the 1st and 5th submissions is the time range we choose in the voting method. We found that 98.5% of traffic is predicted as the same label between 1st and 5th submission. We show the number of connections that were predicted differently and their predicted labels in Table 7. Since for the 5th submission, we only changed the time range in post-process, we can see how the time range affects the result by comparing the two predictions. In Table 7, the 5th predictions gave one group of connections Normal label instead of Probing-Port sweep label and gave another group of connections Probing-Port sweep label instead of Normal. We believe this difference boosts the performance. Plus, the detection of Probing-IP sweep also improves significantly. Thus, we can say that our assumption in the previous paragraph is correct.

Table 7. Difference between first and fifth submissions

1st submit	5th submit	Count
Probing-Port sweep	Normal	104296
Normal	Probing-Port sweep	66180
Probing-Port sweep	Probing-IP sweep	28351
Probing-IP sweep	Probing-Port sweep	2568
Probing-IP sweep	Normal	636
Normal	DDOS-smurf	435
Normal	Probing-IP sweep	14

5. CONCLUSIONS

In sum, we propose an ensemble model with feature engineering and voting-based post-processing to identify various types of cyber attacks. We first analyze the characteristics of these attacks and develop feature engineering techniques based on our observations. Then, we design an ensemble model including XGBoost and deep neural networks for detecting attacks. We also use the voting method in post-processing which boosts the performance. To explain our model, we use SHAP values to evaluate each feature's contribution. The experiment results show the superior of the proposed models, while the testing results suggest overfitting is still a challenge in the NAD problem.

6. REFERENCES

[1] Cisco, "Cisco annual internet report, 2018 - 2023 white paper," *Cisco Public Information*, 2020.

[2] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[3] Claude Turner, Rolston Jeremiah, Dwight Richards, and Anthony Joseph, "A rule status monitoring algorithm for rule-based intrusion detection and prevention systems," *Procedia Computer Science*, vol. 95, pp. 361–368, 2016.

[4] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd international conference on knowledge discovery and data mining (SIGKDD)*, 2016, pp. 785–794.

[5] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[6] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *IEEE INFOCOM*, 2009, pp. 424–432.

[7] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-means+id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 345–354, 2007.

[8] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "Xgboost classifier for ddos attack detection and analysis in sdn-based cloud," in *IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2018, pp. 251–256.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[10] J. Ran, Y. Ji, and B. Tang, "A semi-supervised learning approach to ieee 802.11 network anomaly detection," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[11] Y. Yan, L. Qi, J. Wang, Y. Lin, and L. Chen, "A network intrusion detection method based on stacked autoencoder and lstm," in *IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[12] P. Devan and N. Khare, "An efficient xgboost–dnn-based classification model for network intrusion detection system," *Neural Computing and Applications*, vol. 32, 2020.

[13] X. Ma and W. Shi, "Aesmote: Adversarial reinforcement learning with smote for anomaly detection," *IEEE Transactions on Network Science and Engineering*, 2020.

[14] Svante Wold, Kim Esbensen, and Paul Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, 1987, Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[15] Vinod Nair and Geoffrey E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*, 2010, p. 807–814.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[17] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[18] Scott M. Lundberg and Su-In Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017, p. 4768–4777.