Studying machine learning techniques for intrusion detection systems

Quang-Vinh Dang

Studying machine learning techniques for intrusion detection systems

Quang-Vinh Dang[0000−0002−3877−8024]

Data Innovation Lab, Industrial University of Ho Chi Minh city, Ho Chi Minh city, Vietnam
dangquangvinh@iuh.edu.vn

Abstract. Intrusion detection systems (IDSs) have been studied widely in the computer security community for a long time. The recent development of machine learning techniques has boosted the performance of the intrusion detection systems significantly. However, most modern machine learning and deep learning algorithms are exhaustive of labeled data that requires a lot of time and effort to collect. Furthermore, it might be late until all the data is collected to train the model.
In this study, we first perform a comprehensive survey of existing studies on using machine learning for IDSs. Hence we present two approaches to detect the network attacks. We present that by using a tree-based ensemble learning with feature engineering we can outperform state-of-the-art results in the field. We also present a new approach in selecting training data for IDSs hence by using a small subset of training data combined with some weak classification algorithms we can improve the performance of the detector while maintaining the low running cost.

Keywords: Intrusion Detection System · Machine learning · Classification.

1 Introduction

Intrusion Detection System (IDS) is an important component in a computer security system [33]. The task of a IDS is to detect malicious activities in the computer system, hence enable quick reaction to the attacks. The task of IDSs can be traced back to 1980 in the work of Anderson [4] where the author proposed a "computer security threat monitoring and surveillance system" that can detect the following behaviors: Use outside of normal time, Abnormal frequency of use, Abnormal volume of data reference, and Abnormal patterns of reference to programs or data.

Given the vital role of computer systems in our daily life today, the IDS attracts more and more attention from both academia and industry in recent years. According to IDC Cybersecurity Spending Guide 2019, worldwide spending on security might exceeds 100 billions of US Dollar in 2019 [13]. According to Kaspersky Security Lab, while the number of denial services which keeps growing since 2013 [43], the attackers now also focus on more advanced attacks [25].

On the other hand, more and more advanced attack techniques have been invented over time, started with lone-wolf attacks in the early of 1990s to corporate- and government-supported today. As the consequence, the traditional methods based on signature and experts rule are no longer sufficient [34].

Over years, IDSs that based on machine learning techniques have been developed [33]. These techniques utilize the cutting-edge development in machine learning research community in addition to the huge dataset gathered in cybersecurity research. We review state-of-the-art results in Section 2.

The main problem of the existing methods based on machine learning techniques is that they are trained on one single huge dataset, hence prone to be over-fitting when new types of attack are presented. These techniques usually required a huge data that might not be always ready but might take time to be collected. On the other hand, many widely used machine learning algorithms in IDSs do not support online-learning, i.e. they need to be trained from beginning if new data arrives. These techniques usually require a high computational power. Moreover, even the researchers have designed comprehensive machine learning systems for IDSs, th.e predictive performance is not yet to be perfect, hence there exists a room for the improvement.

In this paper, we survey the existing machine learning techniques for IDSs and perform a details benchmark on the real dataset collected by [41] in Section 2. Then we present our two approaches in using machine learning for IDSs: either we use a complicated machine learning algorithm with a *blind* attack, or use a light and fast algorithm with some training sampling techniques. We show that the supervised-based approaches on IDSs are mostly solved by heavy machine learning algorithms such as *xgboost* [12]. We propose a technique to sub-sampling the training data set hence we can improve the predictive performance of *weak* algorithms such as Naive Bayes classifier.

2 Literature Review

In general, there are two main approaches in using machine learning for IDSs: supervised learning approaches in which a model tries to distinguish between benign and malicious traffic flow, and unsupervised learning approaches, mostly anomaly-detection based methods, in which a model tries to distinguish between benign and other traffic flow. We do not discuss in this paper the traditional signature-based intrusion detection approach [24] such as the well-known system Snort [37].

2.1 Supervised Learning Approaches

Most of off-the-shelf supervised learning approaches have been used in IDS studies [22]. In short, the main task of a supervised classifier is to predict a network flow is a benign or malicious attack, given a set of network flows with labels (benign / attack) before for training.

Traditional Machine Learning Approaches Traditional machine learning approaches such as SVM, logistic regression or decision tree have been used for a long time in IDSs.

The authors in [28, 26, 3, 42] proposed to use decision-tree for the classification problem. A decision-tree is a classification algorithm that aiming to build a sequential rules in term of *if-else* by minimizing the loss function in separating the groups. The most common loss function using in building a decision tree is Gini [3]. While the authors of [28, 26, 3] simply build a decision tree on a given dataset, [42] combined the decision-tree technique with Genetic Algorithm [30] for feature generation to achieve a better predictive performance.

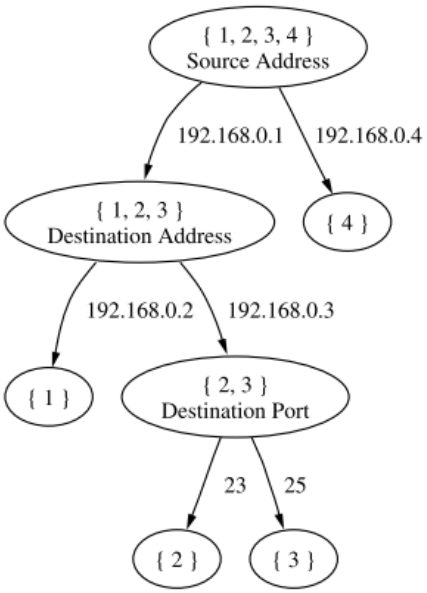

Fig. 1. A decision tree built by Kruegel et al. [26]

Another popular traditional classification algorithm is used widely in IDS is Support Vector Machine (SVM) [35, 5]. The core principle of SVM is to find an optimized hyper-plane that can classify between two classes. In practice, the most difficult task in building SVM models is to find a good kernel.

Combination of several techniques has been proposed in early of 2000s [42]. By the development of ensemble learning techniques, random forest has been chosen as the most widely used technique [36]. The idea of random-forest is to build multiple decision trees then combine the individual prediction of each tree. Random forest has been proven to outperform other classification techniques in well-defined dataset [16]. The random forest algorithm is visualized in Figure 2.

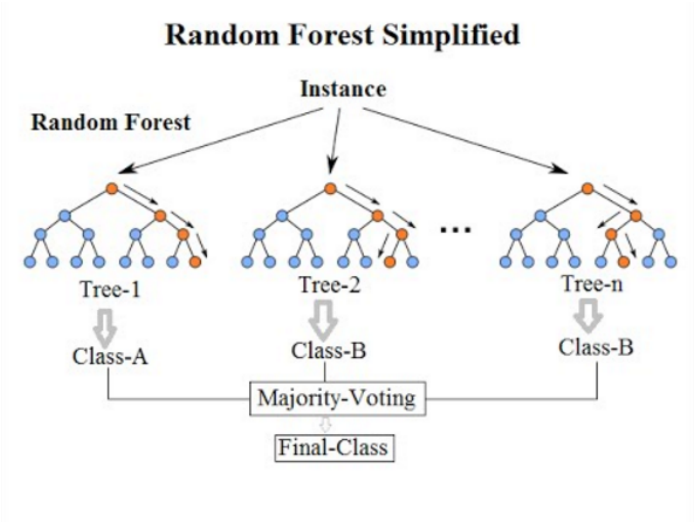

Fig. 2. Random Forest

Deep Learning Based Approaches In recent years, deep learning research achieves several important breakthrough results [16–18, 15]. It is no surprise that researchers utilize the power of deep learning methods in IDS. However, mostly feed-forward neural networks are being used in IDSs. A multi-layer feed-forward neural network is visualized in Figure 3.

The authors in [19] proposed to use a multi-layer feed-forward neural networks instead of traditional methods such as logistic regression or SVM in IDSs. The authors of [45] considered not only features generated by the system log but also using word embedding techniques to learn from system calls.

Besides a simple multi-layer feed-forward neural networks, several research studies have utilized more complicated networks such as ConvNet or Recurrent Neural Networks [46]. The approach of [46] is presented in Figure 4. The main problem of that approach and other similar approaches based on spatial and temporal relations of network flows is that these information must be kept within the system.

2.2 Anomaly-Detection Approaches

Different from supervised-based approaches, anomaly-detection approaches do not assume a labeled dataset but try to learn what is *benign* flow and detect any un-normal network flows [14]. Hence, the main assumption of an anomaly-detection algorithm is that the *benign* flows is the majority class in the data.

Anomaly-detection algorithms can be divided into two sub-groups: statistical-based algorithms and machine-learning based algorithms. Statistical algorithms rely on statistical attributes of the data such as Z-score or Mahalanobis distance

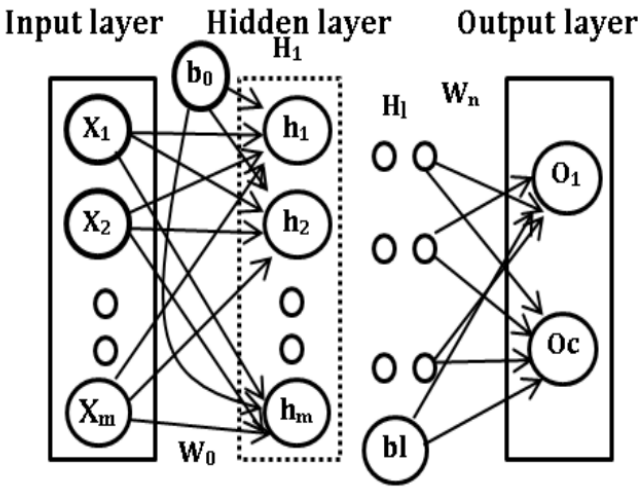

Fig. 3. The structure of a deep neural network.

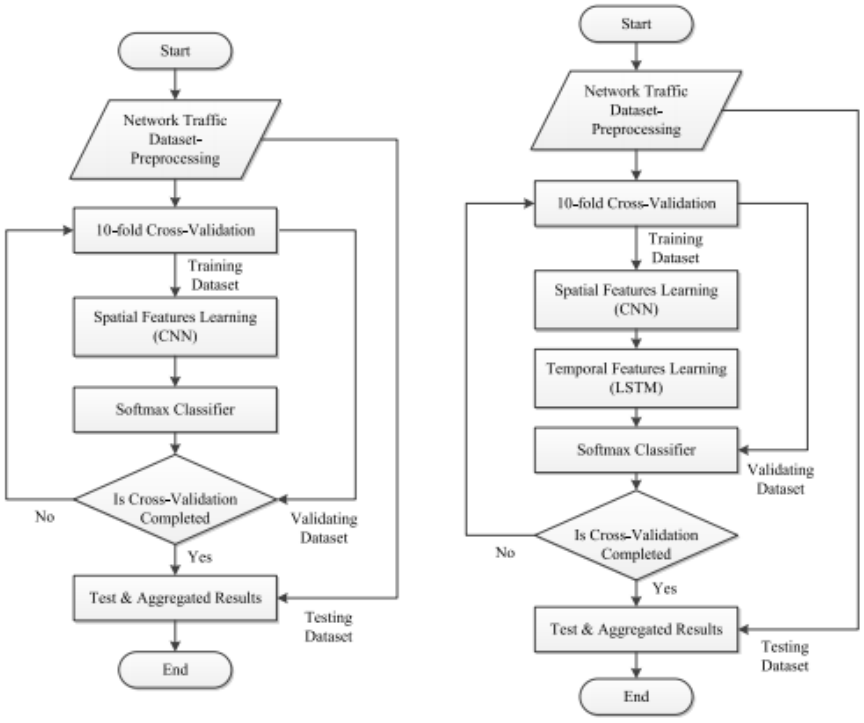

Fig. 4. HAST-IDS system [46]

to detect anomaly. We summary several popular distance functions in Table 1 based on the survey of [9] and in Table 2 based on the study of [8].

Name	*Measure,* $S_i(x_i, y_i)$	*Name*	*Measure,* $S_i(x_i, y_i)$
Euclidean	$\sqrt{\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2}$	Weighted Euclidean	$\sqrt{\sum_{i=1}^{d} \alpha_i \lvert x_i - y_i \rvert^2}$
Squared Euclidean	$\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2$	Squared-chord	$\sum_{i=1}^{d} (\sqrt{x_i} - \sqrt{y_i})^2$
Squared X^2	$\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i}$	City block	$\sum_{i=1}^{d} \lvert x_i - y_i \rvert$
Minkowski	$\sqrt[p]{\sum_{i=1}^{d} \lvert x_i - y_i \rvert^p}$	Chebyshev	$\max_i \lvert x_i - y_i \rvert$
Canberra	$\frac{\sum_{i=1}^{d} \lvert x_i - y_i \rvert}{x_i + y_i}$	Cosine	$\frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}}$
Jaccard	$\frac{\sum_{i=1}^{d} x_i y_i}{\sum_{i=1}^{d} x_i^2 + \sum_{i=1}^{d} y_i^2 - \sum_{i=1}^{d} x_i y_i}$	Bhattacharyya	$-\ln \sum_{i=1}^{d} \sqrt{(x_i y_i)}$
Pearson	$\sum_{i=1}^{d} (x_i - y_i)^2$	Divergence	$2 \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{(x_i + y_i)^2}$
Mahalanobis	$\sqrt{(x-y)^t \sum^{-1} (x-y)}$	-	-

Table 1. Similarity functions for numerical data [9]

w_k, k=1…d	*Measure,* $S_k(x_k, y_k)$	w_k, k=1…d	*Measure* $S_k(x_k, y_k)$
$\frac{1}{2}$	$Overlap = \begin{cases} 1 & \text{if } x_k = y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$	$Eskin = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{n_k^2}{n_k^2+2} & \text{otherwise} \end{cases}$
$\frac{1}{d}$	$IOF = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{1}{1+\log f_k(x_k) \times \log f_k(y_k)} & \text{otherwise} \end{cases}$	$\frac{1}{d}$	$OF = \begin{cases} 1 & \text{if } x_k = y_k \\ \frac{1}{1+\log \frac{N}{f_k(x_k)} \times \log \frac{N}{f_k(y_k)}} & \text{otherwise} \end{cases}$

Table 2. Similarity functions for categorical data [8]

On the other hand, machine-learning anomaly-detection algorithms rely on machine learning techniques to learn the pattern of the normal data, then detect if a new instance belongs to the class of normal data or not. Eskin et al. [20] defined One-class SVM (OCSVM) based on the idea of SVM in classification, but instead of separating two classes the OCSVM algorithm tries to separate the data from its origin. Liu et al. [29] based on the idea of random forest to build the *isolation forest* algorithm that determine the outlier score by the depth of the tree that required to classify a single data point. The Isolation Forest is visualized in Figure 5. We can see that it is difficult to classify a normal point from the dataset because it is quite similar from other points, while it is relatively easy to classify an anomaly point.

A good amount of research in outlier detection can be found in [1]. There exists several survey of anomaly detection in particular for IDSs, such as [10] and [6]. In general, anomaly-detection algorithms might have a better generalization compared to supervised algorithms because they do not require labeled data and can be adapted for the dynamic nature of the attacks. However, the main concern of anomaly-detection algorithms is the high running time that make them not applicable for real-time systems [44].

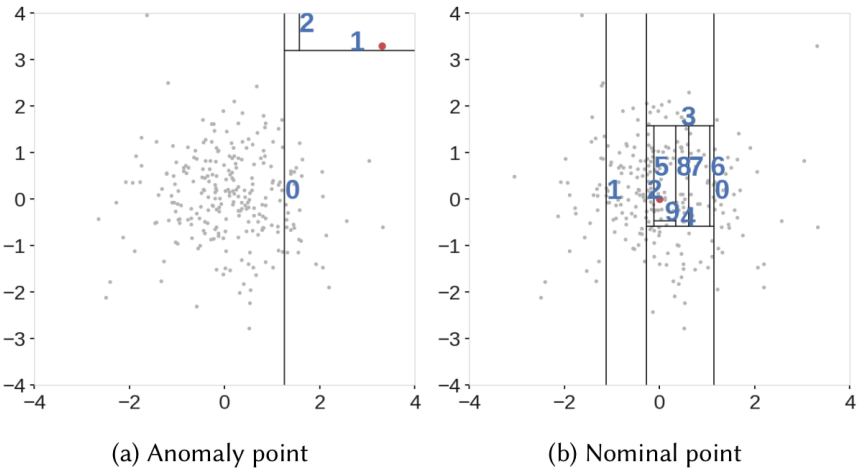

Fig. 5. Isolation Forest tries to isolate an instance from others. Harder an instance to be isolated, more *normal* it is.

3 Our Approaches

In this study we mainly rely on two learning approaches: Naive Bayes for a lightweight solution and xgboost for heavy computational power resource.

3.1 Naive Bayes learning

Naive Bayes is probably one of the most simple learning algorithm we might have today. Naive Bayes relies on Bayes's theorem as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

Naive Bayes assume the independence between features, hence it can be learnt very fast as:

$$P(f_1, f_2, ..f_n|BENIGN) = \prod_{1}^{n} P(f_i|BENIGN) \tag{2}$$

then assign the prediction as the highest probability class.

3.2 xgboost

xgboost stands for eXtreme Gradient Boosting [12] is a recent ensemble technique that has been introduced as one member of gradient boosting family [21]. The main idea of gradient boosting techniques is to build multiple sequential learners

(will be referred as *weak learners*) where the next learner tries to correct the error made by the previous one.

Hence, the final boosting model of xgboost will be:

$$F_m(x) = F_0(x) + \sum_{i=1}^{M} \rho_i h_i(x, a_i) \tag{3}$$

$F_0(x)$ is the initial model: $F_0(x) = argmin_\rho \sum_{i=1}^{N} l(y_i, \rho)$, i.e. $F_0(x)$ can be initialized as constant. ρ_i is the weight value of the model number i, h_i is the base model (decision tree) at the i^{th} iteration.

xgboost also introduced a new regularization term. The objective function is usually defined as:

$$obj(\theta) = l(\theta) + \Omega(\theta) \tag{4}$$

with l is the loss function (e.g. squared-error $\sum (y_i - \hat{y}_i)^2$) and Ω is regularization term.

In the original version of gbm, no regularization term is used in the objective function. In contrast, xgb explicitly added regularization term, and an author of xgb in fact called their model as "regularized gradient boosting"[1].

On the other hand, gbm introduced *shrinkage* as regularization techniques that are being used also by xgb. To use shrinkage, while updating the model in each iteration:

$$F_m(X) = F_{m-1}(X) + \nu.\rho_m h(x; a_m), 0 < \nu \leq 1 \tag{5}$$

The idea is, instead of taking the *full* step toward the steepest-descent direction, we take only a *part* of the step determining by the value of ν.

xgboost is fully integrated to distributed data management systems such as Apache Spark that allows us to perform xgboost in very huge dataset. To the best of our knowledge there is not yet any attempt to use xgboost in IDSs.

4 Experimental Results

4.1 Datasets

Probably the KDD'99 dataset is the most popular dataset has been used for evaluation of the IDSs[2] [23, 2, 38]. However, the dataset has been criticized by several research studies, such as the details analysis done by [32, 31]. Mainly, the dataset is questioned for the generation process and the inconsistency in the data description [40]. Furthermore, the dataset is created on a Solaris-based system that has very little in common with popular operating systems today (Mac OS, MS Windows, Ubuntu) [2].

[1] https://www.quora.com/What-is-the-difference-between-the-R-gbm-gradient-boosting-machine-and-xgboost-extreme-gradient-boosting/answer/Tianqi-Chen-1

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

In this study, we decided to use the attacking datasets gathered by [41] and made public by Canadian Institute for Cybersecurity[3]. We will refer to the dataset as ISCXIDS2012 dataset as suggested by the creators. The dataset is collected in total seven days in 2010 with different types of attacks.

We describe the details number of each type of network flows in following:

- Monday
  - 529918 BENIGN flows.
- Tuesday
  - 432074 BENIGN flows.
  - 7938 FTP-Patator[4].
  - 5897 SSH-Patator.
- Wednesday
  - 440031 BENIGN.
  - 231073 DoS Hulk.
  - 10293 DoS GoldenEye.
  - 5796 DoS slowloris.
  - 5499 DoS Slowhttptest.
  - 11 Heartbleed.
- Thursday morning
  - 168186 BENIGN.
  - 1507 Web Attack Brute Force.
  - 652 Web Attack XSS.
  - 21 Web Attack SQL Injection.
- Thursday afternoon
  - 288566 BENIGN.
  - 36 Infiltration.
- Friday morning
  - 189067 BENIGN.
  - 1966 Bot.
- Friday afternoon 1
  - 97718 BENIGN.
  - 128027 DDos.
- Friday afternoon 2
  - 127537 BENIGN.
  - 158930 PortScan.

Each network flow is provided with a list of 78 features. A snapshot of the dataset is presented in Figure 6. We might notice that the distribution of the classes is extremely imbalanced: while the majority of the network flows are BENIGN, the DDos flows occupied a huge share of the network but there is very few Heartbleed flows for instance.

[3] https://www.unb.ca/cic/datasets/ids.html
[4] https://tools.kali.org/password-attacks/patator

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	...	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	49188	4	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
1	49188	1	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
2	49188	1	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
3	49188	1	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
4	49486	3	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN

Fig. 6. A snapshot of the dataset.

4.2 Metrics

As the dataset is extremely imbalanced, the common used metrics such as accuracy do not necessarily reflect the true performance of an algorithm [11]. We will use ROC AUC (Receiver Operating Characteristic - Area Under the Curve) or AUC for short to evaluate our algorithm. The ROC is plotted against True Positive Rate and False Positive Rate as visualized in Figure 7. The AUC will be calculated as the area of the grey area and will take the value between 0 and 1, higher is better.

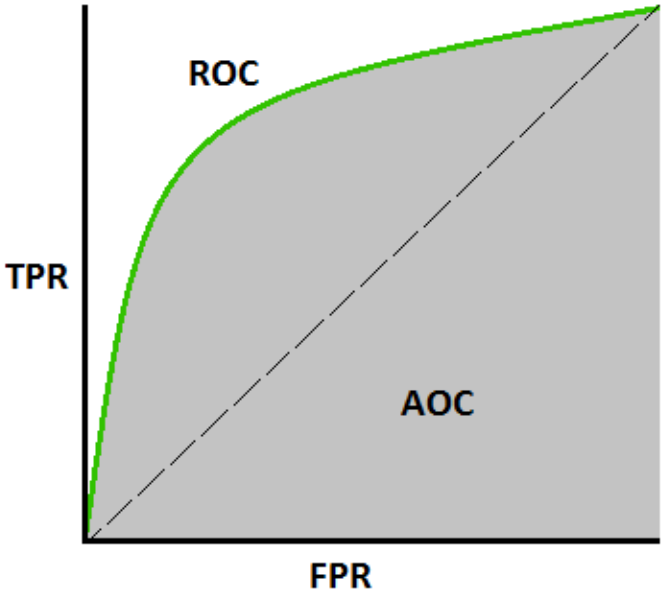

Fig. 7. ROC AUC measures the Area Under the Curve between the plot line created by True Positive Rate and False Positive Rate of a model on a dataset.

AUC can penalize the *majority* prediction, i.e. to predict everything as the major class. In this case the accuracy will be very high, but the AUC will be 0.

We note that AUC is available for binary classification only.

4.3 Data Division

We divide the original dataset into three parts: training set, validation set and test set by the ratio 60%, 20% and 20% respectively. We keep the validation set and test set as fixed set and perform sampling on training set only in case.

4.4 Results

4.5 Full training data

Binary Classification In binary settings, we convert all attacks into one single class "ATTACK" in comparison to "BENIGN". The focus here is to detect network attack without detecting the exact attack type. We display the predictive performance of the model in Table 3. We notice the significant out-performance of ensembles learning (random forest, xgboost) against other models.

Algorithm	AUC
Naive Bayes	0.5
Logistic Regression	0.55
SVM (linear kernel)	0.62
OCSVM (RBF kernel)	0.57
Random Forest	0.92
xgboost	0.9992

Table 3. AUC of different classifier in binary setting.

We display the feature importance of the xgboost model in Figure 8. We notice that the most importance feature in term of *gain*, i.e. "the relative contribution of the corresponding feature to the model calculated by taking each features contribution for each tree in the model" is *Destination Port*. It might be surprised at the first sight, but in fact most of network attacking flow (68.6%) has the destination port of 80 but the ratio in normal flow is only 10.4%.

Multi-class Classification In multi-class settings, we treat each attack type as different. The confusion matrix of the xgboost model is displayed in Figure 9. The accuracy of the model is 99.8%.

4.6 Sub-training data

As the predictive performance of xgboost is almost perfect if we use the entire training dataset, we suspect that we do not need all the training data for the model. We analyze how much data we actually need for xgboost to perform well.

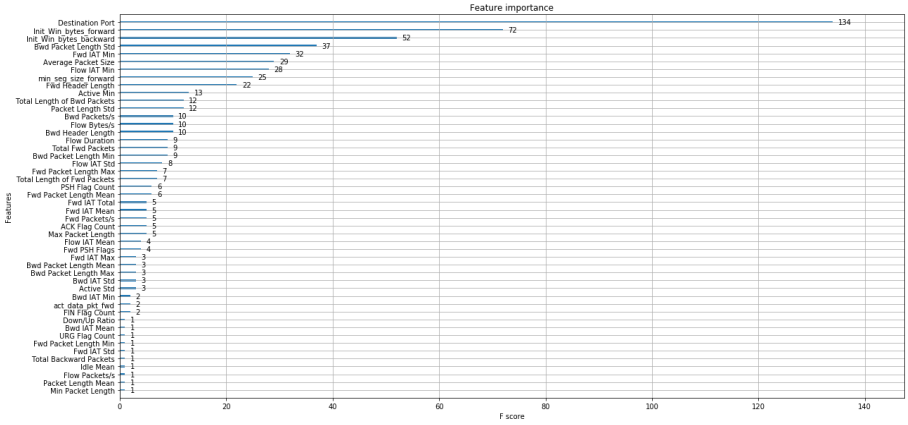

Fig. 8. Feature Importance in binary classification settings

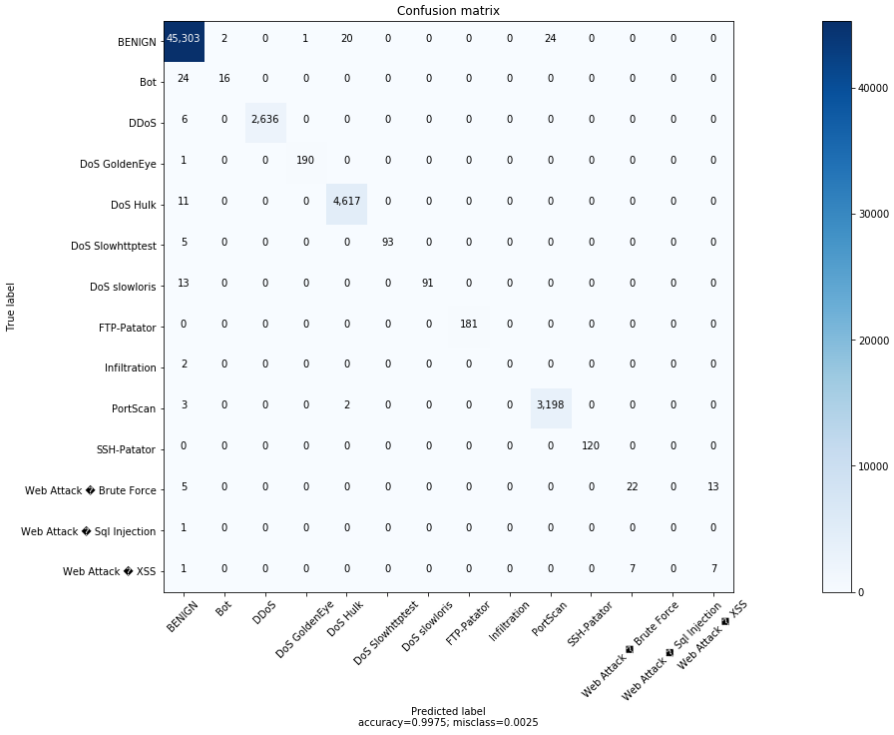

Fig. 9. Confusion matrix of multi-classifcation result

We realize that the xgboost can perform very well in CICIDS'12 dataset with a little training data. In fact, starting from 1% of training data the xgboost model can achieve the AUC of 98.7% already.

Sample selection As the xgboost model can perform well with quite little of training data, our next question is can we find a way to select this set of data for other *weaker* algorithms, such as Naive Bayse. We recall that, by using the entire training dataset the Naive Bayes achieved the AUC of approximate 0.5, i.e. the Naive Bayes does not learn anything and in fact has the same performance with random guessing.

The core reason for Naive Bayes to not perform well is that the distribution of training data is far from the distribution of testing data, while the Naive Bayes model lacks the flexibility to learn that [47]. Our plan is to find the most close sub-set of training data compared to the testing set. However, the distance metrics in high-dimension data might become unstable [7]. Due to that reason, we first apply dimension reduction by Principal Component Analysis (PCA) before calculating the distance.

By selecting top 10% of training data in term of the distance to the testing set we can improve the AUC of Naive Bayes to 0.86 which is significant higher than the *naive* usage on the entire dataset.

4.7 Discussion

CICIDS'2012 is a comprehensive dataset to evaluate the performance of IDSs in an effort to replace the outdated KDD'99 dataset. CICIDS'12 has attracted a lot of attention from the research community. The predictive performance has been increased over years. In this study we showed that the problem defined in CICIDS'2012 is almost solved by using gradient boosting technique, implies that we need a new and more complicated dataset as the ground for evaluation of IDSs.

On the other hand, there is not much effort in studying and using simple techniques like Naive Bayes recently due to the attention in deep learning techniques. Nevertheless we show that Naive Bayes still can achieve quite a high predictive performance given a proper training dataset, meaning that there is still a room for improvement of these algorithms.

5 Conclusions & Future Works

In this study, we showed that the the problem of intrusion detection with CICIDS'2012 can be solved mostly perfect with recent ensemble machine learning technique like xgboost, even with a small training dataset. We also show that by using sample selection method based on PCA algorithm we can improve the predictive performance of a simple learning algorithm like Naive Bayes.

In the future research we will study different methods for sample selections, particularly using uncertainty estimation techniques [27, 39].

References

1. Aggarwal, C.C.: Outlier analysis. Springer, 2 edn. (2017)
2. Ahmed, M., Mahmood, A.N., Hu, J.: A survey of network anomaly detection techniques. J. Network and Computer Applications **60**, 19–31 (2016)
3. Amor, N.B., Benferhat, S., Elouedi, Z.: Naive bayes vs decision trees in intrusion detection systems. In: SAC. pp. 420–424. ACM (2004)
4. Anderson, J.P.: Computer security threat monitoring and surveillance, james p. Anderson Co., Fort Washington, PA (1980)
5. Bhamare, D., Salman, T., Samaka, M., Erbad, A., Jain, R.: Feasibility of supervised machine learning for cloud security. CoRR **abs/1810.09878** (2018)
6. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: Methods, systems and tools. IEEE Communications Surveys and Tutorials **16**(1), 303–336 (2014)
7. Blum, A., Hopcroft, J., Kannan, R.: Foundations of data science. Vorabversion eines Lehrbuchs (2016)
8. Boriah, S., Chandola, V., Kumar, V.: Similarity measures for categorical data: A comparative evaluation. In: SDM. pp. 243–254. SIAM (2008)
9. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. International J. Mathematical Models and Methods in Applied Science **1**(2), 1 (2007)
10. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009)
11. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: Data mining and knowledge discovery handbook, pp. 875–886. Springer (2009)
12. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: KDD. pp. 785–794. ACM (2016)
13. Corporation, I.D.: Worldwide semiannual security spending guide (March 2019)
14. Dang, Q.: Outlier detection on network flow analysis. CoRR **abs/1808.02024** (2018)
15. Dang, Q.: Trust assessment in large-scale collaborative systems. (Évaluation de la confiance dans la collaboration à large échelle). Ph.D. thesis, University of Lorraine, Nancy, France (2018)
16. Dang, Q., Ignat, C.: Measuring quality of collaboratively edited documents: The case of wikipedia. In: CIC. pp. 266–275. IEEE Computer Society (2016)
17. Dang, Q., Ignat, C.: An end-to-end learning solution for assessing the quality of wikipedia articles. In: OpenSym. pp. 4:1–4:10. ACM (2017)
18. Dang, Q., Ignat, C.: Link-sign prediction in dynamic signed directed networks. In: CIC. pp. 36–45. IEEE Computer Society (2018)
19. Diro, A.A., Chilamkurti, N.: Distributed attack detection scheme using deep learning approach for internet of things. Future Generation Comp. Syst. **82**, 761–768 (2018)
20. Eskin, E.: Anomaly detection over noisy data using learned probability distributions. In: ICML. pp. 255–262. Morgan Kaufmann (2000)
21. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
22. He, Z., Zhang, T., Lee, R.B.: Machine learning based ddos attack detection from source side in cloud. In: CSCloud. pp. 114–120. IEEE Computer Society (2017)
23. Horng, S., Su, M., Chen, Y., Kao, T., Chen, R., Lai, J., Perkasa, C.D.: A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Syst. Appl. **38**(1), 306–313 (2011)

24. Jyothsna, V., Prasad, V.R., Prasad, K.M.: A review of anomaly based intrusion detection systems. International Journal of Computer Applications **28**(7), 26–35 (2011)
25. Kaspersky: The Kaspersky Lab DDoS Q4 Report (2019)
26. Krügel, C., Toth, T.: Using decision trees to improve signature-based intrusion detection. In: RAID. Lecture Notes in Computer Science, vol. 2820, pp. 173–191. Springer (2003)
27. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: NIPS. pp. 6402–6413 (2017)
28. Li, X., Ye, N.: Decision tree classifiers for computer intrusion detection. Journal of Parallel and Distributed Computing Practices **4**(2), 179–190 (2001)
29. Liu, F.T., Ting, K.M., Zhou, Z.: Isolation forest. In: ICDM. pp. 413–422. IEEE Computer Society (2008)
30. Lu, W., Traore, I.: Detecting new forms of network intrusion using genetic programming. Computational intelligence **20**(3), 475–494 (2004)
31. Mahoney, M.V., Chan, P.K.: An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In: International Workshop on Recent Advances in Intrusion Detection. pp. 220–237. Springer (2003)
32. McHugh, J.: Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Transactions on Information and System Security (TISSEC) **3**(4), 262–294 (2000)
33. Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., Payne, B.D.: Evaluating computer intrusion detection systems: A survey of common practices. ACM Comput. Surv. **48**(1), 12:1–12:41 (2015)
34. Radford, B.J., Apolonio, L.M., Trias, A.J., Simpson, J.A.: Network traffic anomaly detection using recurrent neural networks. CoRR **abs/1803.10769** (2018)
35. Reddy, R.R., Ramadevi, Y., Sunitha, K.V.N.: Effective discriminant function for intrusion detection using SVM. In: ICACCI. pp. 1148–1153. IEEE (2016)
36. Resende, P.A.A., Drummond, A.C.: A survey of random forest based methods for intrusion detection systems. ACM Comput. Surv. **51**(3), 48:1–48:36 (2018)
37. Roesch, M., et al.: Snort: Lightweight intrusion detection for networks. In: Lisa. vol. 99, pp. 229–238 (1999)
38. Sallay, H., Bourouis, S.: Intrusion detection alert management for high-speed networks: current researches and applications. Security and Communication Networks **8**(18), 4362–4372 (2015)
39. Segù, M., Loquercio, A., Scaramuzza, D.: A general framework for uncertainty estimation in deep learning. CoRR **abs/1907.06890** (2019)
40. Shafi, K., Abbass, H.A.: Evaluation of an adaptive genetic-based signature extraction system for network intrusion detection. Pattern Analysis and Applications **16**(4), 549–566 (2013)
41. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security **31**(3), 357–374 (2012)
42. Stein, G., Chen, B., Wu, A.S., Hua, K.A.: Decision tree classifier for network intrusion detection with ga-based feature selection. In: ACM Southeast Regional Conference (2). pp. 136–141. ACM (2005)
43. Symantec: Internet security threat report (2014)
44. Tiwari, A.: Real-time intrusion detection system using computational intelligence and neural network: Review, analysis and anticipated solution of machine learning.

In: ICITAM. Advances in Intelligent Systems and Computing, vol. 699, pp. 153–161. Springer (2017)
45. Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S.: Deep learning approach for intelligent intrusion detection system. IEEE Access **7**, 41525–41550 (2019)
46. Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M.: HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access **6**, 1792–1806 (2018)
47. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine learning **23**(1), 69–101 (1996)