2021

Identifying Mislabeling in Machine Learning Based Intrusion Detection System

Bofan Li
Illinois State University, 3093070121il@gmail.com

IDENTIFYING MISLABELING IN MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM

BOFAN LI

38 Pages

Machine learning has shown strong potential in improving the performance of an Intrusion Detection Systems (IDS). In a machine learning based IDS, the problem is commonly formulated as a supervised classification, in which various training datasets are used to train a selected model to learn how various network features are related to different types (i.e., benign traffic or a type of network attack) of network traffic. Each training dataset usually includes a large amount of data samples, and each data sample contains many network features and their associated type of traffic called label. Most recent studies focus on developing a better machine learning model to achieve higher performance in an IDS. Very little research has been done in understanding the quality of training datasets, especially mislabeling affects the performance of a machine learning based IDS.

In this thesis, we focus on the mislabeling issue in a machine learning based IDS. We first show the impact of mislabeling on the performance of such an IDS. Then, we propose a new algorithm called Heuristic Mislabel Identification (HMI) based on Data Shapley [6] to identify mislabels in training datasets. Based on different mislabeling scenarios, HMI heuristically and iteratively divides a training dataset into multiple groups to narrow down the location or range of mislabels. We have evaluated our method using a widely adopted IDS training dataset (i.e., CICIDS2017). The evaluation results show that HMI can identify 84% random mislabels and

78% mislabels from a single data source. The precision on both experiment above is 100% which means the suspect group must contain mislabeling samples.

KEYWORDS: Intrusion Detection System; Machine Learning; Mislabeling; Shapley

IDENTIFYING MISLABELING IN MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM

BOFAN LI

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Information Technology

ILLINOIS STATE UNIVERSITY

2021

IDENTIFYING MISLABELING IN MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM

BOFAN LI

COMMITTEE MEMBERS:

Yongning Tang, Chair

Xing Fang

ACKNOWLEDGMENTS

I would like to thank Professor Tang, committee chair and tutor. Thank you for all your time, care and patience that was put into this research project. Without him, I would never achieved this far. Thank you for every mistake that you corrected me, every idea that you inspired me and all your support. And thank you Professor Fang for giving me advice at the meeting.

Thanks to my parents for their all-aspects support for my graduate study.

B. L

CONTENTS

TABLES

FIGURES

CHAPTER I: INTRODUCTION

In this section, the background of our thesis, the problem we addressed, and challenges will be clearly claimed. Our contribution and experiment results will be listed.

Background

Recent years, fast developments in the Internet have had a profound impact on the world. Billions of people are connected to the Internet. The rapid developments in information and communication technologies around the world present a significant challenge for network security. Our research platform, IDS, plays an important role in network security. IDS (intrusion detection system) is a hardware or software program that detects intrusion on a network and monitors the network traffic for suspicious activity and alerts.

Machine learning algorithms have been widely implemented for anomaly-based IDS. The supervised learning algorithms have a higher detection rate and a lower rate of false alarms than the unsupervised learning techniques [21]. Some supervised learning algorithms which claim an accuracy over 97% with a few false alarms below 2% [7] have shown strong potential in improving the performance of an IDS. These models all depend on one premise, high quality data.

Mislabeling, however, from some unknown system errors during a data collection, or malicious manipulation for certain purposes could have a negative influence on data quality.

Challenges

As we mentioned before, in decades, the framework of building machine learning models has been implemented and studied a lot. Various machine learning models have been developed

successfully like random forest, svm, xgboost etc. However, mislabeling which could greatly affect data quality only received little attention for anomaly-based IDS. Data's quality and algorithm's suitability are two factors that can determine the accuracy of supervised learning algorithms. This emerging need emphasizes the importance of identifying mislabeling questions.

Because mislabeling has a correlation with low data quality. We will use some metrics to evaluate data quality. Then using these metrics detect mislabeling samples. To evaluate data quality in datasets, a simple method called "leave-one-out" (LOO) and a comparable complicated method Data Shapley will be utilized. LOO is a fast valuation scheme compared to Data Shapley. However, it does not satisfy the equitable valuation conditions. Data Shapley has many advanced features [6]. But the algorithm needs a lot of computing resources, even the author found a method called Truncated Monte Carlo to optimize the algorithm.

In order to have the same advanced features as Data Shapley and deal with the problem of large amounts of calculations especially for IDS which contains huge network flows. Calculating Data Shapley in groups is one of the solutions.

Contributions

In this thesis, we proposed a new method called Heuristic Mislabel Identification (HMI) based on Data Shapley which can keep the same advanced features as Data Shapley and reduce the amount of computation. The new method can evaluate data value in groups and detect mislabeling groups.

We can build an accurate random forest model for CICIDS 2017 dataset. To remove useless features and accelerate the speed of building models. Feature selection and data sample processes are shown in Chapter III. To simulate mislabeling in different scenarios, we create four

types of artificial mislabeling. We assume the target dataset used in training and calculating values is collected from different sources and the testing data is clean without mislabeling samples. Some sources contain mislabeling samples. Three types of mislabeling from single sources. One type of mislabeling is from multiple sources. We will examine our method with four types of artificial mislabeling.

The experimental results on a sample of CICIDS 2017 dataset show that, in comparison to all baseline methods, our method can achieve considerable and consistent improvements by heuristic grouping. In particular, we show that our method detects 84% random mislabels and 78% attacks concealing mislabels from a single data source. The precision on both experiment above is 100% which means the suspect group must contain mislabeling samples. For single attack from single source and single attack from multiple sources, the recall of both types of attacks has a low performance. However, the precisions are both over 50%. The precision of single attack from single source is 100%. The precision of single attack from multiple sources is 66%. Therefore, our method can work well on these four types mislabeling.

CHAPTER II: RELATED WORK

Machine Learning Based IDS

IDS (intrusion detection system) is a hardware or software program that detects intrusion on a network. Intrusion is described as any type of unauthorized activity that causes harm to a computer system [1]. The CIA defined three aims to achieve information security which are Confidentiality, Integrity, and Availability. As technology develops, there are more and more varieties of attacks. In this condition, the international standard ISO/IEC 13335-1 (2004) adds accountability, authenticity into the goals to achieve information security [2]. In contrast, intrusion is going to break these goals.

For example, distributed denial-of-service (DDoS) attack [3] is defined as a deliberate attempt to prevent legitimate use of a service. This attack has a negative effect on availability.

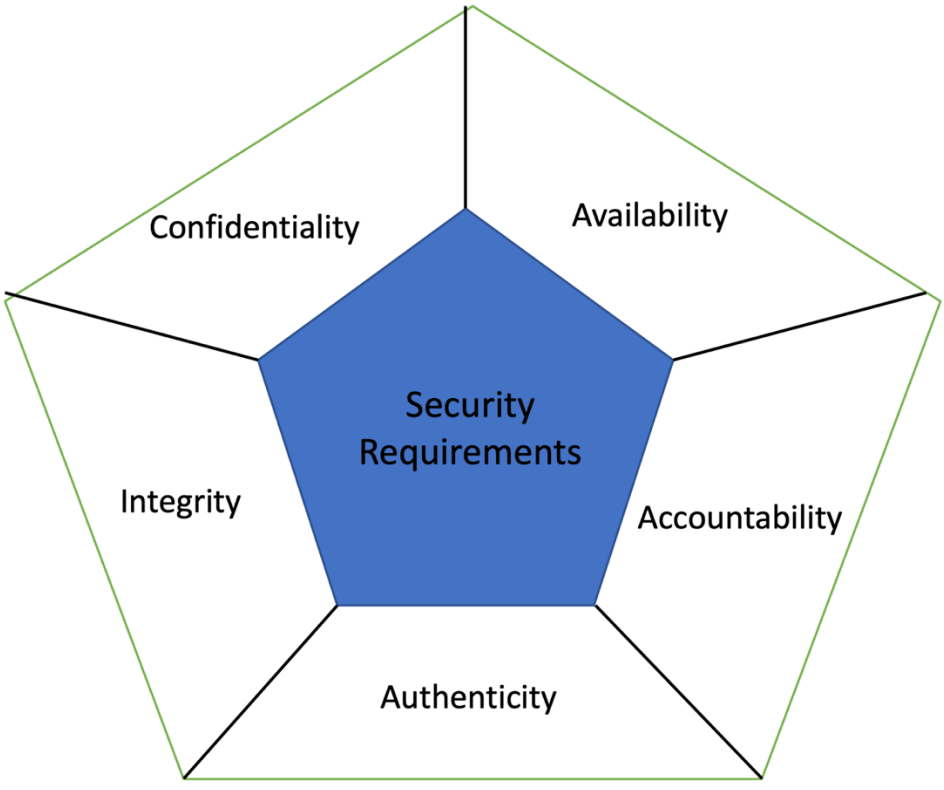

Figure 1. Five elements of security Requirements

IDS can be split into two categories: network-based IDS and host-based IDS. These two categories are based on the location of IDS. Another way to categorize IDS will be Signature-based intrusion detection systems and Anomaly-based intrusion detection systems. The differences are the following:

Signature-based IDS involves looking through network traffic for harmful data or packet sequences [4]. Attacks that are detected as misuse follow well-defined patterns that take advantage of system flaws and software bugs. These attacks can be detected by rules because they follow well-defined patterns and signatures.

```
alert icmp any any -> any any (msg:"ICMP flood"; sid:1000001; detection_filter: count 500, seconds 3;)
```

Figure 2. An example of SNORT(IDS software) rules detecting DOS

Signature-based IDS has a low false positive rate and high processing speed because rules detecting attacks are clearly defined in advance. However, signature-based intrusion detection systems can not detect Zero-day attacks.

Anomaly-based detection can detect Zero-day attacks. But anomaly-based detection can result in a significant number of false positives. In order to eliminate the large number of alerts generated, a lot of resources and time will be wasted. Our experiment is going to be based on anomaly-based IDS with machine learning.

Researchers have developed a lot of machine learning models for intrusion detection systems. Each of these models has its own characteristics. We choose using a random forest model for our IDS. The reason that we choose random forest to build a model is that it has a high performance on CICIDS dataset [7]. Compared to deep learning algorithms like CNN, random forest is less time-consuming and can export features' importance directly from the model.

Traffic classification is a very important step in IDS. Previously, the class of traffic was identified by its registered and known port. Deep Packet Inspection(DPI) is another strategy that has acquired a lot of traction in this industry[11]. To classify network traffic, DPI makes a match between the packet payload and a collection of stored signatures. When protocols are encapsulated, DPI will fail because privacy policies and laws limits access to packet content. Nowadays, Machine Learning (ML) emerged as a liable method to address the previous challenges, not only for traffic classification but also for prediction and new knowledge discovery. Many machine learning models have been implemented for IDS. For example, this paper uses the same dataset CICIDS 2017[16] to build models and extract feature importance. They summarize different model performance working on KDD kup another common dataset. They then build several models and test their performance on CICIDS 2017.

Data Quality and Mislabeling

Big data construction aims to organize data, improve organizational insight and competitiveness, and achieve business innovation and industrial upgrading. The goal of enhancing data quality is to help big data construction to succeed. Data quality isn't merely a technical issue. It can also affect business and management operations. Data quality in IDS is also a very important topic. The quality of the labels is directly related to the quality of the machine learning model. There are two possible reasons for mislabeling occurrence in IDS. The first is the errors in the data label annotation. Another is attackers could have the ability to contaminate data sources. Once attackers change the labels, the model will be trained with bias which may cause false alarm and missed attacks. How to detect the mislabeling data in machine learning has become an urgent topic to overcome to obtain improved classification accuracy.

This survey [18] defines three types of label noise by different sources and provides three methods to deal with label noise problems. Label noise cleansing is one of the three methods proposed. They take the entropy of the conditional distribution [19] as an example. Confident classifications are related to instances with a low entropy. Our paper method also is used to clean mislabeling groups in training set data. Our method is based on Data shapely which low data shapely values correspond to mislabeling instances.

Mislabeling Identification

There was an experiment similar to ours. They [15] also work on a network traffic dataset and modify labels to stimulate error labels. However, they use Deep Learning (DL) as an approach to detect abnormal data and we used Machine Learning (ML).

According to Nicolas M. Muller and Karla Markert [14], they provide a nonparametric end-to-end pipeline for detecting mislabeled instances in some common datasets including CIFAR-100, Fashion-MNIST in their study.

CHAPTER III: TECHNICAL APPROACH

Shapley Value and Data Shapley

We need a number to evaluate our data. The number is like the price of the product. Data Shapley was proposed by Amirata Ghorbani and James Y. Zou[6]. Their framework is based on Shapley Value. The Shapley value is one solution addressing cooperative game theory problem[12]. Shpapley value is a common method used to fairly and quantitatively evaluate the marginal contribution of players. It originated from cooperative games and is used in a wide range of fields, including Shapley Value as ML feature selection and ranking the importance of training data.

The marginal contribution is the benefit that one person brings to the whole group after joining. In other words, the whole group benefit with this player joining minus the whole group benefit without this player , is this person's marginal contribution.For example, three people corresponding to three players in game theory build a n apartment. Their wages are paid by the number of floors in the apartment. A is a good worker. He can build 100 floors. B can finish 80 floors. C can finish 60. When three of them work together, they encourage each other, they can achieve 300 floors. If A and B work together, they can achieve 200. If A and C work together, they can achieve 170. If B and C work together, they can achieve 150. We use v to describe their efficiency. V(A) = 100, V(B) = 80, V(C) = 60, V(AB) = 200, V(AC) = 170, V(BC) = 150, V(ABC) = 300.

They can cooperate in 6(number of players，n) n! ways.

Probability	Order of arrival	A's marginal contribution	B's marginal contribution	C's marginal contribution
1/6	first A then B last C; ABC	V(A) = 100	V (AB)-V(A)=200-100=100	V (ABC)-V(AB)=300-200=100
1/6	first A then C last B ; ACB	V (A) = 100	V (ACB)-V(AC)=300-170=130	V (AC)- V(A)=170-100=70
1/6	first B then A last C; BAC	V (BA)- V (B)=200-80=120	V (B) = 80	V (BAC)- V (BA)=300-200=100
1/6	first B then C last A; BCA	V (BCA)- V (BC)=300-150=150	V (B) = 80	V (BC)- V (B)=150-80=70
1/6	first C then B last A; CBA	V (CBA)- V (CB)=300-150=150	V (CB)- V (B)=150-80=70	V (C) = 60
1/6	first C then A last B; CAB	V (CA)- V (C)=170-60=110	V (CAB)- V (CA)=300-170=130	V (C) = 60

Table 1. Shapley value Calculation process

In the Data Shapley paper, they treat each data instance as a player and the model performance as a wage. The marginal contribution in the paper is the difference of performance. They mentioned a "leave-one-out" (LOO) method which evaluates how much performance will change if they remove the data instance. We view i as the value of instance i. D= {1,...,n} is the

total training set. S as a subset of D, S D. V(D) is the performance of a model trained by training set D. V(D - {i}) is the performance of a model trained by training set D without instance i. The LOO method calculated the value of instance by equation: φ_i= V(D) - V(D-{i}). It is a simple way to calculate the data value.

The disadvantage of this method is "this valuation scheme does not satisfy the equitable valuation conditions". Shapley value achieves "Efficiency"， "Symmetry"， "Dummy", and "Additivity". Therefore, Data Shapley inherits these attributes. Data Shapley calculate the data value by equation:

$$\varphi_i = C \sum_{S \subseteq D-\{i\}} \frac{V(S \cup [i]) - V(S)}{\binom{n-1}{|s|}} \quad (1)$$

Data Shapley not only found a way to calculate the value of the instance, but also optimized the computational complexity by extending Monte-Carlo approximation methods and a truncated method. Data Shapley can be used in many applications. According to their paper, it can be used for "Identifying data quality" and "Using value to adapt to new data". Many experiments show TMC-Shapley has a better performance than LOO. In our experiment, we ran a random forest model on the subset of CICIDS 2017 dataset with 100 samples. After calculating the values of each instance, we sorted the values and removed the highest value to the lowest value from the sub-dataset. We found the accuracy performance decreasing from 100%, 80%, 70%, 40%. The performance is dropping due to the limitation size of our testing data. However, we still can see TMC-Shapley has a bigger influence than "LOO".

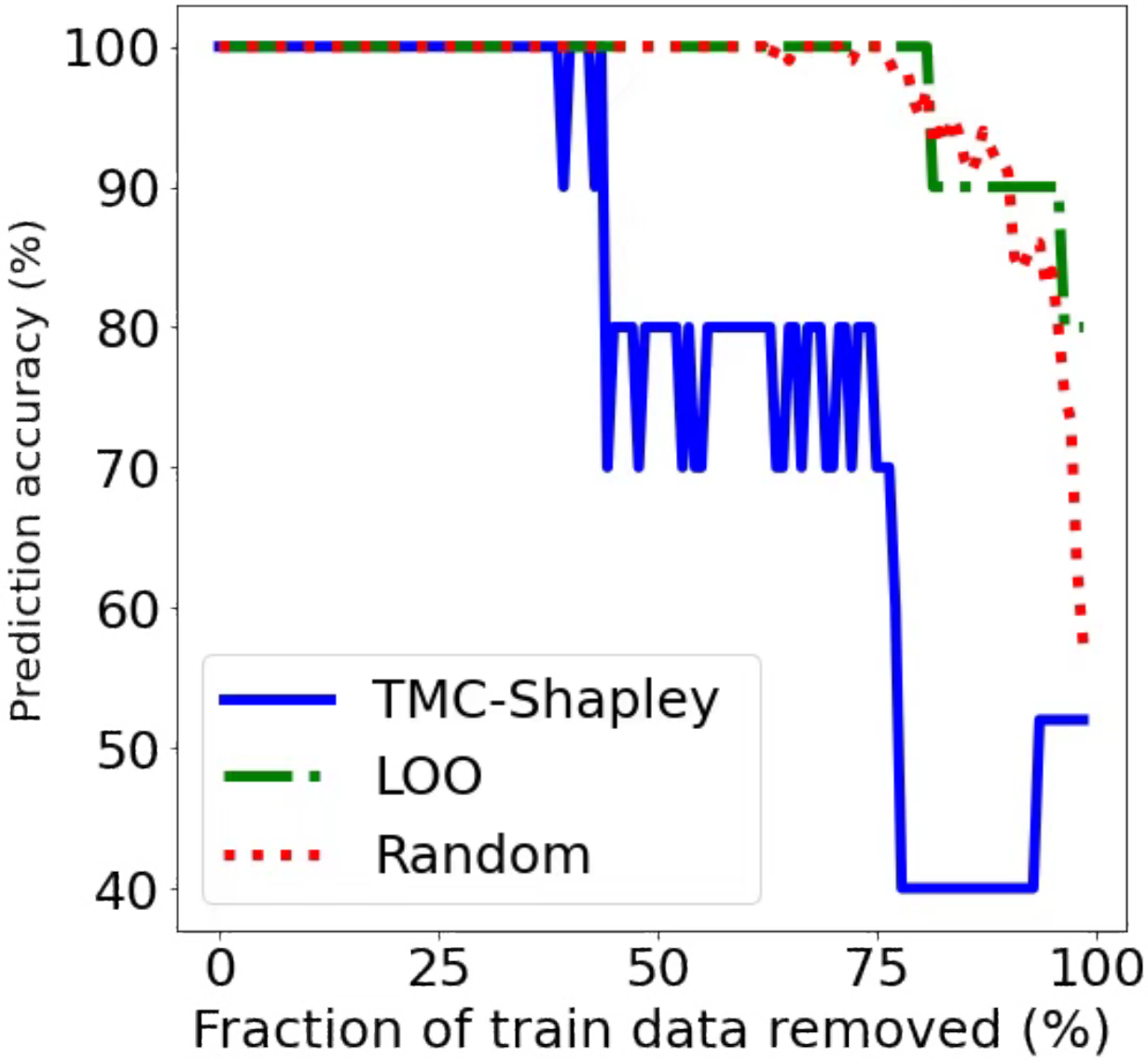

Figure 3. Data Shapley on IDS removing high value instance

The other experiment is we ran a random forest model on the subset of CICIDS 2017 dataset with 420 samples divided into 140 groups. We then apply Data Shapley on the subset of data with 40% random flip labels. After calculating the values of each instance, we sorted the values and removed the lowest value to the highest value. We found the accuracy performance increasing from 63% to around 95%. The reason for the performance increase is that many

mislabeled instances were removed. In the last part of the paragraph, accuracy is reduced because there is not enough data to support the model.

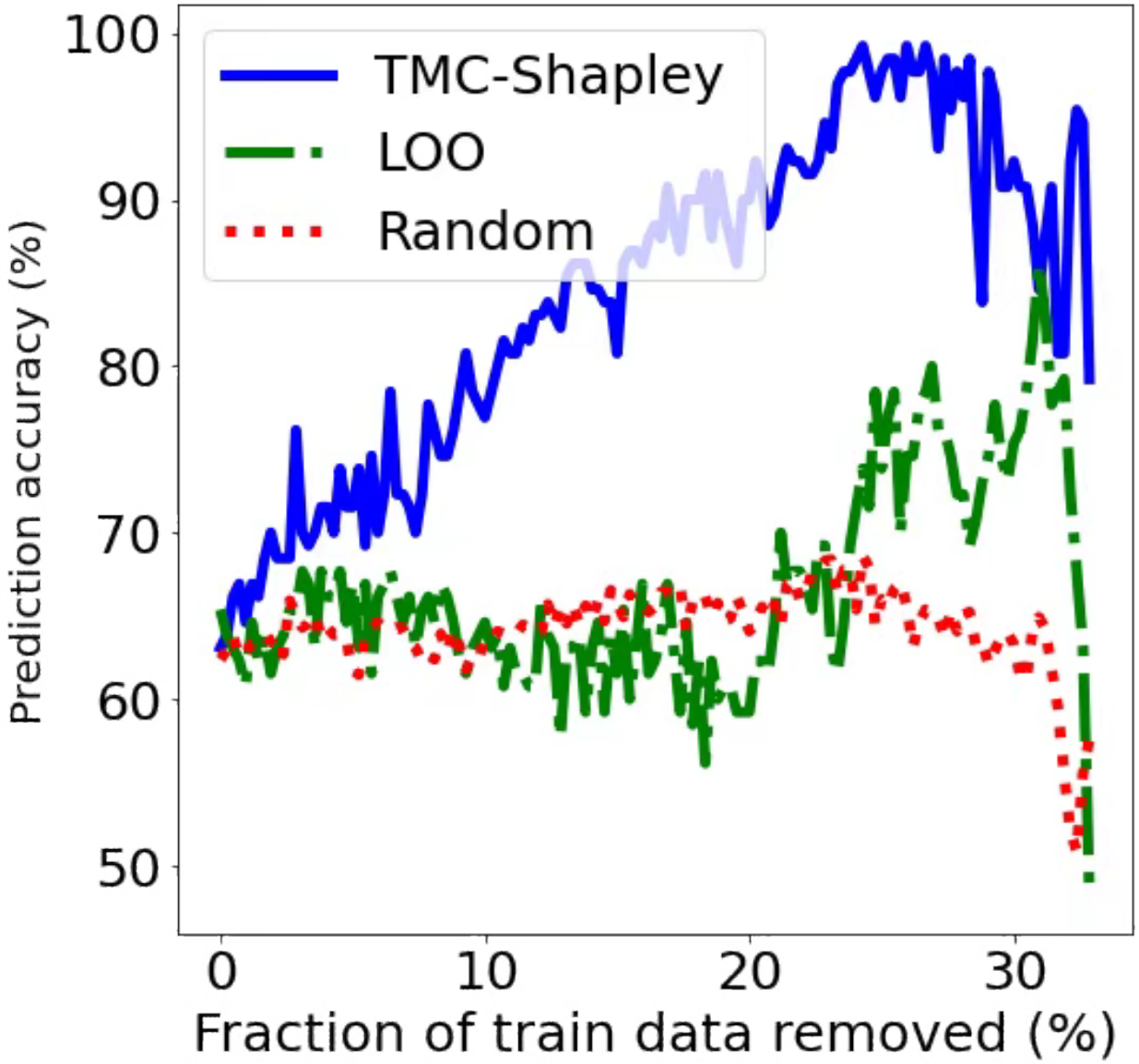

Figure 4. Data Shapley on IDS removing low value instance

The inspiration for our paper came from the low Data Shapley value of the instance means the instance has a small or negative effect on the overall performance. If we can get a high performance from a normal dataset, once one normal instance's label is changed, this behavior

might hurt the performance which can be detected by Data Shapley value. The result looks good. However, we can only run Data Shapley on a small size of training set. Because the computation resources are limited. The Data Shapley value still needs a lot of computation even with optimization. We are going to analyze the computational complexity in the following section.

Computation Challenge

Even Data Shapley has these advantages mentioned before, according to equation (1), it claims calculating Data Shapley value requires an exponentially large number of computations related to the size of the training set. Data Shapley addresses this problem by extending Monte-Carlo approximation methods and a truncated method. Monte-Carlo approximation methods is a kind of sample algorithm which sacrifices accuracy but saves computing resources. From their code published, Monte-Carlo approximation methods can control the precision of Data Shapley value by a parameter called err in the dshap.run() function. The default parameter will lead to the value convergence in 10% deviation of the precisely Data Shapley value. The Data Shapley author suggests “usually 3n Monte Carlo samples is sufficient for convergence”. 3n is the number of iterations. The number of iterations is another hyperparameter in the code. One iteration process is shown below. One iteration matches one row in table 1. It randomly permutes the total train set (D). The subset(S) of D is the real training set from an empty set to full size or meets the truncated condition. When S is empty, the performance is a random score. S is increased by adding one instance in the set in the next loop. And the new S is used to build a new model and test the performance. The new performance score minus old performance score is the marginal contribution of instance just adding. When S size gets bigger and bigger, the performance will tend to be stable. The variance of performance is less than tolerance * v(D) for five consecutive

times. The algorithm is truncated. Therefore, every itermatation don't need to loop the full size of D. When the algorithm is truncated, the other instances without adding into S will be equal to 0 marginal contribution in this iteration. We donate m(i) as a marginal contribution of instance (i) in the graph below. In conclusion, even with the truncated algorithm, the worst situation is that one iteration needs to train the size of the D times model. If we use only 100 samples in D, we need to build models 100 times in the worst situation.

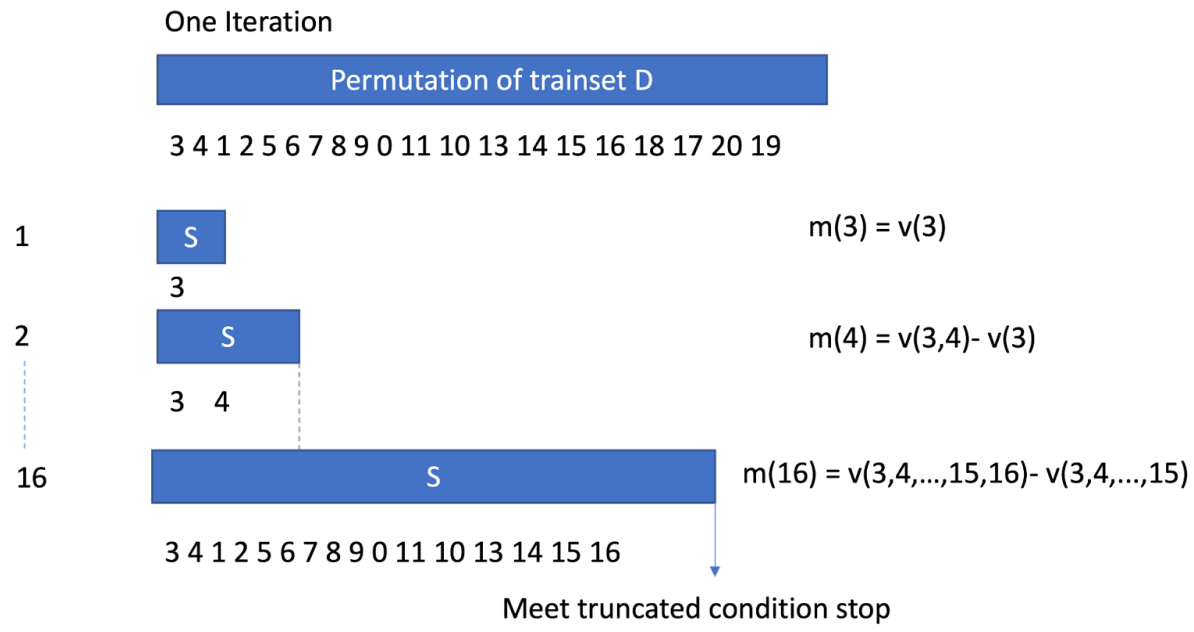

Figure 5. One iteration in Data Shapley

After getting the marginal contribution of each instance, they calculate Data Shapley value by adding all the contributions aggregating by each instance. The result is then divided by the number of iterations from now on. An approximation Data Shapley value is created. The next question is how to decide how many iterations are needed. The Data Shapley algorithm addresses the problem by evaluating the variance of the approximation Data Shapley value. Once the value is no more than 0.1 in nearly 100 variance, the value is treated as convergence and finished. The number of iterations is hard to decide[13]. According to the paper, "The sample

size needed for a study depends on many factors including the size of the model, distribution of the variables, amount of missing data, reliability of the variables, and strength of the relationships among the variables."

Therefore, we only can estimate the number of iterations roughly by our experiments. In our experiment, usually around 10n Monte Carlo samples is sufficient for convergence. 'n' is the number of samples in the train set. For example, we use 6000 samples in a train set. We then use a group to divide the data set. Each group has 60 samples, so there are 100 groups. If we don't group the data, the result can be done in Google Colab in one day. The mean time of each iteration costs 46.9s. In one iteration, the relationship between time cost and the size of data size are shown below. When the dataset has 6000 samples, training a random forest model needs around 1 second. The cost of time and data size has a positive linear correlation. The total time cost is shown in Table 2.

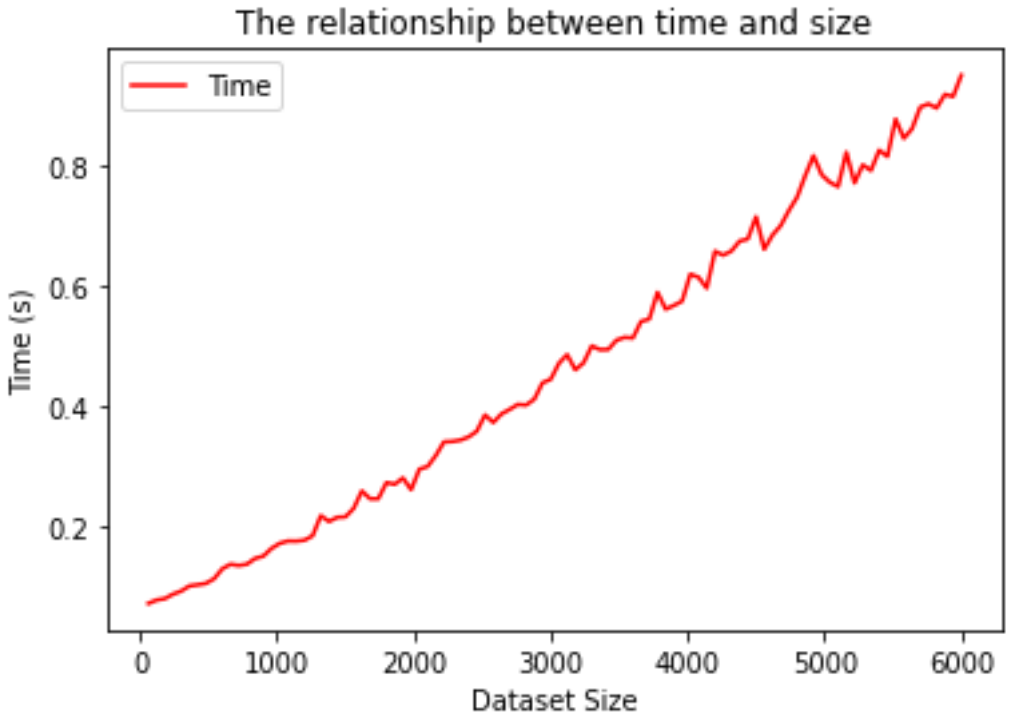

Figure 6. The relationship between Dataset size and time on random forest model

trainset	the number of group	the number of iteration	time
6000	100	1200	56808s (15.7h)
420	140	1800	25200s (7h)
140	140	1500	14400s (4h)
2014	21	540	156s

Table 2. Time consumption

T(total) = The number of iterations * The number of group* T(model)

T(model) has a positive linear correlation with the number of train sets which is shown in the graph above. The number of group can be precisely calculated by the number of train sets divided by the size of the group. One iteration is independent, so it can be run parallel. However, even if it can run parallel, Data Shapley still needs a lot of computing resources. The number of iterations As we wrote before, it is hard to know. According to our experiment a group with large size i's more likely to converge early. Therefore, we proposed a method called Heuristic Mislabel Identification to address the problem that Data Shapley needs many computer resources.

Heuristic Mislabel identification

Algorithm 1 Heuristic Mislabel Identification

Input: Train set data with mislabel $df1$, Test set data $df2$, search-deep $deep$, The number of group $glist$

Output: The index suspected label G

$n = 0$
while $n \leftarrow deep$ **do**
  the number of group$ng \geq glist[n]$
  $sources$: A dict containing specific group information using in Data Shapley
  if $G == None$ **then**
    $sources$ changed according to ng
  else
    $sources$ changed according to ng and G
    Only changed suspect G according to ng
    Old sources without suspect stay the same
  end if
  $V = DataShapley(df1, df2, sources)$
  Find suspect group G according to V
  $N \leftarrow N + 1$
end while
return G

Figure 7. Heuristic mislabel identification algorithm

Data Shapley values can be grouped when we want to know their marginal contribution as a group. Each instance in the same group will be endowed with the same value. Group can reduce Data Shapley computing time linearly. Therefore, we can begin with a small number of the size of the group. The data is divided into different groups. After we get the value of the group, we find the suspicious group with value. Lowest value means it hurts the model performance or doesn't help the model performance. One premise we know is that random Forest has high accuracy in CICIDS 2017. This is confirmed by our experiments and other papers[8]. Therefore,

low Data Shapley value has a high possibility including mislabeling. Once the suspicious group is detected, we divide these groups into smaller groups with a new group number.

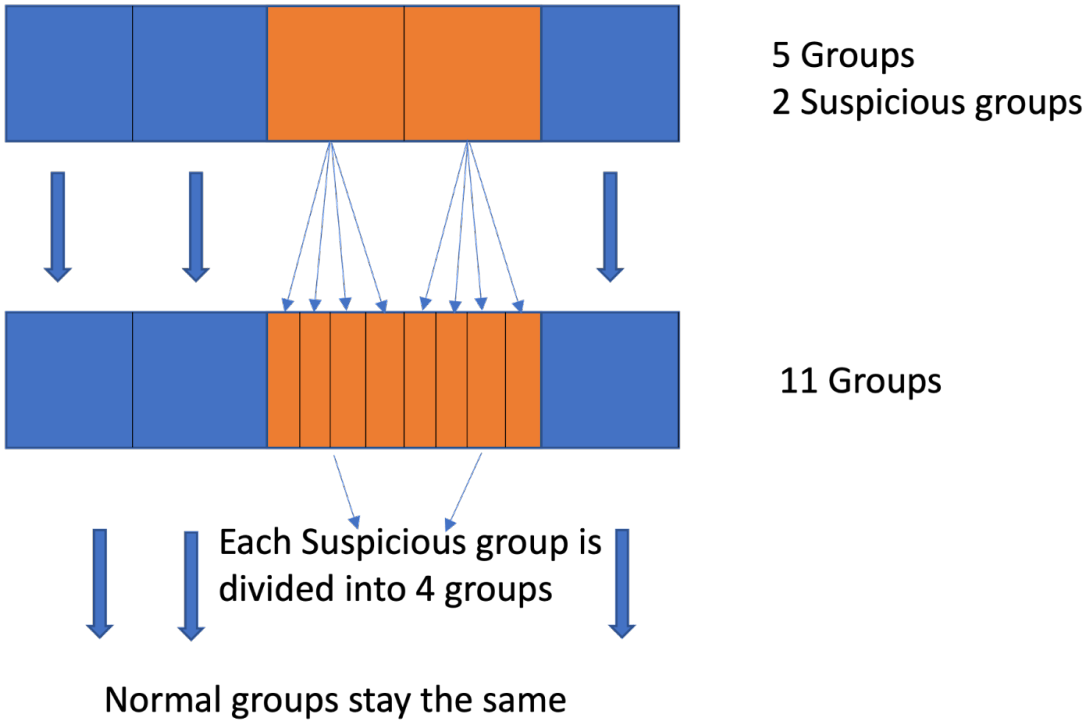

Figure 8. How to divide groups in Heuristic mislabel identification algorithm

Then we use the new group information to calculate Data Shapley value. After this time we get more value from groups, we decide whether to continue to make a new group or return the program with suspicious groups.

CHAPTER IV: EVALUATION

Performance Metrics

		Actual condition	
		Condition positive(P)	Condition negative(N)
Predicted condition	Predicted Condition positive(PP)	TP	FP
	Predicted Condition negative(PN)	FN	TN

Table 3. Performance Metrics

Precision is named positive predictive value that is the ability of the classifier not to label as positive a sample that is negative. (TP/TP+FP) This metric lets us know how many mislabeled ranges which we detect are misjudged.

Recall is also named true positive rate that is the ability of the classifier to find all the positive samples. (TP/TP+FN) This metric lets us know how many mislabels have been discovered.

Accuracy (ACC) is (TP+TN)/(TP+TN+FP+FN) the overall performance. This metric is used in testing the performance of the random forest model.

Experiment Settings

CIC-IDS2017 was created by the Canadian institute for cybersecurity. They evaluated the 11 datasets and suggested most of these datasets are unreliable and out of date. The data we used is CSV format. The CSV file was produced by a tool called CICFlowMeter. CICFlowMeter can automatically analyze PCAPs files and convert PCAPs files into a CSV format file. Each network flow corresponds to a row of data. The CIC-IDS2107 dataset in CSV format is a tabular dataset. There are 2830743 rows and 79 columns in the data. After we clean the dataset by removing rows with NAN and all 0 columns, the distribution of labels are shown below.

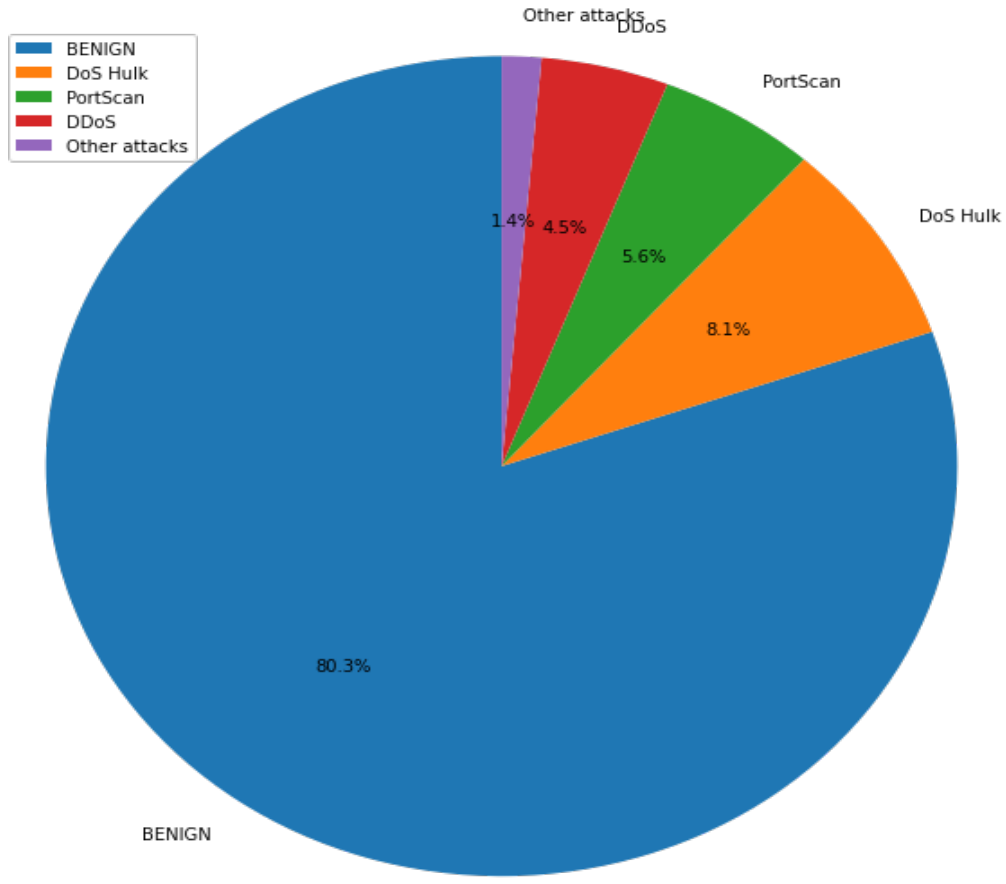

Figure 9. Distribution of CICIDS2017 attacks

The number of instances in the CICIDS 2017 dataset is too many to do experiments with Data Shapley. Therefore, we sample the dataset. To select features, we do a machine learning process to get the importance of features.

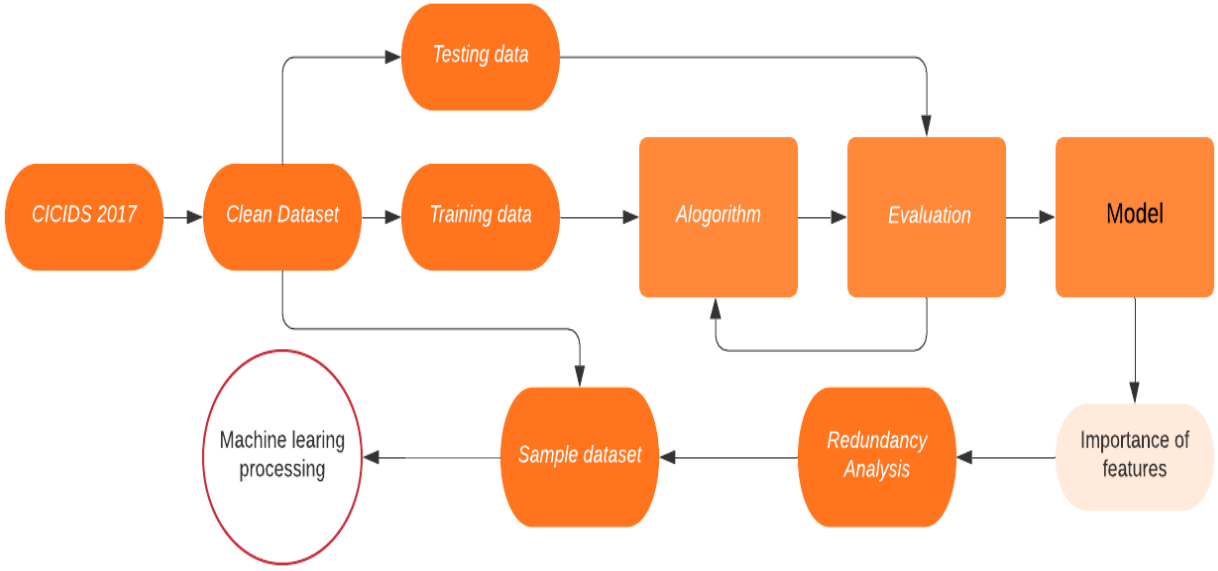

Figure 10. Extracting importance with machine learning processes

Data Cleaning

To run a machine learning model, we remove data instances containing 'NAN' and 'infinity'. Since all 8 columns are '0', we have to remove all of them. The data size goes from (2830743, 79) to (2827876, 71).

Data Preprocessing

There are 15 types of attacks in CICIDS 2017. Benign accounts for around 80.3%, DoS Hulk accounts for 8%, PortScan accounts for 5.6%, DDOs account for 4.5%, other attacks only accounts for 1.4%. We then use MinMaxScaler to Scale each feature to a specific range. We use a common rate of 7:3 for machine learning algorithms, resulting in 1,979,513 rows in the training set and 848,363 rows in the testing set.

Data Modeling

We build a random forest model on the dataset with the default parameters built-in sklearn.

Model Evaluation

We evaluate the random forest model with the test dataset. This model achieves a high performance. Accuracy reaches 99.84%. The other metrics are shown in the graph.

```
Accuracy: 0.9984


Classification Report

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classificatio
  _warn_prf(average, modifier, msg_start, len(result))
                         precision    recall  f1-score   support

                 BENIGN     0.9993    0.9991    0.9992    681647
                    Bot     0.8045    0.5544    0.6564       579
                   DDoS     0.9999    0.9996    0.9998     38393
          DoS GoldenEye     0.9947    0.9970    0.9959      3009
               DoS Hulk     0.9963    0.9975    0.9969     68922
       DoS Slowhttptest     0.9924    0.9899    0.9911      1578
          DoS slowloris     0.9955    0.9927    0.9941      1788
            FTP-Patator     0.9996    0.9996    0.9996      2392
             Heartbleed     1.0000    1.0000    1.0000         2
           Infiltration     1.0000    0.8000    0.8889        10
               PortScan     0.9940    0.9996    0.9968     47664
            SSH-Patator     1.0000    0.9994    0.9997      1773
  Web Attack Brute Force     0.7467    0.7943    0.7698       423
Web Attack Sql Injection     0.0000    0.0000    0.0000         5
          Web Attack XSS     0.4148    0.3146    0.3578       178

               accuracy                         0.9984    848363
              macro avg     0.8625    0.8292    0.8431    848363
           weighted avg     0.9984    0.9984    0.9984    848363
```

Figure 11. Classification Performance

Feature Selection

We export the importance of the model. We found the results are unstable, even with the same train set. Therefore, we build the model ten times, accumulate their features' importance, and sort the features' importance. The result is shown below. We select 16 features because there is a gap between 16th and 17th feature importance. The result is different from the paper[8] introducing CICIDS 2017 dataset. Because they build the random forest model on a whole dataset and multiply "the average standardized mean value of each feature split on each class". We do not need to multiply the value, because we do a MinMax scale in the data preprocessing steps, even feature standardization is not necessary to build a random forest model.

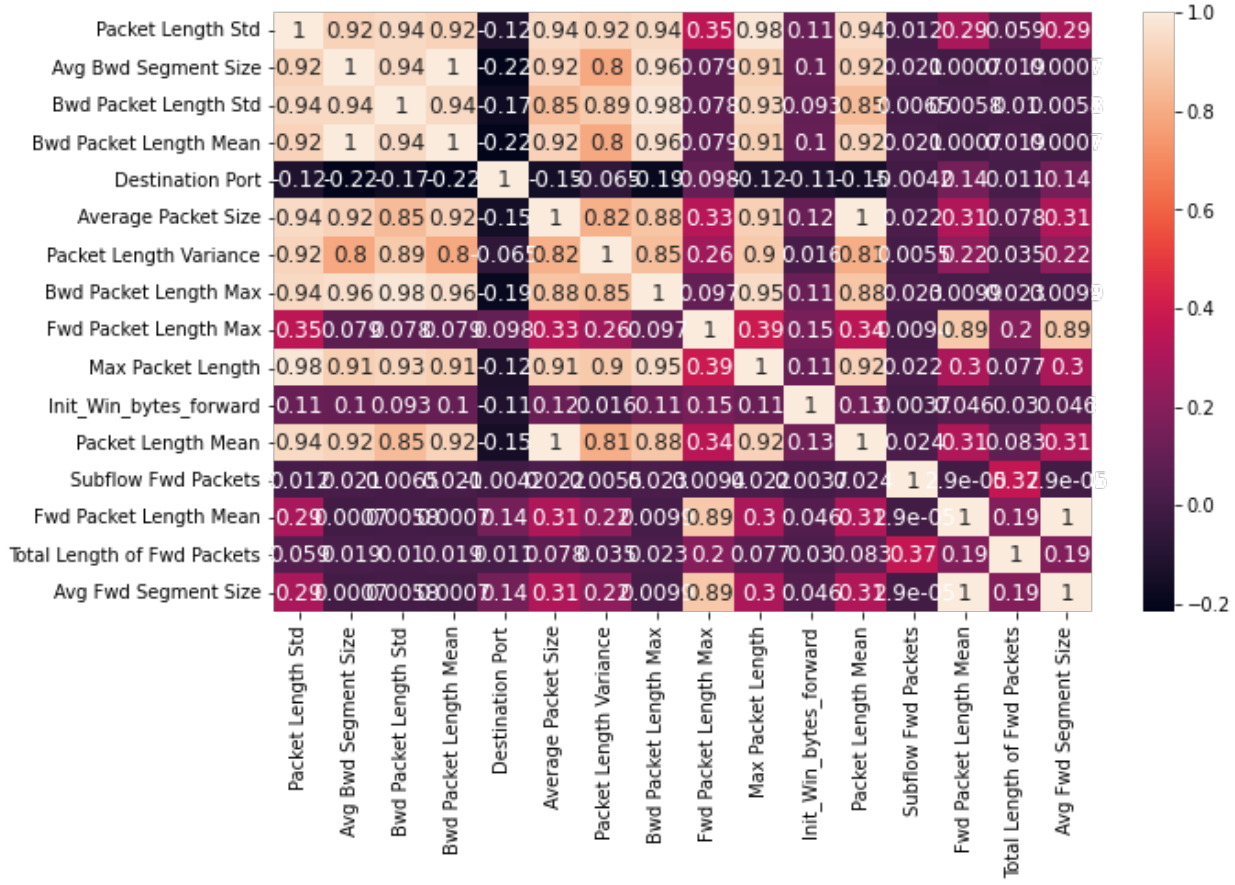

Figure 12. Redundant analysis

We analyze the correlation ship with the 16 most important features. We can find ‘Ava Bwd Segment Size’ has a high correlation ship with ‘Bwd Packet Length Mean’. So these two features can be defined as redundant features. We should use one of the redundant features instead of using both of them. We select the features with no pairs that have a correlation more than 0.9. Finally, 'Packet Length Std' , 'Destination Port', 'Fwd Packet Length Max', 'Init_Win_bytes_forward', 'Subflow Fwd Packets', 'Fwd Packet Length Mean', 'Total Length of Fwd Packets', and 'Avg Fwd Segment Size', 8 features are selected. We build a model only with these features and evaluate the model. The mode still has good performance. The model’s accuracy is 99.73%.

Then we sample 4000 rows of the CICIDS 2017 dataset with selected features. The subset retains the same rate of attack in the full dataset. To get a better performance of data, we split the data into 2000 samples training data, 2000 samples testing data. The model accuracy is 91.7%. We are going to use this sample to do next experiments. The training set we are going to use has 2014 samples which contain all types of attacks. We oversample ‘heartbleed’ and other four types of attacks whose size is small to ensure all attacks can exist in the experiments.

We have four ways to modify labels to stimulate attacks. The scenario is that we have a new set of data from different sources that we need to put into the training set. However, the data is contaminated from single sources or multiple sources. We propose three cases 1,2,3 from a single source and case 4 from multiple sources. The details of these cases are in the following essay.

Case 1: Single Source Random Mislabeling

Algorithm 2 Single source random mislabeling

Input: clean train set label $train_y$

Output: train set label with mislabeling $Mtrain_y$

```
Mtrain_y ← train_y
for i in random range of train_y do
    Mtrain_y[i] ← randomChange(train_y[i])
end for
return Mtrain_y
```

Figure 13. Modify the labels to simulate attack type1

This scenario mimics an attack where a hacker can access a portion of the data source. He arbitrarily changes the data label in an attempt to influence the model.

In our experiment, we use the subset data mentioned in experiment settings. The first part of the experiment is a robust test. We want to know how much the mislabel will influence the model performance. We modify the mislabeling range from 0% to 50% of the train set. For every 5% increase we train a random forest model to test model performance on the mislabeled dataset. The result is shown in the following graph. When we flip 40% of train set labels, the model overall performance reduces to 70%. The range of performance changes is 23%.

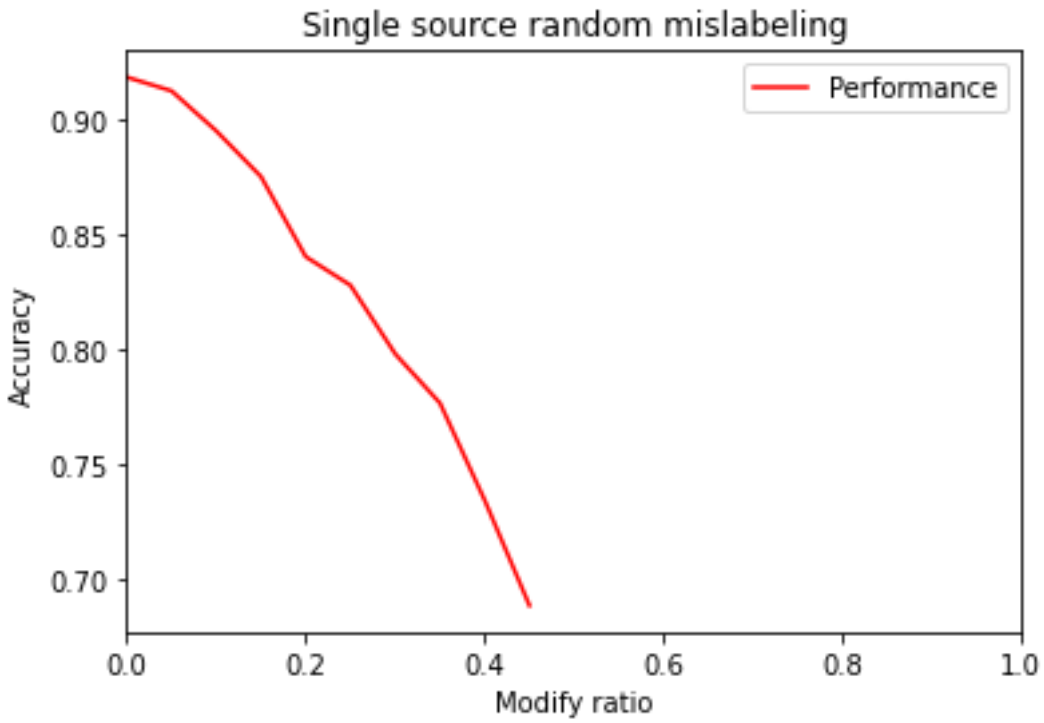

Figure 14. Robustness analysis type1

The next part of the experiment, we apply our Heuristic Mislabel Identification algorithm to the dataset. We flip 20% of the train set's labels. This data is divided into 21 groups. Each group contains 100 samples, except the final one with 14 samples. We calculate group Data Shapley values on the dataset. There are four groups with a Data Shapley value of 0. Others are all positive numbers. We consider these four groups to be suspicious groups. Next, we divide these four groups into 40 groups. Each suspicious group is divided into 10 groups. We combine these 40 groups and other 21-4=17 groups. We get 57 groups of data. Then we calculate Data Shapley values for these 57 groups.

After we get the values, we check it and pick the lowest values. 84% of the mislabeling area is successfully identified. The precision is 100%.

Case 2: Single Source Attack Concealing Mislabeling

Algorithm 3 Single source attack concealing mislabeling

Input: clean train set label $train_y$
Output: train set label with mislabeling $Mtrain_y$

```
Mtrain_y ← train_y
for i in random range of train_y do
    if train_y[i] == attack then
        Mtrain_y[i] ← Benign
    end if
end for
return Mtrain_y
```

Figure 15. Modify the labels to simulate attack type2

This scenario mimics an attack where a hacker can also access a portion of the data source, same as the case 1. The difference is that he changes all attack labels to benign and tries to conceal attack samples to make the model can't learn the attack's features. In our experiment, we do the same preprocessing of data as case 1. About the robust test, we modify the mislabeling range from 0% to 50% of the train set. For every 5% increase we train a random forest model to test model performance on the mislabeled dataset. The result is shown in the following graph. When we flip 40% of train set labels, the model overall performance reduces to 88.5%. The range of performance changes around 4%.

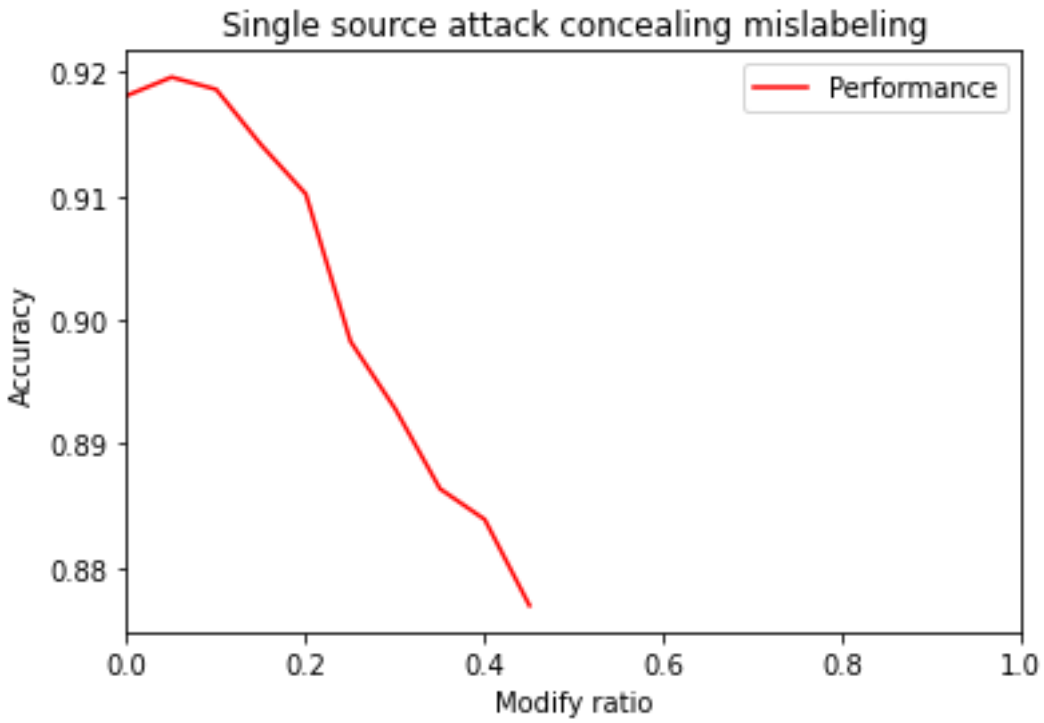

Figure 16. Robustness analysis type2

The second part of experiments in case 2, Heuristic Mislabel Identification. After we calculate the value of 21 groups in the first round, we detect four groups whose value is under 0 that means this four group has a bad influence on the model performance. Then we divide these groups into 40 smaller groups and combine with other 16 groups. Next, we use these new groups to calculate Data Shapley values. Some groups whose size is 10 have a positive Shapley value. It might mean there are a few numbers of samples in the group that are mislabeling. We select the suspect group by the low Shapley value.

The recall rate is 78% which means 78% mislabels are detected by the group. The precision is 100% which means all suspect groups contain mislabeling.

Case 3: Single Source Single Attack Mislabeling

Algorithm 4 Single source single attack mislabeling

Input: clean train set label $train_y$
Output: train set label with mislabeling $Mtrain_y$

```
Mtrain_y ← train_y
for i in random range of train_y do
    if train_y[i] == a type of attack then
        Mtrain_y[i] ← Benign
    end if
end for
return Mtrain_y
```

Figure 17. Modify the labels to simulate attack type2

This scenario mimics an attack where a hacker can also access a portion of the data source, same as case 1 and case2. The difference is that he changes only one type of attack label to benign and tries to conceal these specific attack samples to make the model can't learn the specific attacks' features. In our experiment, we do the same preprocessing of data as case 1. We search a random range, 20% of the train set, and change the label from a specific attack to benign. For the robust test, we still use the same range of modifying ratio from 0% to 50%. For every 5% increase we train a random forest model to test model performance on the mislabeled dataset. The range of performance changes around 1.2% which is a small number for the performance.

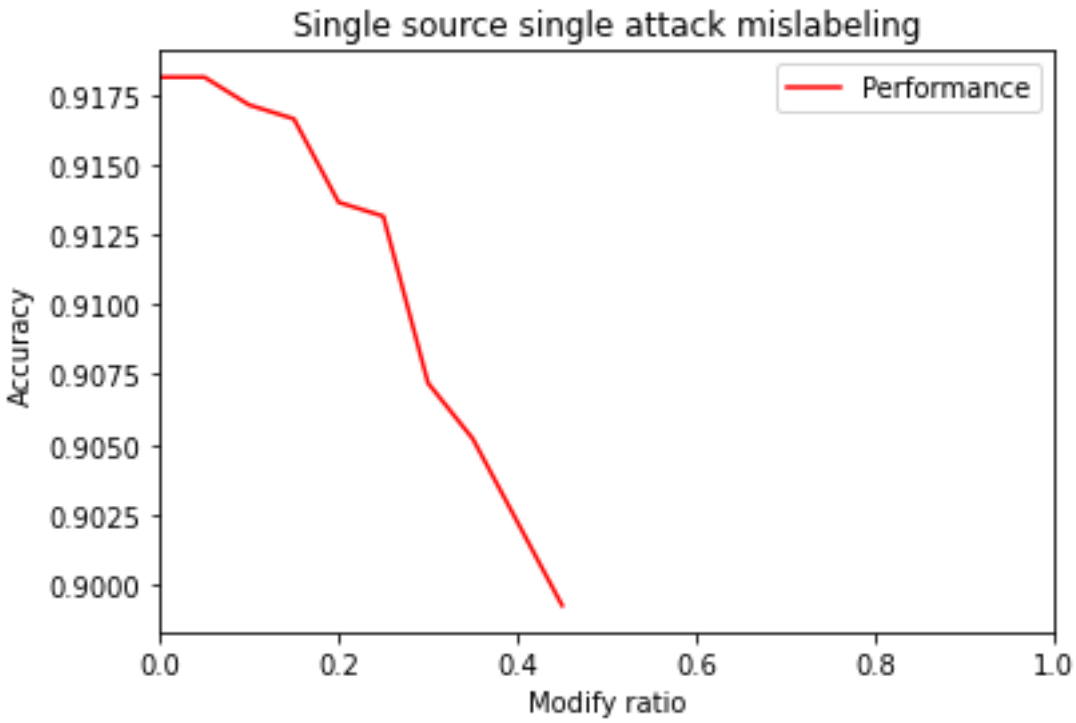

Figure 18. Robustness analysis type3

The second part of experiments in case 3. After we calculate the value of 21 groups in the first round, we detect one group whose value is under 0 that means only one group has a bad influence on the model performance. The group contains 100 samples. Then we divide this group into 10 smaller groups and combine with other 20 groups. Next, we use these new groups to calculate Data Shapley values.

The recall is 13%. It is a very low performance. Because in the first round, our algorithm only detects one group has a value lower than 0. We review the value of groups in the first round. We found the first and second lowest values contain mislabeling groups. However, even recall has a low value, when taking the two lowest value groups, the precision is 100% which means these groups contain mislabeling samples.

Case 4: Multi Source Single Attack Mislabeling

Algorithm 5 Multi source single attack mislabeling

Input: clean train set label $train_y$
Output: train set label with mislabeling $Mtrain_y$

```
count = 0
for i in train_y do
    if i == a type of attack then
        count ← count + 1
    end if
end for
Number of attacks that need to be modified n
n ← count* modified ratio
count = 0
for i in random permutation of train_y do
    if  count > n then
        break loop
    end if
    if train_y[i] == a type of attack then
        train_y[i] ← Benign
        count ← count + 1
    end if
end for
Mtrain_y ← train_y[i]
return  Mtrain_y
```

Figure 19. Modify the labels to simulate attack type4

This scenario mimics hackers who can access the full size of a train set and have the authority to change labels. He may want to hide one specific type of attack by changing any percent of the specific attack samples in the train set. In our experiment, we split the dataset as the same as before. But for the robust test, wel use the same range of modifying ratio from 0% to 100%. For every 5% increase we train a random forest model to test model performance on the mislabeled dataset. The overall performance drops from 91.8% to 84.4%. The range of performance changes is around 7%.

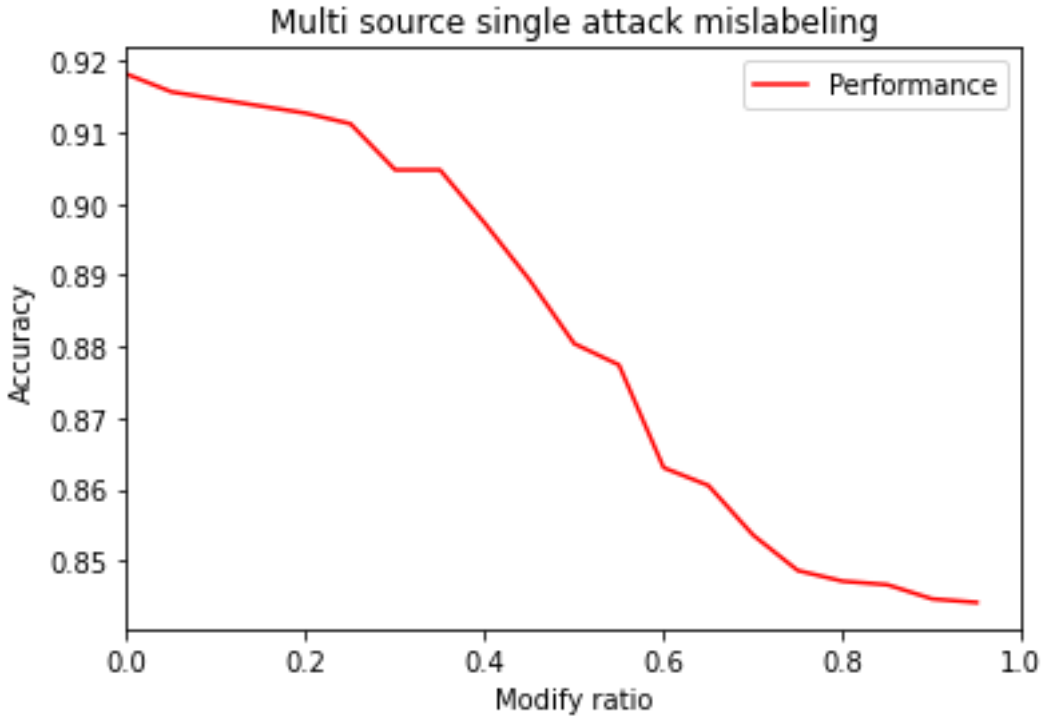

Figure 20. Robustness analysis type4

The second part of experiments in case 4. After we calculate the value of 21 groups in the first round, we detect one group whose value is under 0 that means only one group has a bad influence on the model performance. It has the same results as case3. The group contains 100 samples. Then we divide this group into 10 smaller groups and combine with other 20 groups. Next, we use these new groups to calculate Data Shapley values.

The recall is 14%. It is also a very low performance. Because the modified sample distribution is too scattered. However, the precision is 66%. Therefore, our method can still work on this type of attacks.

CHAPTER V: DISCUSSION AND FINAL REMARKS

In this paper, we propose Heuristic Mislabel Identification (HIM) to optimize the problem of Data Shapley consuming a lot of computing resources. The experiment results of HIM showed that it can be effectively utilized to identify single source random mislabeling and single source attack concealing mislabeling. HIM has limited effect on single source single attack mislabeling and multi-source single attack mislabeling. One possible reason is the ratio of mislabeling accounts for too little of the total data.

In the future, we can explore the following directions: 1) the iteration of approximating Shapley value could be run parallelly. Therefore, we could also make our HIM run parallelly. In this condition, a larger train set data could be used in the experiments which will include more attack samples; and 2) the suspect group number of HIM can be set bigger. It may cause more false positive detection. However, the recall of attacks could increase.

REFERENCES

[1] Khraisat, A., Gondal, I., Vamplew, P. et al. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecur 2, 20 (2019). https://doi.org/10.1186/s42400-019-0038-7

[2] Rossouw Von Solms and Johan Van Niekerk. 2013. From information security to cyber security. Comput. Secur. 38 (October, 2013), 97–102. DOI:https://doi.org/10.1016/j.cose.2013.04.004

[3] Jelena Mirkovic and Peter Reiher. 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev. 34, 2 (April 2004), 39–53. DOI:https://doi.org/10.1145/997150.997156

[4] Veeramreddy, Jyothsna & Prasad, V. & Prasad, Koneti. (2011). A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications. 28. 26-35. 10.5120/3399-4730.

[5] Rädsch, Tim & Eckhardt, Sven & Leiser, Florian & Pandl, Konstantin & Thiebes, Scott & Sunyaev, Ali. (2021). What Your Radiologist Might be Missing: Using Machine Learning to Identify Mislabeled Instances of X-ray Images. 10.24251/HICSS.2021.157.

[6] Ghorbani, Amirata & Zou, James. (2019). Data Shapley: Equitable Valuation of Data for Machine Learning.

[7] Elmrabit, Nebrase & Zhou, Feixiang & Li, Fengyin & Zhou, Huiyu. (2020). Evaluation of Machine Learning Algorithms for Anomaly Detection. 1-8. 10.1109/CyberSecurity49315.2020.9138871.

[8] Sharafaldin, Iman & Habibi Lashkari, Arash & Ghorbani, Ali. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. 108-116. 10.5220/0006639801080116.

[9] Guyon, Isabelle & Elisseeff, André. (2003). An Introduction of Variable and Feature Selection. J. Machine Learning Research Special Issue on Variable and Feature Selection. 3. 1157 - 1182. 10.1162/153244303322753616.

[10] Mishra, Preeti & Varadharajan, Vijay & Tupakula, Uday & Pilli, Emmanuel. (2018). A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. IEEE Communications Surveys & Tutorials. PP. 1-1. 10.1109/COMST.2018.2847722.

[11] Pacheco, Fannia & Exposito, Ernesto & Gineste, Mathieu & Baudoin, Cédric & Aguilar, Jose. (2018). Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey. IEEE Communications Surveys & Tutorials. PP. 1-1. 10.1109/COMST.2018.2883147.

[12] Winter, E. (2002). Chapter 53 The shapley value. Handbook of Game Theory With Economic Applications, 3, 2025-2054.

[13] Muthén, Linda & Muthén, Bengt. (2009). How to Use a Monte Carlo Study to Decide on Sample Size and Determine Power. Structural Equation Modeling: A Multidisciplinary Journal. 9. 10.1207/S15328007SEM0904_8.

[14] Muller, Nicolas & Markert, Karla. (2019). Identifying Mislabeled Instances in Classification Datasets. 1-8. 10.1109/IJCNN.2019.8851920.

[15] Chen, Wencheng & Li, Hongyu & Zeng, Yi & Ren, Zichang & Zheng, Xingxin. (2019). Model Uncertainty for Annotation Error Correction in Deep Learning Based Intrusion Detection System. 137-142. 10.1109/SmartCloud.2019.00033.

[16] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," in IEEE Access, vol. 8, pp. 132911-132921, 2020, doi: 10.1109/ACCESS.2020.3009843.

[17] Rajpurkar, P. et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv Prepr. arXiv1711.05225 (2017).

[18]Benoˆıt Frenay and Michel Verleysen. Classification in the presence of ´ label noise: a survey. IEEE transactions on neural networks and learning systems, 25(5):845–869, 2014.

[19]J.-W. Sun, F.-Y. Zhao, C.-J. Wang, and S.-F. Chen, “Identifying and correcting mislabeled training instances,” in Proc. Future Generat. Commun. Netw., v

[20] Panigrahi, Ranjit & Borah, Samarjeet. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. 7. 479-482.

[21] M. R., G.R., Ahmed, C.M. & Mathur, A. Machine learning for intrusion detection in industrial control systems: challenges and lessons from experimental evaluation. Cybersecur 4, 27 (2021). https://doi.org/10.1186/s42400-021-00095-5

APPENDIX: SOURCE CODE AND DATASETS

All related source code can be found in the link below:

https://github.com/Bofan1120/Mislabeled_ids/blob/main/notebook/DataShapleyIDS_all_IDS_1.ipynb

The dataset can be found here: https://www.unb.ca/cic/datasets/ids-2017.html